**EKONOMICKÁ UNIVERZITA V BRATISLAVE**
**FAKULTA HOSPODÁRSKEJ INFORMATIKY**

# KOMPARÁCIA ALGORITMOV PREHĽADÁVANIA STAVOVÉHO PRIESTORU NA KONKRÉTNOM PRÍKLADE

**Diplomová práca**

**Študijný program:** Manažérske rozhodovanie a informačné technológie

**Študijný odbor:** 3.3.24 Kvantitatívne metódy v ekonómii

**Školiace pracovisko:** Katedra aplikovanej informatiky FHI

**Vedúci záverečnej práce:** RNDr. Eva Rakovská, PhD.

**Bratislava 2012** **Bc. Jozef Beňo**

**Čestné vyhlásenie**

**Čestne vyhlasujem, že záverečnú prácu som vypracoval samostatne a že som uviedol všetku použitú literatúru.**

**Dátum: 18.6.2012** **.....**

**Poďakovanie**

Na tomto mieste by som sa rád poďakoval vedúcej diplomovej práce, RNDr. Eve Rakovskej, PhD. za cenné rady o odbornú pomoc a tiež mojím najbližším za trpezlivosť a podporu.

# Abstrakt

BEŇO, Jozef: *Komparácia algoritmov prehľadávania stavového priestoru na konkrétnom príklade.* – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; katedra aplikovanej informatiky. – Vedúci záverečnej práce: RNDr. Eva Rakovská, PhD. – Bratislava: FHI EU, 2012, 90 s.

Cieľom záverečnej práce je porovnať algoritmy prehľadávania stavového priestoru pri ich použití vo vyhľadávaní trasy na dvojrozmernej rastrovej mape s definovanými obmedzujúcimi vlastnosťami. Študované vlastnosti sú: terén, nadmorská výška a zdroje, ktoré sú nevyhnutné na pohyb a ktorých množstvo je limitované tak, že trasa musí v určitých intervaloch prechádzať bodom s prítomným zdrojom. Dôsledkom zavedenia zdrojov je vyššia možnosť neexistencie trasy medzi začiatkom a cieľom a možnosť existencie slučiek v optimálnej trase. Práca je rozdelená do troch základných kapitol. Prvá kapitola je venovaná stavovým priestorom a algoritmom hľadania cieľa v nich. Podrobne sa zaoberáme neinformovaným a informovaným prehľadávaniami. V ďalšej časti sa analyzujú rôzne typy máp a spôsoby hľadania cesty v nich s použitím algoritmov prehľadávania stavového priestoru. Hlavná časť kapitoly sa sústreďuje na rastrové mapy, kde sa detailne venujeme rozširujúcim vlastnostiam mapy: terénu, nadmorskej výške a zdrojom. Následne navrhujeme úpravy algoritmov, aby bolo možné generovať trasu aj pri limitovaných zdrojov. Záverečná kapitola obsahuje výsledky pokusov a ich vyhodnotenie.

**Kľúčové slová: prehľadávanie stavového priestoru, hľadanie trasy, a*, heuristika**

# Abstract

BEŇO, Jozef: *Comparison of state space search algorithms on a particular example* – The University of Economics in Bratislava. Faculty of Economic Informatics; Department of Applied Informatics. – Supervisor: RNDr. Eva Rakovská, PhD. – Bratislava: FHI EU, 2012, 90pp.

The aim of the thesis is to compare the state space search algorithms for their use in the search path for two-dimensional raster map with defined limiting properties. Properties studied are: terrain, altitude and resources necessary to move and whose quantity is limited, so that the route must pass through at certain intervals with pre-existing point sources. The consequence is the introduction of resources and increase of probability of path nonexistence or the possibility of loops in the optimal route. The work is divided into three main chapters. The first chapter is devoted to state space search algorithm and targets them. Uninformed and informed searches are discussed in detail. The next section analyzes different types of maps and finding their way in using state space search algorithms. The main part of the chapter is devoted to the raster maps, which provide detailed explanation of expanding properties of maps: terrain, altitude and resources. Consequently, we suggest algorithms modifications in order to generate a route with limited resources. The final chapter contains results of the experiments and their evaluation.

**Keywords: state space search, path finding, raster map, a star, heuristics**

# Obsah

# Zoznam ilustrácií a tabuliek

# Úvod

Prehľadávanie stavového priestoru je v umelej inteligencii široko používaný prístup na riešenie problémov. Jednou z najpoužívanejších oblastí aplikácie je hľadanie trasy. Väčšina prác venovaných problematike počíta aj s nákladmi, tiež označovanými ako cena, ktoré je potrebné vynaložiť na prechod trasou. Cieľom je nájsť trasu s minimálnymi nákladmi alebo nájsť "dostatočne dobrú" trasu za použitia čo najmenších výpočtových a pamäťových zdrojov.

V tejto sa práci rozširujeme uvedený koncept. Pohyb na mape obmedzíme nutnosťou mať k dispozícii zdroje, ktoré je možné dopĺňať len v definovaných miestach mapy a ktoré pri pohybe klesajú resp. aspoň nerastú. Príkladom z reálneho sveta je napr. cestovanie automobilom, kde je potrebné v určitých intervaloch dočerpať palivo alebo oddýchnuť si. V uvedenom príklade je viditeľná aj existencia limitu na množstvo zdroja, objem nádrže, ktorý sa v práci sa označuje ako kapacita.

Zavedenie nového prvku do priestoru hľadania trasy má výrazný vplyv na jej vlastnosti, mení sa aj topológia trasy. Ako si ukážeme, na rozdiel od štandardných modelov, prináša podmienka na zdroje možnosť slučiek aj v optimálnej trase resp. v trase najkratšej dĺžky. Slučka môže mať tvar vrátenia sa po určitom úseku späť.

Cieľom práce je analyzovať model so zdrojmi, upraviť algoritmy tak, aby v ňom našli trasu a porovnať ich medzi sebou resp. porovnať kvalitu heuristík.

# 1 Súčasný stav riešenej problematiky doma a v zahraničí

## 1.1 Stavový priestor

Stav je úplný opis sveta, pričom úplnosť je braná z pohľadu riešenia určitého problému, t.j. v modeli sa abstrahuje od nedôležitých vlastností. Stavový priestor je množina platných stavov sveta.

Stavový priestor je v dynamických systémoch opísaný pomocou orientovaného grafu a množiny počiatočných a koncových stavov.

Každý stav systému je reprezentovaný práve jedným vrcholom grafu a možné prechody medzi stavmi sú reprezentované hranami, t.j. ak zo stavu A existuje priamy prechod do stavu B, tak existuje aj hrana orientovaná od vrcholu pre stav A do vrcholu stavu B. Akcia transformujúca jeden stav na druhý sa označuje ako operátor. Stavy, ktoré nemajú výstupné hrany, sú koncové stavy.

### *1.1.1 Formálna definícia stavového priestoru*

Stavový priestor je štvorica $[N, A, S, G]$, kde:

- N je množina vrcholov (stavov),
- A je množina hrán spájajúcich vrcholy,
- S je neprázdna podmnožina množiny N, ktorá obsahuje začiatočné stavy,
- G je neprázdna podmnožina množiny N, ktorá obsahuje cieľové stavy.

Pre ľubovoľný vrchol z množiny N, t.j. pre ľubovoľný stav, a teda aj pre cieľové stavy, musí existovať (aspoň jedna) cesta z niektorého zo začiatočných vrcholov/stavov.

Pre šach je stavovým priestorom množina všetkých možností uloženia figúrok na šachovnici, ku ktorým je možné sa dopracovať platnými ťahmi. Platné ťahy podľa pravidiel sú operátormi stavového priestoru šachu.

Cieľ môže byť definovaný explicitne ako množina stavov alebo pomocou predikátu. Test dosiahnutia cieľa sa vykoná cieľovým testom, ktorý P. Návrat (2001) opisuje ako „hodnotenie nejakého stavu (stavov) alebo ako hodnotenie cesty v stavovom

priestore.“ Ďalej definuje zadanie problému, ktorý sa rieši prehľadávaním stavového priestoru, ako päticu obsahujúcu:

- stavy;
- začiatočný stav;
- operátory;
- cieľový test;
- cenu cesty.

## 1.2 Prehľadávanie stavového priestoru

Prehľadávanie stavového priestoru je proces postupného zvažovania stavov za účelom dosiahnutia cieľového stavu. Ide o nájdenie cesty v grafe stavového priestoru medzi začiatočným stavom a cieľovým stavom. Pri prehľadávaní stavového priestoru sa vychádza zo začiatočného stavu a operátormi sa generujú nové stavy existujúcich stavov (vygenerovaných a počiatočného stavu). Pri prehľadávaní sa ako začiatočný stav môže použiť aj cieľový stav a hľadať cestu k začiatočnému stavu. Môže byť však problém nájsť inverzné operátory alebo nájsť stavy, z ktorých sa použitím operátora vygeneruje skúmaný stav.

#### *1.2.1.1 Všeobecný algoritmus a strom prehľadávania*

Všeobecný algoritmus prehľadávania stavového priestoru je postavený na postupnom prechádzaní stavov, pričom sa začína z jedného (začiatočného) stavu a aplikovaním operátorov sa identifikujú ďalšie stavy. V algoritme sa používajú množiny stavov:

- množina, resp. presnejšie zoznam vygenerovaných stavov, ktoré čakajú na preskúmanie,
- množina preskúmaných stavov, (ktoré nie sú cieľové).

V jednom kroku algoritmu sa spracúva jeden (práve skúmaný) stav, t.j.

1. Testuje sa, či je tento stav cieľový.
2. Aplikujú sa operátory a generujú sa nové stavy, ktoré sa zaradia (určitým spôsobom) do zoznamu čakajúcich stavov.

a. Hĺbka týchto nových stavov v strome prehľadávania je o jednu vyššia ako hĺbka stavu, z ktorého sú generované.

3. Stav sa uloží do preskúmaných stavov.
4. Vyberie sa nový stav na skúmanie z prvého miesta zoznamu čakajúcich stavov.

```
DECLARE
  zOtvorené  STROM[1] /*prvok zoznamu: minimálne stav, prípadne iné informácie */
  zUzavreté  STROM
  sSpracovavany PRVOK /*prvok zoznamu/vrchol stromu*/
ZAČIATOK
  zOtvorené.PRIDAJ(začiatočnýStav)
  KÝM zOtvorene.NieJePrazdny TAK
     /*zoznam nie je prázdny */
     sSpracovavany = zOtvorene.VyberPrvy /*odstráni stav prvý v poradí a vráti
ho*/
     AK (sSpracovany je sCielovy) TAK VRÁT(sSpracovany)

     /* spracovávaný stav nie je cieľový, generujú sa potomkovia, pričom nie je
       explicitne definované ich poradie */
     zOtvorené.PRIDAJ(sSpracovanany.Stav.GenerujPotomkov)
     zUzavreté.PRIDAJ(sSpracovanany)
  KONIEC CYKLU
  VRÁT(NIČ) /*neúspech, nenašla sa cesta*/
KONIEC
```

**Alg. 1 Všeobecný algoritmus prehľadávania stavového priestoru**

Každý stav je zaradený v zozname, resp. má väzbu na predchádzajú a nasledujúci prvok a zároveň aj miesto v strome, t.j. má väzbu na predka, teda na stav, z ktorého sa tento stav vygeneroval. Predchádzajúci prvok v zozname nemusí byť predok v strome. Potrebné je však, aby pri vyberaní prvku zo zoznamu zostal strom konzistentný, resp. nedovoliť vybrať prvok zoznamu, pokiaľ má v strome potomka, a aj naopak, pre odoberanie vrcholov stromu. Operácia PRIDAJ tak musí zaradiť stavy alebo prvky do zoznamu a zároveň aj do stromu.

Výsledok vyhľadávania je v prípade úspechu cieľový stav, resp. list stromu prehľadávania. V najjednoduchšom prípade postačí len informácia o stave, t.j. že je možné dosiahnuť cieľový stav a ktorý stav to je. Ak je potrebné zistiť aj cestu k cieľu, tak zoznamy stavov musia mať podobu stromov, takže prvok zoznamu reprezentuje vrchol stromu prehľadávania a okrem stavu obsahuje aj referenciu na predka, čiže na stav, z ktorého sa stav vygeneroval použitím určitého operátora.

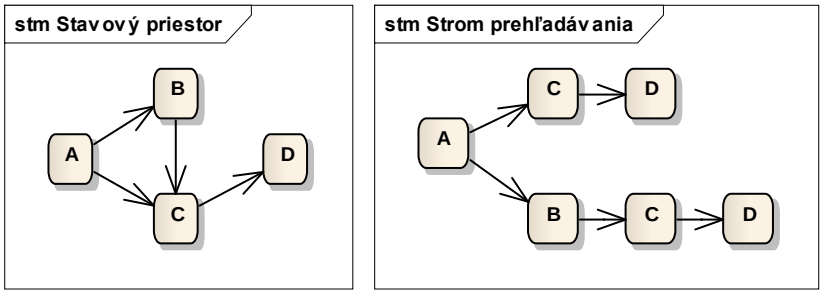
Pokiaľ je výstupom algoritmu postupnosť operátorov, ktorými sa vytvorí cesta stavovým priestorom zo začiatočného stavu do cieľového, tak strom prehľadávania môže obsahovať vo vrchole okrem stavu a referencie na predchádzajúci stav aj operáciu, ktorou sa daný stav vygeneroval z predchádzajúceho stavu.

[1] Typ STROM je zároveň aj ZOZNAM (implementuje oba interface-y)

Spôsob zaraďovania nových stavov do zoznamu stavov čakajúcich na spracovanie označíme v algoritmoch pojmom „politika“. Konkrétne algoritmy prehľadávania aplikujú rôzne politiky a môžu vyžadovať evidenciu doplnkových parametrov v každom vrchole stromu prehľadávania.

### *1.2.1.2 Strom prehľadávania*

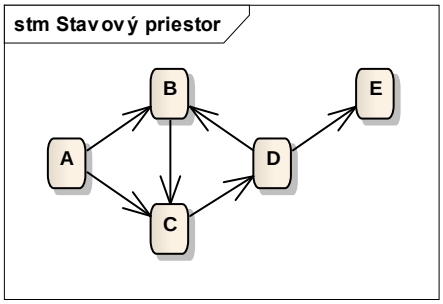
Strom prehľadávania je graf opisujúci postup prehľadávania stavového priestoru zo začiatočného stavu. Koreňom stromu prehľadávania je stav, z ktorého sa začína prehľadávanie. Vo všeobecnom algoritme sú zoznamy stavov, resp. vrcholov evidované aj ako stromy, aby bolo možné vyskladať cestu od nájdeného cieľového k začiatočnému stavu.



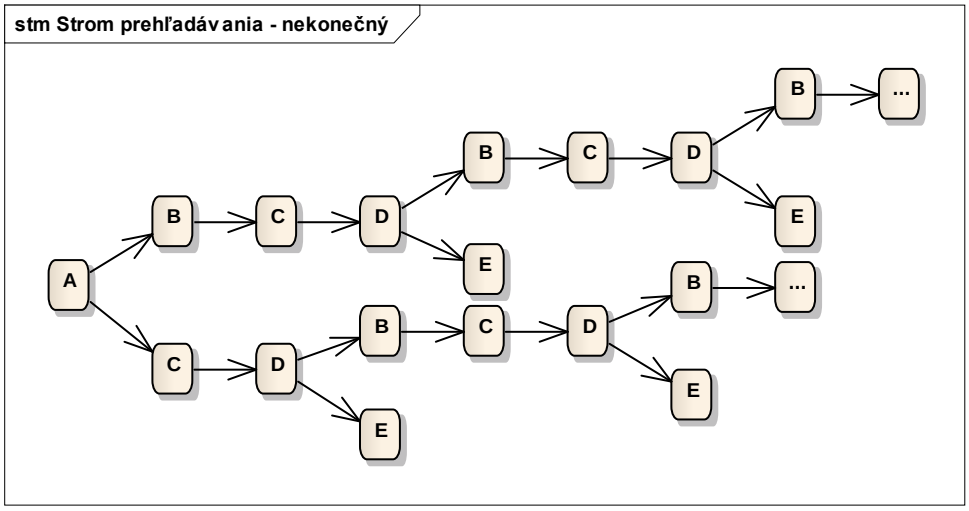

**Obr. 1 Príklad stavového priestoru a jeho stromu prehľadávania**

Na obrázku Obr. 1 je uvedený stavový priestor a strom prehľadávania pre prípad, keď začiatočný stav je stav A stavového priestoru.

Zatiaľ čo stavový priestor je orientovaný graf, ktorý môže obsahovať aj cykly, tak strom prehľadávania cykly neobsahuje, ide o graf typu strom. Môže byť však nekonečný, a to práve v prípade cyklu v stavovom priestore.




**Obr. 2 Stavový priestor s cyklom**




**Obr. 3 Nekonečný strom prehľadávania pri SP s cyklom**

Pokiaľ ani jeden stav nie je cieľový, tak všeobecným algoritmom sa pre stavový priestor na obrázku Obr. 2 vytvorí nekonečný strom prehľadávania na obrázku Obr. 3.

### *1.2.1.3 Optimalizácia základného algoritmu*

Aby sa predišlo nekonečným cyklom, resp. nekončeným stromom, tak sa pri pridávaní nových stavov do otvoreného zoznamu kontroluje, či už pridávaný stav neexistuje v zozname otvorených alebo v zozname uzavretých stavov. Kontrola v zozname otvorených stavov nemá dopad na odstránenie nekončených cyklov, je doplnená do algoritmu za účelom zrýchlenia prehľadávania.

```
DECLARE
  zOtvorené  STROM
  zUzavreté  STROM
  sSpracovavany PRVOK
ZAČIATOK
  zOtvorené.PRIDAJ(začiatočnýStav)
  KÝM zOtvorene.NieJePrazdny TAK
     sSpracovavany = zOtvorene.VyberPrvy
     AK (sSpracovany je sCielovy) TAK VRÁT(sSpracovany)

     PRE KAŽDÝ STAV sStav V (sSpracovanany.Stav.GenerujPotomkov)
       AK zUavrete.NEOBSAHUJE(sStav) A zOtvorené.NEOBSAHUJE(sStav) TAK
         zOtvorené.PRIDAJ(sStav)
     zUzavreté.PRIDAJ(sSpracovanany)
  KONIEC CYKLU
  VRÁT(NIČ)
KONIEC
```

**Alg. 2 Všeobecný algoritmus prehľadávania stavového priestoru**

Strom prehľadávania je pri tomto algoritme závislý od zaradenia potomkov spracovávaného stavu do zoznamu otvorených stavov ako aj ich poradia medzi sebou.

### 1.2.2 Oblasti využitia

Okrem oblasti, ktorá je naším primárnym záujmom má prehľadávanie stavového priestoru mnoho ďalších využití. G. Adam (2011) uvádza tieto použitia:

- riešenie problémov (hlavolamy, hry ako šach, apod. optimalizácia programov),
- logické dokazovanie (dokazovanie predpokladov manipuláciou databázy faktov),
- jazyk (porozumenie prirodzenému jazyku, preklad, kontrola chýb),
- programovanie (tvorba počítačových programov podľa popisu),
- učenie (učenie sa z príkladov),
- expertné systémy (interakcia používateľa s ES pomocou dialógu),
- robotika a vnímanie (manipulačné robotické zariadenia najmä z priemyslu na vykonávanie opakovaných činností, rozpoznávania objektov a tieňov v obrazových scénach),
- systémy a jazyky.

Medzi ďalšie použitia prehľadávania stavového priestoru možno zaradiť aj hľadanie najkratšej cesty alebo problém obchodného cestujúceho. (Návrat, 2001)

### 1.2.3 Vlastnosti algoritmov

Šnajder (2012) a Williams (2003) popisujú vlastnosti algoritmov prehľadávania stavového priestoru takto:

**Tab. 1 Vlastnosti algoritmov prehľadávania**

| Vlastnosť | Popis |
|---|---|
| Spoľahlivosť | Do akej miery je výsledok vrátený algoritmom správny. |
| Úplnosť | Do akej miery je algoritmus schopný nájsť riešenie, pokiaľ riešenie existuje. |
| Optimálnosť | Do akej miery vracia algoritmus optimálny výsledok (najnižšie náklady/cena). |
| Časová komplexnosť | Náročnosť na výpočtový čas, počet vygenerovaných vrcholov. |
| Priestorová zložitosť | Náročnosť na pamäť, počet uchovaných vrcholov. |

### 1.2.4 Triedy algoritmov prehľadávania SP

Algoritmy prehľadávania stavového priestoru sa delia (podľa Williamsa (2003)) na:

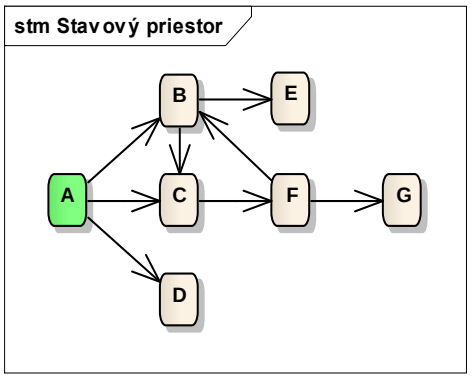
- prehľadávanie naslepo/neinformované;
- prehľadávanie s heuristikou/informované.

## 1.3 Prehľadávanie naslepo

Prehľadávanie naslepo, angl. „Blind Search“, je prehľadávanie stavového priestoru, ktoré využíva len definované znalosti o stavovom priestore, t.j. len počiatočné stavy a operátory. Patria sem nasledovné algoritmy:

- Prehľadávanie najskôr do hĺbky (angl. Depth-first);
- Prehľadávanie najskôr do šírky (angl. Breadth-first);
- Postupné prehlbovanie (angl. Iterative deepening);
- Rovnomerné náklady (angl. Uniform cost);
- Obojsmerné prehľadávanie.

V nasledujúcich kapitolách, ktoré sú venované podrobnejšiemu opisu vymenovaných algoritmov, sa použije stavový priestor na obrázku Obr. 4.




**Obr. 4 Stavový priestor**

Stav A je začiatočný stav. Cieľový stav nie je určený z toho dôvodu, aby algoritmy prehľadávania preskúmali celý stavový priestor.

### *1.3.1 Prehľadávanie najskôr do hĺbky*

Pri prechádzaní stromom sa vyberajú najprv stavy na nižších úrovniach stromu prehľadávania. Poradie potomkov medzi sebou nie je definované a zostáva na implementácii.

```
DECLARE
  zOtvorené  STROM
  zUzavreté  STROM
  sSpracovavany PRVOK
ZAČIATOK
  zOtvorené.PRIDAJ(začiatočnýStav)
  KÝM zOtvorene.NieJePrazdny TAK
     sSpracovavany = zOtvorene.VyberPrvy
     AK (sSpracovany je sCielovy) TAK VRÁT(sSpracovany)

     PRE KAŽDÝ STAV sStav V (sSpracovanany.Stav.GenerujPotomkov)
       AK zUavrete.NEOBSAHUJE(sStav) A zOtvorené.NEOBSAHUJE(sStav) TAK
         zOtvorené.PRIDAJ_NA_ZACIATOK(sStav)
     zUzavreté.PRIDAJ(sSpracovanany)
  KONIEC CYKLU
  VRÁT(NIČ)
KONIEC
```

**Alg. 3 Algoritmus Prehľadávanie najskôr do hĺbky („Depth First Search“)**

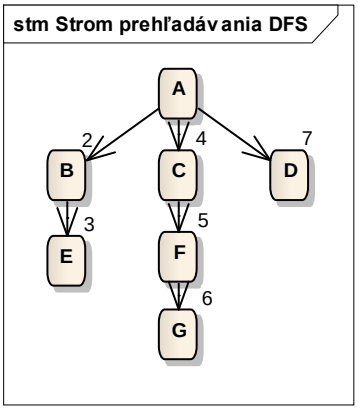
Zoznam otvorených vrcholov sa správa ako zásobník, t.j. fakticky sa používa LIFO (angl. „Last In First Out“) prístup k zoznamu.

Pri generovaní stromu prehľadávania pre stavový priestor (Obr. 4 Stavový priestor) a iterácií sa postupovalo tak, že poradie potomkov (pridávaných stavov) bolo určené abecedne z názvu stavu.

**Tab. 2 Postup prehľadávania priestoru algoritmom Prehľadávanie najskôr do hĺbky**

| Iter. | Sprac. | Otvorené | Uzavreté |
|---|---|---|---|
| - | - | A | - |
| 1 | A | B, C, D | A |
| 2 | B | E, C, D | A, B |
| 3 | E | C, D | A, B, E |
| 4 | C | F, D | A, B, E, C |
| 5 | F | G, D | A, B, E, C, F |
| 6 | G | D | A, B, E, C, F, G |
| 7 | D | - | A, B, E, C, F, G, D |

Pri spracovávaní stavu F algoritmus identifikoval, že stav B už je v zozname spracovaných, a preto ho nepridal do zoznamu otvorených stavov.




**Obr. 5 Strom prehľadávania pre algoritmus Prehľadávanie najskôr do hĺbky**

Pokiaľ by sa nekontrolovalo, či existuje potomok v zozname spracovaných a otvorených stavov, tak je možné prehľadávanie implementovať aj pomocou rekurzívnej funkcie. Cesta sa potom ukladá do systémového zásobníka. Je tu však veľké riziko zacyklenia a opakovaného prehľadávania určitého stavu, keďže nie je možné zistiť, či sa daný stav už nespracovával.

```
FUNKCIA DepthFirstSearch(pStav Stav) VRAT ZOZNAM
  vZoznam  ZOZNAM
ZAČIATOK
  AK (pStav je sCielovy) TAK VRÁT ZOZNAM(pStav) -- založí zoznam
  PRE KAŽDÝ STAV sStav V (sSpracovanany.Stav.GenerujPotomkov)
    vZoznam = DepthFirstSearch(sStav)
    AK vZoznam NIE JE NIČ TAK VRAT(vZoznam.PRIDAJ(sStav))
  KONIEC CYKLU
  VRÁT(NIČ)  -- neúspech, ani jeden potomok nie je cieľový stav
KONIEC
/* Inicializácia */
ZAČIATOK
  DepthFirstSearch(zaciatočnýStav)
KONIEC
```

**Alg. 4 Rekurzívny algoritmus Prehľadávanie najskôr do hĺbky**

Ak označíme priemerný počet potomkov vrcholu písmenom b a písmenom m maximálnu hĺbku stromu prehľadávania, tak priestorová náročnosť je definovaná vzťahom bm. Časovú náročnosť vyjadruje vzťah $b^m$. Pokiaľ je riešenie v hĺbke d stromu prehľadávania, pričom d ≪m, tak algoritmus má vysokú časovú náročnosť. Algoritmus negarantuje, že nájdené riešenie je optimálne.

V prípade, že strom prehľadávania je nekonečný, tak algoritmus nemusí nájsť riešenie, ktoré existuje v konečnej hĺbke, v prípade, ak prehľadáva inú nekonečnú vetvu. Algoritmus tak nie je úplný.

### *1.3.2 Prehľadávanie najskôr do šírky*

Pri prechádzaní stromom prehľadávania sa najprv spracujú všetky vrcholy na jednej úrovni a až potom sa začnú spracovávať vrcholy na nižšej úrovni. Ide o FIFO (angl. „First In First Out“) prístup k zoznamu, čiže zoznam otvorených vrcholov pracuje ako rad.

```
DECLARE
  zOtvorené  STROM
  zUzavreté  STROM
  sSpracovavany PRVOK
ZAČIATOK
  zOtvorené.PRIDAJ(začiatočnýStav)
  KÝM zOtvorene.NieJePrazdny TAK
     sSpracovavany = zOtvorene.VyberPrvy
     AK (sSpracovany je sCielovy) TAK VRÁT(sSpracovany)

     PRE KAŽDÝ STAV sStav V (sSpracovanany.Stav.GenerujPotomkov)
       AK zUavrete.NEOBSAHUJE(sStav) A zOtvorené.NEOBSAHUJE(sStav) TAK
```

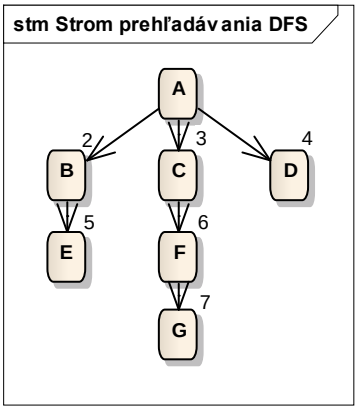
```
          zOtvorené.PRIDAJ_NA_KONIEC(sStav)
      zUzavreté.PRIDAJ(sSpracovanany)
  KONIEC CYKLU
  VRÁT(NIČ)
KONIEC
```

**Alg. 5 Algoritmus Prehľadávanie najskôr do šírky („Breath First Search“)**

Potomkovia spracovávaného stavu sa do zoznamu otvorených vrcholov radili podľa ich abecedného poradia.

**Tab. 3 Postup prehľadávania priestoru algoritmom Prehľadávanie najskôr do šírky**

| Iter. | Sprac. | Otvorené | Uzavreté |
|---|---|---|---|
| - | - | A | - |
| 1 | A | B, C, D | A |
| 2 | B | C, D, E | A, B |
| 3 | C | D, E, F | A, B, C |
| 4 | D | E, F | A, B, C, D |
| 5 | E | F | A, B, C, D, E |
| 6 | F | G | A, B, C, D, E, F |
| 7 | G | - | A, B, C, D, E, F, G |




**Obr. 6 Strom prehľadávania pre algoritmus Prehľadávanie najskôr do šírky**

Algoritmus je úplný. Tým, že sa prehľadávajú najprv všetky stavy na jednej úrovni, predtým ako sa skúmajú stavy na nižšej úrovni, nájdená cesta je najkratšia cesta. Za predpokladu, že každý prechod medzi stavmi je rovnako nákladný, je algoritmus aj optimálny.

Pri generovaní stromu prehľadávania sa vygenerujú všetci potomkovia skúmaného stavu, až kým sa nenarazí na cieľový stav. Pokiaľ sa označí písmenom „b“ priemerne počet stavov, do ktorých sa dá dostať zo stavu stavového priestoru a písmenom „d“ hĺbka, v ktorej sa nájde cieľový stav, tak počet vrcholov, ktoré algoritmus BFS vygeneruje, je daný vzťahom $1 + b + b^2 + b^3 + ... + b^d + b^{d+1}$, takže jeho hlavná zložka je $b^{d+1}$.

Priestorová náročnosť je najväčší problém algoritmu, už aj pre jednoduché úlohy prudko, exponenciálne narastá spotreba priestoru. Napr. pokiaľ má vrchol priemerne 6

susedných stavov (stavov, do ktorých existuje hrana) a cieľový vrchol je v hĺbke 15, k cieľu je potrebné prejsť 15 stavmi, a tak je potrebné vygenerovať približné $6^{16}$ vrcholov, t.j. približne 129 629 238 163 050 258 624 287 932 416 vrcholov ($10^{30}$). Už pri 10B (bytoch) na jeden vrchol ide o desiatky terabytov. Prehľadávanie najskôr do šírky je preto použiteľné len pre malé problémy.

### *1.3.3 Postupné prehlbovanie*

Pri postupnom prehlbovaní prehľadávania sa používa prehľadávanie do hĺbky, pričom sa hĺbka obmedzuje na určitú hodnotu, ktorá sa postupne zväčšuje. Kombinujú sa tak výhody algoritmov „Depth First Search“ a „Breath First Search“.

```
FUNKCIA LimitedDepthFirstSearch(maxHĺbka NUMBER, začiatočnýStav STAV) VRÁŤ STAV
  zOtvorené  STROM
  sSpracovavany PRVOK
ZAČIATOK
  zOtvorené.PRIDAJ(začiatočnýStav)
  KÝM zOtvorene.NieJePrazdny TAK
     sSpracovavany = zOtvorene.VyberPrvy
     AK (sSpracovany je sCielovy) TAK VRÁT(sSpracovany)
     AK (sSpracovavany < maxHĺbka) TAK
       /* hĺbka sa inkremenuje pri generovaní potomkov */
       zOtvorené.PRIDAJ_NA_ZACIATOK(sSpracovanany.Stav.GenerujPotomkov)
  KONIEC CYKLU
  VRÁT(NIČ)
KONIEC
DECLARE
  sStav  STAV
ZAČIATOK
  PRE hĺbka OD 1 DO NEKONEČNO
    sStav = LimitedDepthFirstSearch(hĺbka, začiatočnýStav)
    AK sStav NIE JE NIČ TAK VRÁT sStav
  KONIEC CYKLU
KONIEC
```

**Alg. 6 Rekurzívny algoritmus Postupné prehlbovanie („Iterative Deepening“)**

Zjavná nevýhoda algoritmu je v tom, že stavy pri vrchole stromu prehľadávania sú spracovávané mnohokrát. Keďže však počet stavov rastie exponenciálne s hĺbkou, tak spracovanie stavov v nízkych hĺbkach výrazne nezhorší celkový čas spracovania.

Časovú náročnosť možno vyjadriť vzťahom: $(d+1)1 + db + (d-1)b^2 + ... 3b^{d-2}+2b^{d-1} + 1b^d$. V porovnaní s algoritmom Prehľadávanie najskôr do šírky, ktorého časová náročnosť je určená výrazom $b^{d+1}$, sa ušetrí posledná úroveň prehľadávania.

Priestorovú náročnosť znižuje fakt, že sa pri expanzii stavu nekontroluje, či novo vygenerovaný stav bol už spracovávaný. Keďže hĺbka prehľadávania je obmedzená, tak nemôže dôjsť k nekonečnému zacykleniu. Neudržiava sa tak zoznam spracovaných stavov.

Počet vygenerovaných stavov v zozname otvorených je tak závislý len od priemerného počtu potomkov stavu a hĺbky stromu, v ktorej sa nachádza cieľový stav.

### *1.3.4 Prehľadávanie s rovnomernými nákladmi*

Prehľadávanie podľa nákladov vychádza zo všeobecného algoritmu. Veličina Náklady udáva náročnosť prechodu medzi stavmi v sledovanej veličine, ktorá je pre kvalitu výsledku kľúčová. Nákladmi môžu byť čas, palivo, energia, atď.

Pri generovaní potomkov sa počítajú celkové náklady na prechod zo začiatočného stavu do potomka spracovávaného stavu a zoznam otvorených stavov je zoradený podľa tejto miery, pričom sa na spracovanie vyberá vždy vrchol s najmenšími celkovými nákladmi.

Pri kontrole existencie stavu medzi spracovanými a otvorenými stavmi je potrebné brať do úvahy aj celkové náklady a v prípade potreby vymeniť stav v otvorených, keďže do stavu môže viesť viacero ciest s rôznymi nákladmi. Vyradenie stavu zo zoznamu otvorených neovplyvní výsledok algoritmu, ale ušetrí priestor potrebný na evidenciu zoznamu otvorených, nespracovaných stavov.

```
DECLARE
  zOtvorené  STROM
  zUzavreté  STROM
  sSpracovavany PRVOK
ZAČIATOK
  zOtvorené.PRIDAJ(začiatočnýStav)
  KÝM zOtvorene.NieJePrazdny TAK
     sSpracovavany = zOtvorene.VyberPrvy
     AK (sSpracovany je sCielovy) tak VRÁT(sSpracovany)

     PRE KAŽDÝ STAV sStav V (sSpracovanany.Stav.GenerujPotomkov)
         ZORAĎ PODĽA CelkovéNáklady
       sStav.CelkovéNáklady =
          sSpracovavany.CelkovéNáklady + NÁKLADY(sSpracovávaný, sStav)
       AK zUavrete.NEOBSAHUJE(sStav) TAK
         AK zOtvorene.NEOBSAHUJE(sStav) TAK
           zOtvorené.PRIDAJ_PODLA_NAKLADOV(sStav)
         INAK
           AK zOtvorené.DAJ_STAV(sStav).CelkovéNáklady > sStav.CelkovéNáklady TAK
             zOtvorené.NAHRAĎ(sStav)
     KONIEC CYKLU
     zUzavreté.PRIDAJ(sSpracovanany)
  KONIEC CYKLU
  VRÁT(NIČ)
KONIEC
```

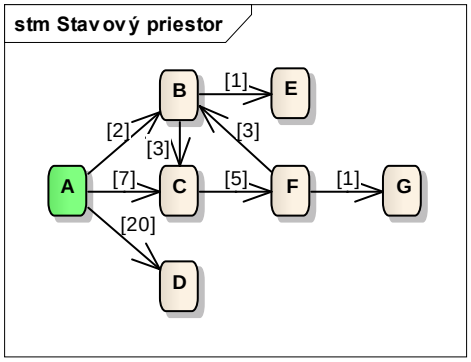
**Alg. 7 Algoritmus Prehľadávanie s rovnomernými nákladmi („Uniform Cost Search“)**

V zozname spracovaných, zatvorených stavov stačí evidovať iba existenciu uzla, pretože nie je možné nájsť kratšiu cestu do stavu, ktorý je už spracovaný. Ak sa radia stavy do zoznamu otvorených stavov podľa celkových nákladov a do zoznamu spracovaných

stavov sa zapisuje vždy uzol s najnižšou cenou z otvorených, tak platí, že maximum nákladov na prechod zo začiatočného stavu v zozname zatvorených je menšie alebo rovné minimu nákladov v zozname otvorených.

Funkcia `zOtvorené.PRIDAJ_PODLA_NAKLADOV(sStav)` zaradí potomka do zoznamu podľa jeho nákladov, pričom stav s najmenšími nákladmi je zaradený ako prvý. Pri tomto algoritme možno považovať zoznam otvorených stavov za prioritný rad.

V algoritme použitý výraz `zOtvorené.DAJ_STAV(sStav)` vracia záznam zoznamu, ktorý obsahuje identický stav, ako vstupný stav. Tento záznam však môže obsahovať odlišné doplnkové atribúty ako predchádzajúci stav (predok v strome) a celkové náklady.



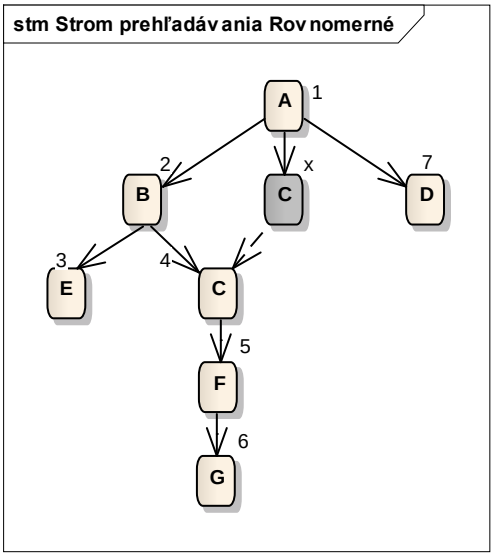

**Obr. 7 Príklad stavového priestoru s nákladmi na prechod medzi stavmi (váhami hrán)**

Výraz `zOtvorené.NAHRAĎ(sStav)` zmení vlastnosti záznamu zoznamu podľa aktuálneho potomka (sStav) alebo odstráni pôvodný záznam a nahradí ho záznamom sStav, ktorý má iné celkové náklady a iného predka v strome prehľadávania.

Pre stavový priestor s váhami hrán (čiže s nákladmi na prechod medzi stavmi) na obrázku Obr. 7, algoritmus prebehne podľa tabuľky Tab. 4. V tabuľke sú celkové náklady uvádzané ako horný index stavu, napr. $F^{11}$ vyjadruje, že cesta zo stavu A (začiatočného stavu) do stavu F má celkové náklady 11.

**Tab. 4 Postup prehľadávania priestoru podľa algoritmu UCS**

| Iter. | Sprac. | Otvorené | Uzavreté |
|---|---|---|---|
| - | - | A | - |
| 1 | $A^0$ | $B^2$, $C^7$, $D^{20}$ | $A^0$ |
| 2 | $B^2$ | $E^3$, $C^5$, $D^{20}$ | $A^0$, $B^2$ |
| 3 | $E^3$ | $C^5$, $D^{20}$ | $A^0$, $B^2$, $E^3$ |
| 4 | $C^5$ | $F^{10}$, $D^{20}$ | $A^0$, $B^2$, $E^3$, $C^5$ |
| 5 | $F^{10}$ | $G^{11}$, $D^{20}$ | $A^0$, $B^2$, $E^3$, $C^5$, $F^{10}$ |
| 6 | $G^{11}$ | $D^{20}$ | $A^0$, $B^2$, $E^3$, $C^5$, $F^{10}$, $G^{11}$ |
| 7 | $D^{20}$ | - | $A^0$, $B^2$, $E^3$, $C^5$, $F^{10}$, $G^{11}$, $D^{20}$ |




**Obr. 8 Strom prehľadávania pre algoritmus Rovnomerné náklady („Uniform Cost Search“)**

V iterácii 2 došlo k zmene zoznamu otvorených stavov, keď algoritmus identifikoval, že do stavu C existuje lacnejšia cesta cez stav B ako priama cesta z A. V zozname otvorených staviv sa tak vymenil záznam $C^7$ za záznam $C^5$, pričom sa zmenila aj pozícia stavu v strome prehľadávania. Na začiatku iterácie bol predok záznamu so stavom C vrchol so stavom A a po zmene (na konci iterácie) bol predok záznamu so stavom C vrchol so stavom B.

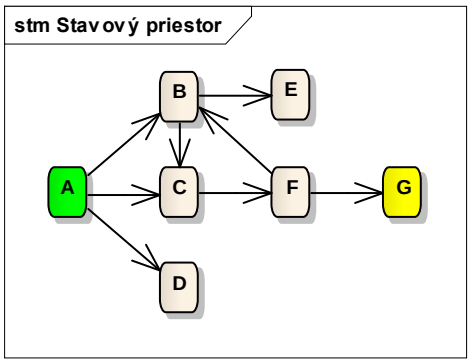
Algoritmus je úplný a optimálny. Pokiaľ v stavovom priestore existuje cesta zo začiatočného do cieľového stavu, tak algoritmus ju nájde a navyše pokiaľ je takých ciest viacero, tak nájde tú s najmenšími nákladmi.

### *1.3.5 Obojsmerné prehľadávanie*

Obojsmerné prehľadávanie je prístup k riešeniu problémov v stavovom priestore, keď sa naraz vykonáva hľadanie cesty zo začiatočného stavu do cieľového a z cieľového do začiatočného. Ak tieto dve hľadania dospejú k prvému spoločnému stavu, tak sa prehľadávanie skončí úspechom.

Vyhľadávanie smerom od cieľa k začiatočnému stavu vyžaduje existenciu inverzných operátorov k operátorom generovania stavov zo smeru začiatočného stavu a explicitnú definíciu cieľového stavu.

Na obrázku Obr. 9 je stavový priestor uvedeného príkladu, v ktorom sme ako cieľ určili stav G. Predpokladá sa, že existuje inverzný operátor, teda že je možné identifikovať stavy, z ktorých existuje prechod do spracovávaného stavu.

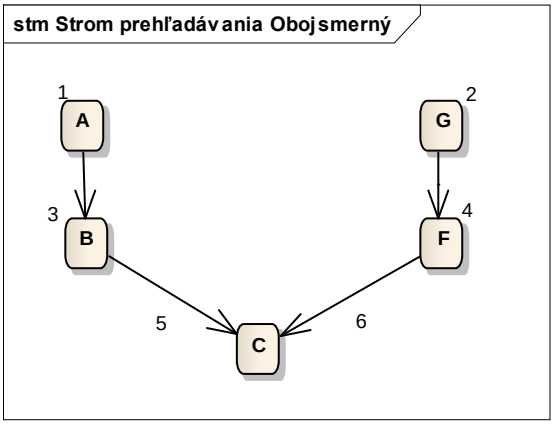



**Obr. 9 Príklad stavového priestoru pre obojsmerné prehľadávanie**

**Tab. 5 Obojsmerné prehľadávanie**

| Iter. | Smer | Sprac. | Otvorené | Uzavreté |
|---|---|---|---|---|
| - | - | - | A | - |
| - | - | - | G | - |
| 1 | Z→C | A | ($B^z$, $C^z$, $D^z$) | $A^z$ |
| 2 | C→Z | G | ($B^z$, $C^z$, $D^z$), ($F^c$) | $A^z$, $G^c$ |
| 3 | Z→C | B | ($C^z$, $D^z$, $E^z$), ($F^c$) | $A^z$, $B^z$, $G^c$ |
| 4 | C→Z | F | ($C^z$, $D^z$, $E^z$), ($C^c$) | $A^z$, $B^z$, $G^c$ |

algoritmus kontroluje aj zoznam otvorených stavov na prienik vygenerovaných a spracovaných stavov z oboch smerov, tak už po štvrtej iterácii je vrátený úspech. Ak by algoritmus porovnával len uzavreté stavy, tak by sa trasa našla v šiestej iterácii, ale v prípade použitia algoritmu Rovnomerných nákladov by išlo o optimálnu trasu.




**Obr. 10 Strom prehľadávania Obojsmerného prehľadávania**

Hĺbka stromu prehľadávania je oproti iným algoritmom nižšia. Cieľ sa nachádza už v tretej úrovni „dvojstromu“.

### *1.3.6 Porovnanie algoritmov*

Nech b je (priemerný) počet potomkov vrcholu grafu, pre stavový priestor, resp. priemerný počet stavov, do ktorých je možné sa z jedného stavu dostať. Táto veličina sa označuje ako faktor grafu. Nech d je hĺbka cieľového stavu v strome prehľadávania, v ktorej sa nachádza cieľový stav, t.j. d je počet stavov, ktorými sa musí prejsť medzi začiatočným a cieľovým stavom. Ak C* sú minimálne náklady na prechod zo začiatočného do cieľového stavu a ε sú minimálne náklady na prechod medzi stavmi, potom $d = \frac{C^*}{\varepsilon}$. Nech m je maximálna hĺbka stromu prehľadávania. Vlastnosti algoritmov sú potom opísané v tabuľke Tab. 6.

**Tab. 6 Porovnanie slepých algoritmov prehľadávania stavového priestoru (Návrat, 2001)**

| Vlastnosť | Najskôr do šírky | Rovnomerné nákl. | Najskôr do hĺbky | Postupné prehlbov. | Obojsmerné |
|---|---|---|---|---|---|
| Úplnosť | Áno | Áno | Nie | Áno | Áno |
| Optimálnosť | Áno | Áno | Nie | Áno | Áno |
| Časová náročnosť | $b^d$ | $b^{C^*/\varepsilon}=b^d$ | $b^m$ | $b^d$ | $b^{d/2}$ |
| Priestorová náročnosť | $b^d$ | $b^{C^*/\varepsilon}=b^d$ | b·m | bd | $b^{d/2}$ |

## 1.4 Prehľadávanie s heuristikou

Heuristické prehľadávanie využíva doplňujúce znalosti o stavovom priestore, resp. o modelovanej skúmanej problematike. Označuje sa preto aj ako informované hľadanie. Používa sa na výber stavu na spracovanie zo zoznamu otvorených stavov, pričom cieľom je vybrať taký stav, ktorý je na hľadanej trase.

Heuristika je informácia, pomocou ktorej sa rozhodujeme medzi viacerými možnosťami konania, ktorá z nich asi najefektívnejšie povedie k cieľu (Návrat, 2001). Ide o pravidlo, ktoré sa odhadlo na základe skúseností, ale ktoré nie je potvrdené pevnou teóriou, ide o poznatok získaný „od oka“ alebo „z brucha“. Heuristické ohodnocovanie stavov preto negarantuje, že vybraný stav je aj v skutočnosti najvhodnejším stavom na spracovanie.

### *1.4.1 Princípy heuristiky*

Heuristika je funkcia, ktorá prideľuje každému stavu stavového priestoru hodnotu, ktorá je odhadom vzdialenosti stavu od cieľového stavu, pričom hodnota funkcie v cieľovom stave je rovná nule.

Označme heuristickú funkciu h, potom h: S → $R^+$. Pre s ϵ G platí h(s) = 0. G je množina cieľových stavov, G ⊆ S.

Zatiaľ čo náklady v algoritme Rovnomerné náklady alebo hĺbka v prehľadávacom strome vyjadrovali vzdialenosť od začiatočného stavu, tak heuristika opisuje vzdialenosť od cieľa. Hĺbka a náklady sú konkrétne presné hodnoty a heuristika je odhad, ktorý slúži na zrýchlenie prehľadávania, keďže sa prioritnejšie môžu spracovať stavy, ktoré sú bližšie k cieľu.

#### *1.4.1.1 Príklady heuristík*

Pri hľadaní cesty v priestore je možné ako heuristickú funkciu použiť vzdušnú, resp. geometrickú vzdialenosť k cieľu. V iných prípadoch môže byť definovanie heuristiky náročnejšie. Bašić a Šnajder (2012) pre 8-hlavolam špecifikujú heuristické funkcie:

- počet zle umiestnených polí;
- súčet vzdialeností polí od ich správneho umiestnenia.

#### *1.4.1.2 Heuristická funkcia*

Heuristická funkcia sa označuje ako optimistická v prípade, ak pre žiadny stav stavového priestoru nemá heuristická funkcia vyššiu hodnotu, ako sú skutočné náklady na prechod do cieľového stavu. Ak skutočné náklady na prechod zo stavu s do cieľového stavu označíme g(s) tak pre optimistickú heuristiku platí:

$$\forall s \epsilon S, h(s) \leq \mathrm{g}(s).$$

Heuristika, pre ktorú platí, že $\forall s \epsilon S, h(s) = \mathrm{g}(s)$, nájde najrýchlejšiu cestu. Ak $\exists s \epsilon S, h(s) > \mathrm{g}(s)$, tak heuristika negarantuje nájdenie najkratšej cesty.

Heuristickú funkciu označujeme ako monotónnu alebo konzistentnú, ak platí:

$$\forall s_2 \, \epsilon \, nasledovníci(s_1), h(s_1) \leq \mathrm{h}(s_2) + d(s_1\text{-}, s_2),$$

kde výraz $d(s_1, s_2)$ označuje hodnotu hrany medzi stavmi $s_1$ a $s_2$, resp. náklady na prechod medzi týmito dvomi stavmi.

Nech $h_1$ a $h_2$ sú dve optimistické heuristické funkcie v jednom stavovom priestore. Heuristika $h_1$ dominuje $h_2$ ,ak platí

$$\forall s \epsilon S, h_1(s) \leq h_2(s).$$

Pokiaľ algoritmus $A_1$ používa heuristiku $h_1$ a algoritmus $A_2$ heuristiku $h_2$, kde $h_1$ dominuje $h_2$, tak hovoríme, že $A_1$ je informovanejší.

Kvalitná heuristika by mala byť aj ľahko vypočítateľná.

### *1.4.2 Algoritmy*

Medzi základné heuristické algoritmy patria:

- Chamtivé hľadanie (angl. Greedy Search);
- A* algoritmus;
- Horolezecký (Hill Climbing) algoritmus.

### *1.4.3 Chamtivé hľadanie*

Chamtivé vyhľadávanie je vlastne rozšírenie prehľadávania Najprv do hĺbky v tom, že medzi vygenerovanými potomkami spracovávaného stavu sa vyberie ten s najlepšou hodnotou heuristickej funkcie, teda ten, o ktorom sa predpokladá, že je najbližšie, resp., že smeruje k cieľu.

Chamtivým sa algoritmus nazýva preto, že sa rozhoduje bez ohľadu na minulosť a dôležitá je len hodnota heuristickej funkcie medzi potomkami aktuálneho stavu.

```
DECLARE
  zOtvorené  STROM
  zUzavreté  STROM
  sSpracovavany PRVOK
ZAČIATOK
  zOtvorené.PRIDAJ(začiatočnýStav)
  KÝM zOtvorene.NieJePrazdny TAK
     sSpracovavany = zOtvorene.VyberPrvy
     AK (sSpracovany je sCielovy) TAK VRÁT(sSpracovany)

     PRE KAŽDÝ STAV sStav V (sSpracovanany.Stav.GenerujPotomkov)
         ZORAĎ PODĽA heuristika VZOSTUPNE
       AK zUavrete.NEOBSAHUJE(sStav) TAK
         AK zOtvorene.NEOBSAHUJE(sStav) TAK
           zOtvorené.PRIDAJ_PODLA_HEURISTIKY(sStav)
         INAK
```

```
            AK zOtvorené.DAJ_STAV(sStav).heuristika > sStav.heuristika TAK
              zOtvorené.NAHRAĎ(sStav)
      KONIEC CYKLU
      zUzavreté.PRIDAJ(sSpracovanany)
  KONIEC CYKLU
  VRÁT(NIČ)
KONIEC
```

**Alg. 8 Algoritmus chamtivého hľadania**

Algoritmus je neoptimálny, nájdené riešenie nemusí byť najlepšie. Časová a priestorová náročnosť je závislá od výrazu bm, kde „b“ je priemerný počet potomkov vrcholu, resp. počet stavov, do ktorých je možný prechod, a „m“ je maximálna hĺbka stromu prehľadávania.

### *1.4.4 A**

Algoritmus A* používa pre výber stavu na spracovanie celkové náklady na prechod od začiatočného stavu a aj heuristickú funkciu odhadujúcu náklady do cieľa. Algoritmus kombinuje vlastnosti algoritmov Chamtivé prehľadávanie a Prehľadávanie s rovnomernými nákladmi.

```
DECLARE
  zOtvorené  STROM
  zUzavreté  STROM
  sSpracovavany PRVOK
ZAČIATOK
  zOtvorené.PRIDAJ(začiatočnýStav)
  KÝM zOtvorene.NieJePrazdny TAK
     sSpracovavany = zOtvorene.VyberPrvy
     AK (sSpracovany je sCielovy) tak VRÁT(sSpracovany)

     PRE KAŽDÝ STAV sStav V (sSpracovanany.Stav.GenerujPotomkov)
         ZORAĎ PODĽA CelkovéNáklady
       sStav.CelkovéNáklady =
          sSpracovavany.CelkovéNáklady + NÁKLADY(sSpracovávaný, sStav)
       sStav.heuristika = HEURISTIKA(sStav, sCielovy)
       sStav.CelkováHodnota = sStav.CelkovéNáklady + sStav.heuristika

       AK zUavrete.NEOBSAHUJE(sStav) TAK
         AK zOtvorene.NEOBSAHUJE(sStav) TAK
           zOtvorené.PRIDAJ_PODLA_HODNOTY(sStav)
         INAK
           AK zOtvorené.DAJ_STAV(sStav).CelkováHodnota > sStav.CelkováHodnota TAK
             zOtvorené.NAHRAĎ(sStav)
     KONIEC CYKLU
     zUzavreté.PRIDAJ(sSpracovanany)
  KONIEC CYKLU
  VRÁT(NIČ)
KONIEC
```

**Alg. 9 Algoritmus „A*“**

Časová a priestorová náročnosť algoritmu je minimum z hodnôt $b^d$ a $b|S|$. Algoritmus nájde optimálne riešenie v prípade, že heuristická funkcia je prípustná. (Návrat,

2001) Dôkaz je podobný dôkazu optimálnosti algoritmu Prehľadávanie s rovnomernými nákladmi.

### *1.4.5 Horolezecký algoritmus*

Horolezecký algoritmus je zjednodušením Chamtivého algoritmu. Pri rozhodovaní sa nekontrolujú už preskúmané stavy, jediná zvažovaná veličina je heuristika.

```
DECLARE
  sSpracovavany STAV
  sMinimálny    STAV
ZAČIATOK
  sSpracovavany = začiatočnýStav
  CYKLUS
    sMinimálny = (sSpracovanany.GenerujPotomkov).DAJ_NAJMENSI_PODLA_HEURISTIKY
    AK sSpracovavany < sMinimálny TAK VRÁŤ sSpracovavany
    sSpracovavany = sMinimálny
  KONIEC CYKLU
```

**Alg. 10 Horolezecký algoritmus**

Algoritmus nie je kompletný a ani optimálny. Veľmi jednoducho sa skončí v lokálnom minime.

# 2 Cieľ práce, metodika práce a metódy skúmania

Cieľom práce je preskúmať rôzne algoritmy a heuristiky prehľadávania stavového priestoru na prípade hľadania trasy v dvojrozmernej mape. Mapa má pridané vlastnosti, ktoré komplikujú nájdenie trasy. Okrem nepriechodných prekážok je pri hľadaní trasy potrebné brať do úvahy aj terén, nadmorskú výšku a aj dostupnosť zdrojov.

Terén reprezentuje unárny modifikátor pohybu, keď pohyb je modifikovaný len jedným stavom pri prechode medzi stavmi, a to cieľovou dlaždicou. Nadmorská výška reprezentuje binárny modifikátor pohybu, keďže pohyb ovplyvňuje rozdiel výšok dlaždíc, medzi ktorými sa vykonáva prechod[2].

Zdroje reprezentujú obmedzenia pohybu, keďže pri pohybe je k dispozícii len obmedzené množstvo zdroja, ktoré sa každým prechodom zmenšuje (resp. nerastie) a je ho potrebné dopĺňať. Doplnenie zdroja do maximálnej kapacity sa vykoná prechodom na stav s priradenou danou vlastnosťou. Možno nimi modelovať napr. vodu alebo palivo, ktoré sa musia každých niekoľko ťahov obnovovať návštevou oázy v púšti alebo čerpacej stanice, pričom pri plánovaní trasy sú známe ich umiestnenia.

Kapitola najprv analyzuje rôzne druhy máp a spôsob hľadania trasy na nich. Ďalej sa venujeme modifikátorom pohybu: zdrojom, terénu a nadmorskej výške. Záver kapitoly patrí špecifikácii modelovaných scenárov, skúmaných algoritmov a heuristík a tiež opisu základných častí aplikácie na porovnanie vybraných algoritmov a heuristík.

## 2.1 Mapy

Na hľadanie optimálnej trasy je možné využiť algoritmy prehľadávania stavového priestoru. Plocha alebo priestor, v ktorom sa hľadá vhodná trasa, môže byť reprezentovaný rôznymi spôsobmi. Základné rozdelenie je podľa toho, či je priestor súvislý alebo diskrétny, t.j. mapy sa delia na:

- vektorové;
- rastrové.

[2] Nadmorská výška sa môže použiť aj ako unárny modifikátor.

### *2.1.1 Vektorové mapy*

Vektorové mapy sú opísané pomocou množiny polygónov, čím sa zvyšuje ich presnosť. Mapy je možné bez straty presnosti približovať. Navigácia v priestore sa najčastejšie realizuje pomocou:

- navigačných bodov (angl. „waypoints“),
- grafu viditeľnosti (angl. „visibility graph“),
- navigačných sietí (angl. „navigation mesh“).

Pri hľadaní trasy sa predpokladá, že náklady na pohyb sú uniformné a sú určené len vzdialenosťou.

#### *2.1.1.1 Navigačné body*

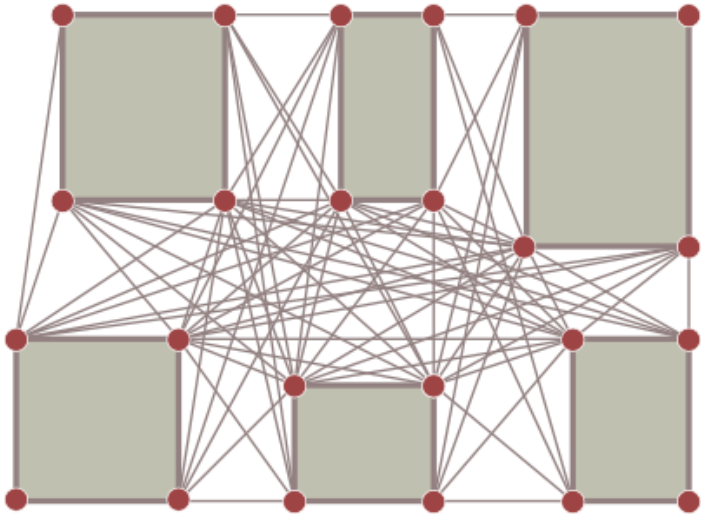
Navigačné body sú preddefinované miesta na mape, ktoré sa zvažujú pri hľadaní trasy, pričom pre navigačný bod sú definované aj susedné body, do ktorých je možný presun. Hľadanie trasy je tak identické s hľadaním cesty v grafe.

Nevýhodou používania navigačných bodov, na ktorú upozorňuje aj Paul Tozour (2008), je neprirodzený tvar nájdenej trasy, resp. neoptimálnosť pohybu po nej v otvorenom priestore. Skvalitniť trasu je možné len za cenu pridávania ďalších bodov, čo však vedie k vysokým pamäťovým nárokom a predlžovaniu výpočtového času.

#### *2.1.1.2 Graf viditeľnosti*

Graf viditeľnosti využíva vrcholy polynómov opisujúcich prekážky alebo nepriechodné plochy. Graf viditeľnosti pre každý vrchol polynómu definuje, do ktorých vrcholov je možné priamo prejsť, t.j. „vidno ich“.

Navigácia pomocou grafu viditeľnosti vychádza z faktu, že najkratšia cesta vedie buď priamo zo štartu do cieľa, alebo vedie cez vrchol alebo vrcholy, rohy prekážok. Nájdená trasa je tak optimálna z hľadiska dĺžky. Problémom je však veľkosť grafu. Pri N vrcholoch môže mať graf až $N^2$.



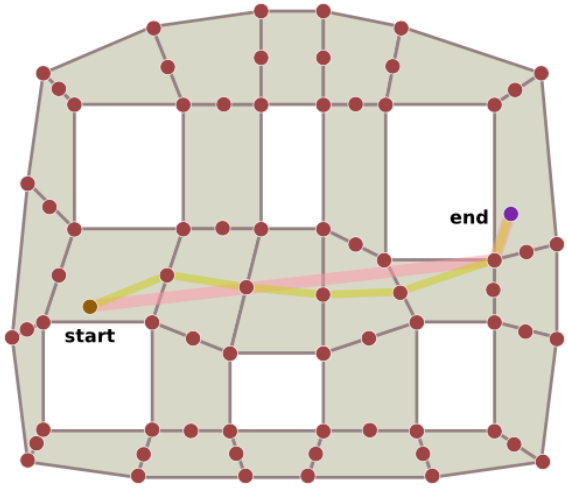
**Obr. 11 Graf viditeľnosti (Amit, 2003)**

Hľadanie trasy je opäť identické s hľadaním trasy v grafe.

### *2.1.1.3 Navigačná sieťovina*

Navigačná sieťovina je množina polygónov, ktoré definujú voľný priestor pre pohyb. Pri hľadaní trasy tak neexistuje obmedzenie pohybu len po definovaných bodoch, ako v prípade použitia navigačných bodov, a v prípade veľkých voľných plôch sú výrazne nižšie aj pamäťové nároky.

Pri hľadaní trasy sa prechádza medzi susednými polygónmi, čiže polygónmi, ktoré zdieľajú hranu alebo vrchol. Amit (2003) uvádza možnosti prechádzania do susedného polygónu cez vrchol a stred hrany (Obr. 12 Navigačná sieťovina s prechodom cez vrcholy a stredy hrán (Amit, 2003)).




**Obr. 12 Navigačná sieťovina s prechodom cez vrcholy a stredy hrán (Amit, 2003)**

Vyhľadenie nájdenej trasy a jej zlepšenie je možné vykonať tak, že po prechode do vrcholu sa zistí, či existuje priama cesta z vrcholu o 2 ťahy späť. Ak áno, tak sa zo stromu prehľadávania, resp. z nájdenej trasy vypustí predchádzajúci vrchol.

Hľadanie trasy pomocou navigačnej sieťoviny je špeciálnou implementáciou prehľadávania stavového priestoru, pričom okrem vrcholov polynómov sa používajú aj pomocné body na ich hranách.

Hlavnou výhodou najmä pri veľkých otvorených plochách je výrazne nižšia pamäťová náročnosť, rýchlosť nájdenia trasy a jej vysoká kvalita.

### *2.1.2 Rastrové mapy*

Pri rastrových mapách je priestor rozdelený na rovnaké pravidelne uložené body, resp. „dlaždice“ (angl. „tiles“), keďže sa reprezentuje plocha. Medzi susednými bodmi sa možno presúvať. Raster, resp. mapa môže mať rôzne definované susedné body. Najčastejšie je počet susediacich bodov 4, 6 alebo 8.

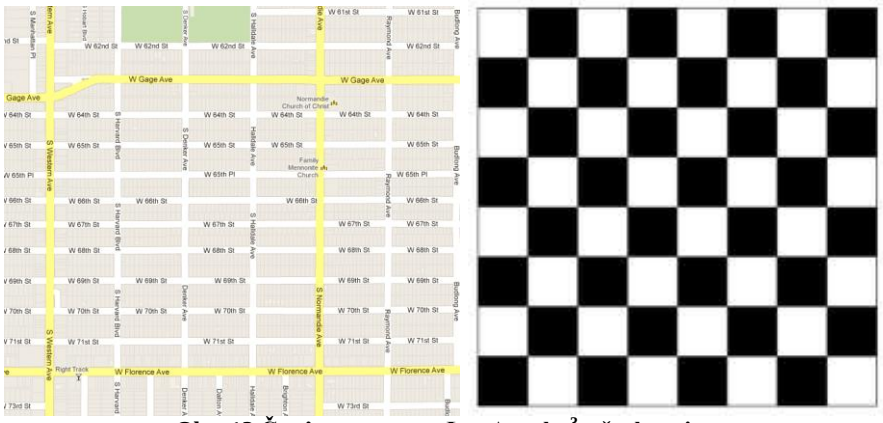
Pokiaľ je počet susedných bodov 4 alebo 8, tak sa mapa zobrazuje pomocou štvorcov, pričom pri 4 susedoch jedného bodu je povolený prechod len cez hrany, zatiaľ čo pri 8 susedných bodoch je možné presúvať sa aj cez vrcholy – po diagonále. Pri 6 susedných bodoch sa mapa zobrazuje pomocou šesťuholníkov. Štvorce a šesťuholníky plne pokrývajú mapu, pokiaľ sa uvažuje o euklidovskom priestore. Štvorcami alebo šesťuholníkmi nie je možné pokryť sférický povrch. Euklidovskú plochu je možné uniformne pokryť aj trojuholníkmi, ale toto pokrytie sa nepoužíva na mapovanie.

V práci sa budeme venovať len klasickému euklidovskému priestoru.

#### *2.1.2.1 Štvorcová mapa*

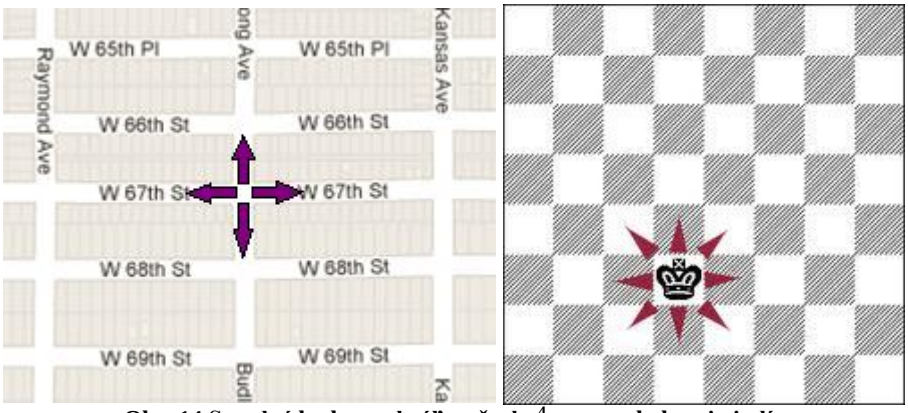
Pri štvorcovej mape je plocha rozdelená na rovnako veľké pravidelne rozmiestnené štvorce, typickým príkladom je šachovnica alebo štandardná pravouhlá mapa ulíc miest, najmä amerických. (Obr. 13 Časť mapy mesta Los Angeles a šachovnica)

Na šachovnici sa môže kráľ a kráľovná pohybovať všetkými smermi, t.j. bod, ktorý nie je na kraji mapy, má prechod do 8 susedných bodov, pričom cena je pre všetky prechody rovnaká.




**Obr. 13 Časť mapy mesta Los Angeles[3] a šachovnica**

V prípade pravidelných ulíc však už diagonálny pohyb auta nie je možný, v tom prípade má bod, pri tejto zjednodušenej mape mesta ide o križovatku, iba 4 susedné body.

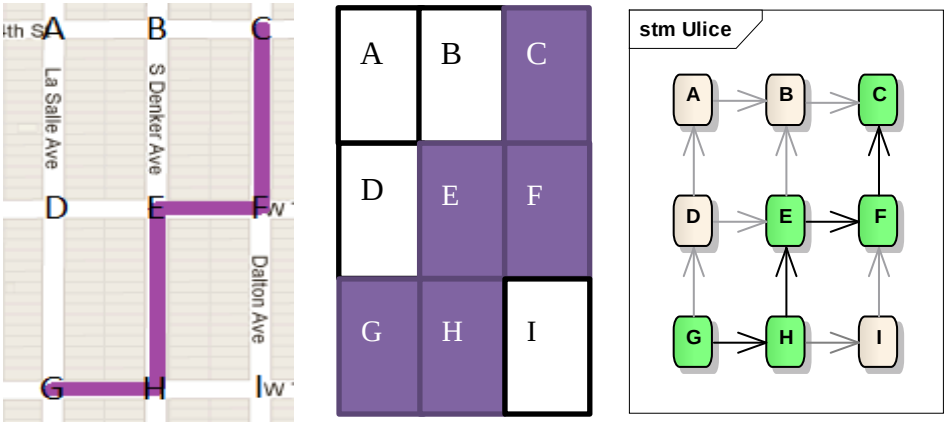



**Obr. 14 Susedné body pre kráľa v šachu[4] a pre pohyb v sieti ulíc**

Pri opise pohybu po mape, resp. po zodpovedajúcom stavovom priestore, sa bude používať pojem sused v zmysle bodu, ktorého reprezentácia na štvorcovej mape má jednu hranu, prípadne jeden vrchol zhodný. V niektorých prípadoch sa môže použiť aj geografické značenie susedov: sever, východ, západ a juh.

Pri navigácii v sieti ulíc sa použije zobrazenie, kde križovatky budú v mape reprezentované dlaždicami a v grafe uzlami. Ulice budú na mape hranami medzi dlaždicami a v grafe hranami medzi uzlami. (Obr. 15 Ekvivalentné reprezentácie mapy a pohybu)

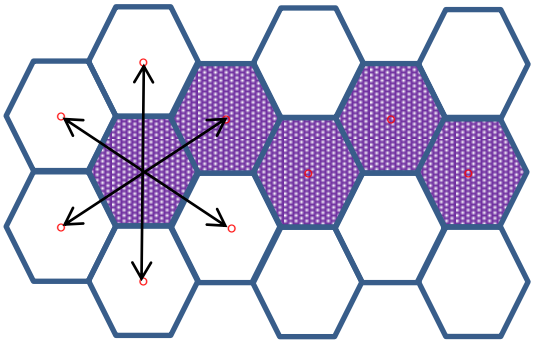
[3] Mapa Los Angeles, Ca. (maps.google.com)
[4] Obrázok prevzatý z http://www.thechesszone.com/chess_rules




**Obr. 15 Ekvivalentné reprezentácie mapy a pohybu z bodu G do bodu C**

### *2.1.2.2 Šesťuholníková mapa*

Šesťuholníková mapa má vzhľad včelieho úľa alebo pletiva či grafitu. Použitie šesťuholníkov na mapovanie má oproti štvorcom výhodu v tom, že jeden bod susedí s inými iba cez hrany a prechod do všetkých šiestich susedov je rovnako nákladný, t.j. stredy všetkých susedných dlaždíc sú rovnako vzdialené. Problém je v tom, že pohyb v určitých smeroch je deformovaný. Ak sú dve hrany napríklad rovnobežné s osou y (sever - juh), tak pohyb na východ alebo západ (po osi x) je neefektívny, má „cik-cak“ tvar (Obr. 16 Šesťuholníková mapa).



**Obr. 16 Šesťuholníková mapa**

Šesťuholníkové mapovanie je populárne v strategických hrách, kde je používané najmä z dôvodu rovnakej vzdialenosti medzi dvomi dlaždicami a toho, že dve dlaždice susedia len cez hrany.

**Obr. 17 Šesťuholníkové mapy v hrách Osadníci z Katanu a Panzer General**[5]

Z topologického hľadiska je použitie šesťuholníkov zhodné s „tehlovou stenou“ (Brimkov, 2004), t.j. obdĺžnikmi alebo štvorcami, kde jeden rad je posunutý o polovicu dĺžky hrany voči radu pod ním, resp. len v jednej osi sú susedné dlaždice posunuté (Obr. 18 Porovnanie pravidelných šesťuholníkov a tehlovej steny (Brimkov, 2004) ).

**Obr. 18 Porovnanie pravidelných šesťuholníkov a tehlovej steny (Brimkov, 2004)**

### *2.1.3 Vzdialenosti v štvorcových mapách*

Najjednoduchšia heuristická funkcia pre hľadanie cesty medzi dvomi bodmi je použitie priamej vzdialenosti stavu, bodu od cieľa. Priama vzdialenosť zabezpečí, že heuristická funkcia je vždy menšia alebo rovná ako skutočná vzdialenosť, resp. náklady prechodu z bodu do cieľa.

Pojem priama vzdialenosť môže označovať rôzne miery podľa toho, ako sa možno po mape, resp. v stavovom priestore pohybovať a aké sú náklady na prechod medzi susednými stavmi.

Pri štvorcových mapách je potrebné zvážiť počet susedov a pokiaľ je možný aj diagonálny presun, tak aj to, či je prechod cez vrchol rovnako nákladný ako cez hranu.

[5] Obrázky prevzaté z www.catan.com a http://panzercentral.com/forum/viewtopic.php?t=45039

Pre vzdialenosť budeme v ďalšom texte používať písmeno d a symbolmi $\bar{a}$ a $\bar{b}$ sa budú označovať body, resp. vektory v priestore, pričom ide o usporiadanú n-ticu čísel, zložiek, kde n je počet rozmerov priestoru. Zložky bodu $\vec{b}$ sa zapisujú ako $\vec{b} = (b_i, b_2, \ldots., b_n)$. Pre 2 rozmerný priestor sa zložky označujú aj ako (x, y).

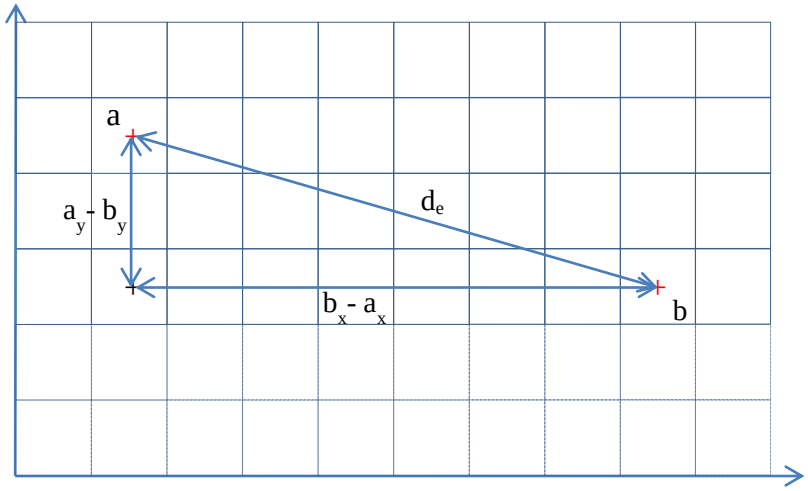
### *2.1.3.1 Euklidovská vzdialenosť*

Euklidovská vzdialenosť je základná geometrická vzdialenosť dvoch bodov. V n-rozmernom priestore sa pre body $\bar{a}$ a $\bar{b}$ počíta vzťahom:

$$d_e(\vec{a}, \vec{b}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

Pre dvojrozmerný priestor, plochu je vzdialenosť bodov $\vec{a} = (a_x, a_y)$ a $\vec{b} = (b_x, b_y)$ nasledovná:

$$d_e(\vec{a}, \vec{b}) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$$




**Obr. 19 Euklidovská vzdialenosť**

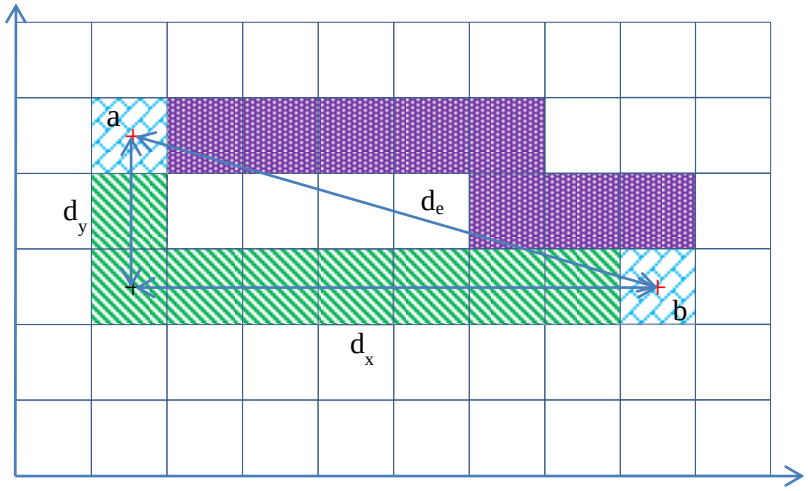
Bodmi sú zvyčajne rovnaké body v rámci meraných dlaždíc; na obrázku Obr. 19 sú takými bodmi stredy dlaždíc, ale vzdialenosť sa nezmení, ani keď sú použité iné referenčné body dlaždice, ako napr. ľavý dolný roh, apod.

### *2.1.3.2 Taxikárska geometria*

Pokiaľ sú povolené len prechody cez hranu, t.j. ide napr. o zjednodušenú mapu siete ulíc, možno použiť geometriu označovanú ako „taxikárska“ (angl. „Taxicab Geometry“). Vzdialenosť v takom priestore sa označuje aj ako manhattenská vzdialenosť.

Vzdialenosť medzi dvomi bodmi, napr. aj medzi dvomi križovatkami v meste, je daná súčtom vzdialeností medzi bodmi v každom smere pohybu. Ak má mesto štvorcové bloky domov jednotkovej dĺžky orientované podľa svetových strán, tak ide o súčet počtu blokov severojužným smerom a počtu blokov východozápadným smerom.

Na obrázku Obr. 20 je znázornený rozdiel medzi taxikárskou a euklidovskou vzdialenosťou a tiež fakt, že v taxikárskej geometrii, na rozdiel od euklidovskej, existuje viacero najkratších ciest.




**Obr. 20 Manhattenská alebo taxikárska vzdialenosť a príklady najkratšej trasy**

Pre n-rozmerný priestor, kde bod $\vec{b}$ má pozíciu definovanú pomocou n-tice reálnych čísel, $\vec{b} = (b_i, b_2, \ldots, b_n)$, platí, že vzdialenosť dvoch bodov $\bar{a}$ a $\bar{b}$, označená ako d, je daná vzťahom:

$$d_t(\vec{a}, \vec{b}) = \|\vec{a} - \vec{b}\| = \sum_{i=1}^{n} |a_i - b_i|$$

Manhattenská vzdialenosť nie je závislá na veľkosti mriežky a aj pokiaľ sa veľkosť blíži limitne k nule, vzdialenosť sa nemení.

Pokiaľ platí, že $a_x$=$b_x$ alebo $a_y$=$b_y$, tak euklidovská a taxikárska vzdialenosť sú zhodné. Najväčší rozdiel medzi euklidovskou a taxikárskou vzdialenosťou je, ak $|a_x-b_x| = |a_y-b_y|$. V tom prípade taxikárska vzdialenosť ($d_t$) a euklidovská vzdialenosť ($d_e$) je:

$$d_t(\vec{a},\vec{b}) = 2|a_x - b_x| = 2|a_y - b_y|$$

$$d_e(\vec{a},\vec{b}) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2} = \sqrt{2(a_y - b_y)^2} = \sqrt{2}|a_y - b_y| = \sqrt{2}|a_x - b_x|$$

Pomer vzdialeností je potom rovný:

$$\frac{d_t}{d_e} = \frac{2|a_x - b_x|}{\sqrt{2}|a_x - b_x|} = \frac{2}{\sqrt{2}} = \sqrt{2}$$

Manhattenská vzdialenosť tak nie je nikdy väčšia ako 1,42-krát euklidovská vzdialenosť.

V taxikárskej geometrii, resp. na štvorcovej mape bez prechodu cez vrchol existuje viacero trás, ktoré majú dĺžku rovnú najkratšej vzdialenosti. Na pravidelnej mape, kde každá križovatka, ktorá nie je na okraji mapy, má 4 susedné križovatky, možno počet optimálnych ciest vypočítať pomocou kombinatoriky.

Pre body $\bar{a}$ a $\bar{b}$ označme hodnoty $|a_x-b_x|$ a $a_y-b_y|$ ako $d_x$ a $d_y$. Na prechod dlaždice a do dlaždice b je teda potrebné prejsť $d_x$ horizontálnych úsekov, t.j. prechod hrany vo východozápadnom smere a $d_y$ vertikálnych úsekov, resp. severojužných prechodov hranou.

Úlohu nájdenia počtu optimálnych trás možno prepísať na úlohu nájdenia počtu možností, na ktorom ťahu sa vykoná vertikálny alebo horizontálny presun. Hľadáme tak v množine $d_x + d_y$ ťahov tie ťahy, kde dochádza buď k horizontálnym ťahom alebo k vertikálnym ťahom. Na množine ťahov, ktoré nie sú v nájdenej množine sa vykonáva druhý pohyb. Množina ťahov je množina, ktorá má rovnaký počet prvkov ako je dĺžka trasy a pre názornosť si ju určíme ako množinu prirodzených čísiel od 1 po $d_x+d_y$. V množine je definované poradie, resp. relácie „<“, pričom „menší“ ťah sa vykoná skôr. Na poradí, v ktorom sa vyberajú ťahy z množiny všetkých ťahov nezáleží, pretože ťahy sa aj tak vykonávajú v ich definovanom poradí, v prípade prirodzených čísel podľa ich hodnoty. Pokiaľ vyberáme množinu ťahov, kde dochádza k vertikálnemu pohybu, je počet možností $p_v$, a pokiaľ vyberáme ťahy horizontálnych pohybov, je počet trás $p_h$. Hodnoty musia byť rovnaké, t.j.

$$p_v = \binom{d_x + d_y}{d_x} = \binom{d_x + d_y}{d_y} = p_h$$

V príklade na obrázku Obr. 20 je $d_x=7$ a $d_y=2$. Počet trás je teda

$$p_v = \binom{d_x + d_y}{d_x} = \binom{9}{2} = 36 = \binom{9}{7} = \binom{d_x + d_y}{d_y} = p_h$$

Množina ťahov je {1, 2, 3, 4, 5, 6, 7, 8, 9}, keďže na prechod z $\bar{a}$ do $\bar{b}$ je potrebných 7 horizontálnych a 2 vertikálne prechody hrán. Pri výbere vertikálnych ťahov, t.j. výber dvoch prvkov z množiny 9, sú trasy zobrazené na obrázku Obr. 20 výberu {1, 2} a {6, 9}. Pohyb po mape, čo je možné chápať aj ako aplikovanie operátorov pri prechádzaní stavového priestoru, je tak „→→→→→↓→→↓“ resp. „↓↓→→→→→→→→“.

Pokiaľ sa vyberajú ťahy vertikálnych prechodov, tak možnosti výberu sú uvedené v tabuľke Tab. 7:

**Tab. 7 Všetky trasy najkratšej dĺžky pre príklad**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1,2 | 1,3 | 1,4 | 1,5 | 1,6 | 1,7 | | 6,7 |
| 2,3 | 2,4 | 2,5 | 2,6 | 2,7 | | 5,6 | 5,7 |
| 3,4 | 3,5 | 3,6 | 3,7 | | 4,5 | 4,6 | 4,7 |

Druhý spôsob ako určiť počet trás optimálnej dĺžky v taxikárskej geometrii je sčítavanie počtu trás zo susedných dlaždíc cieľovej dlaždice, ktoré sú vzdialené od začiatočného vrcholu o jeden prechod menej a takto postupne prísť k začiatočnému bodu. Medzi susednými bodmi je len jedna trasa s dĺžkou rovnou vzdialenosti.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | a | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | 1 | 3 | 6 | 10 | 15 | 21 | 28 | 36 | |
| | 1 | 4 | 10 | 20 | 35 | 46 | 74 | | |
| | 1 | 5 | 15 | 35 | 70 | | | | |

**Obr. 21 Počet počtu trás z bodu a**

Pokiaľ sa mapa, resp. matica vzdialeností (Obr. 21 Počet počtu trás z bodu a) pootočí o 45° doľava, vytvoria hodnoty Pascalov trojuholník.

V taxikárskej geometrii majú kružnice, t.j. množina bodov, ktoré sú rovnako vzdialené od jedného miesta – stredu, tvar štvorcov, ktorých hrany zvierajú 45° uhol voči osiam. (Obr. 22 Vzdialenosti od bodu v taxikárskej geometrii)

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | |
| 5 | 4 | 3 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 3 | 2 | 1 | a | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 4 | 3 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | b | |
| 6 | 5 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | |
| 7 | 6 | 5 | 4 | 5 | 6 | 7 | 8 | 9 | | | |

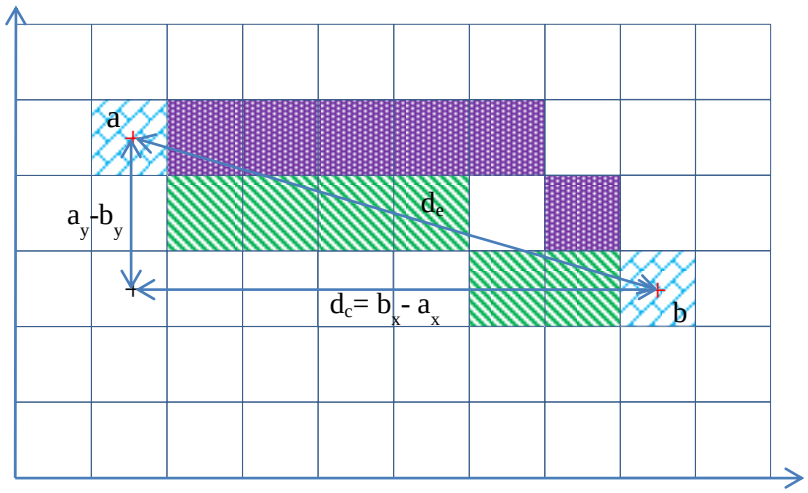
**Obr. 22 Vzdialenosti od bodu v taxikárskej geometrii**

### *2.1.3.3 Čebyševova vzdialenosť*

Vzdialenosti po šachovnici, resp. na štvorcovej mape s povoleným diagonálnym prechodom, kde každý prechod má rovnakú dĺžku, popisuje Čebyševova (Chebyshev) metrika, ktorá je zhodná s metrikou $L_\infty$.

$$d_c(\vec{a},\vec{b}) = \lim_{k\to\infty} \sqrt[k]{\sum_{i=1}^{n} |a_i - b_i|^k} = \lim_{k\to\infty} \left( \sum_{i=1}^{n} |a_i - b_i|^2 \right)^{1/k} = \max_{1\le i\le n} (a_i - b_i)$$

Na obrázku Obr. 23 je uvedený príklad najkratšej trasy a Čebyševovej vzdialenosti medzi bodmi **a** a **b**.




**Obr. 23 Čebyševova vzdialenosť a dve trasy v dĺžke vzdialenosti**

Počet trás s dĺžkou rovnou vzdialenosti sa počíta podobne ako v prípade Manhattenskej vzdialenosti. Namiesto pohybov kolmých na seba sa však používa pohyb v smere jednej osi a pohyb diagonálny.

$$\mathrm{p} = \begin{pmatrix} \min(d_x, d_y) + |d_x - d_y| \\ \min(d_x, d_y) \end{pmatrix} = \begin{pmatrix} \min(d_x, d_y) + |d_x - d_y| \\ |d_x - d_y| \end{pmatrix}$$

V príklade z obrázka Obr. 23 je počet trás medzi bodmi **a** a **b** dĺžky rovnej vzdialenosti daný výrazom:

$$\mathrm{p} = \begin{pmatrix} \min(d_x, d_y) + |d_x - d_y| \\ \min(d_x, d_y) \end{pmatrix} = \binom{2+5}{2} = 21$$

Mapa počtu trás s dĺžkou rovnej vzdialenosti z bodu a je uvedená na obrázku Obr. 24.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | a | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| | 1 | 2 | 1 | 3 | 6 | 10 | 15 | 21 | 28 |
| | 1 | 3 | 3 | 1 | 4 | 10 | 20 | 35 | 56 |
| | 1 | 4 | 6 | 4 | 1 | 5 | 15 | 35 | 70 |

**Obr. 24 Počet počtu najkratších trás z bodu a pri povolenom diagonálnom pohybe**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 5 | 6 | 7 | 8 |
| 3 | 2 | 2 | 2 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 3 | 2 | 1 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 3 | 2 | 1 | a | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 3 | 2 | 1 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 3 | 2 | 2 | 2 | 2 | 2 | 3 | 4 | 5 | 6 | B | 8 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 5 | 6 | 7 | 8 |

**Obr. 25. Vzdialenosti od bodu v Čebyševovej geometrii**

Podobne ako v prípade taxikárskej geometrie, aj pri povolenom prechode cez vrcholy majú kružnice tvar štvorcov, ktoré majú v tomto prípade hrany rovnobežné s osami. (Obr. 25. Vzdialenosti od bodu v Čebyševovej geometrii)

#### *2.1.3.4 Diagonálna vzdialenosť*

Čebyševova vzdialenosť počíta s tým, že náklady na prechod medzi dlaždicami sú rovnaké pri prechode cez hranu a aj pri prechode cez vrchol. Pre dvojrozmerné priestory, plochy v prípade, že je povolený prechod cez vrcholy, ale náklady sú $\sqrt{2}$ krát vyššie ako pri prechode hranou, možno vzdialenosť vyjadriť ako:

$$d_x(\vec{a},\vec{b}) = \sqrt{2}\min\left(|a_x - b_x|, |a_y - b_y|\right) + \left||a_x - b_x| - |a_y - b_y|\right| \\ = \sqrt{2}\min(d_x, d_y) + |d_x - d_y|$$

## 2.2 Podmienka na zdroje

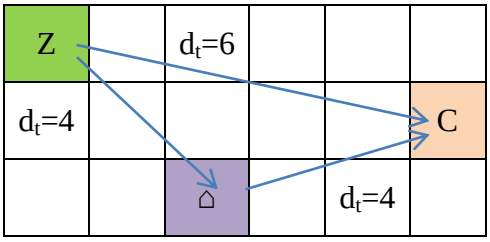
Výrazné zvýšenie náročnosti vyhľadávania trasy predstavuje zavedenie zdroja, pre ktorý platí podmienka, že v nájdenej trase nesmie byť súvislý úsek, kde by bola celková spotreba zdroja vyššia ako definovaný limit bez toho, že by na úseku bola dlaždica so zdrojom.

Spotreba zdroja na jeden prechod môže byť nezávislá od nákladov, ale častý je prípad, keď je spotreba zdroja pri prechode medzi stavmi zhodná s nákladmi na prechod medzi nimi. Spotreba však môže byť aj konštantná pri ľubovoľnom prechode, a teda závislá len od počtu dlaždíc trasy.

Cieľom je modelovať situácie, kde je potrebné pri pohybe po trase dopĺňať zdroj nevyhnutný na pohyb. Pohybujúci sa objekt – agent – nemá dostatočnú kapacitu úložiska zdroja na prejdenie celej trasy a zdroj sa na mape vyskytuje len v definovaných bodoch, v ktorých je ho možné dopĺňať.

Príkladmi môžu byť plavby loďami, ktoré majú zásoby len na určitý počet dní alebo diaľkové lety s medzipristátím na dotankovanie, apod. Podmienka na neprerušenú postupnosť modeluje dotankovanie zdroja do maximálnej kapacity pre každým ďalším úsekom. Modelovať možno aj situáciu, kde prospektor zbiera vzorky na území, ktoré si rozčlenil na dlaždice. Vzorky je možné odovzdať len na určitých miestach na mape a má kapacitu na uschovanie vzoriek len na určitý počet dlaždíc. Zatiaľ čo nákladmi môžu byť

čas alebo palivo, ktoré závisia od vlastností terénu, tak zdrojmi sú vzorky, ktorých kapacita je daná len počtom prejdených dlaždíc bez ohľadu na terén.




**Obr. 26 Pohyb po mape s obmedzenými zdrojmi**

Na obrázku Obr. 26 je príklad trasy na mape, kde je spotreba pri prechode rovná nákladom a tie sú rovné 1 na každý prechod medzi stavmi. Maximálna kapacita je 4, čiže najdlhší úsek trasy bez prechodu dlaždice so zdrojom sa môže skladať najviac zo štyroch dlaždíc. Zdroj je na obrázku označený symbolom ⌂, začiatočný bod je Z a cieľ je C.

Aj keď je najkratšia vzdialenosť 6 a existuje 6 trás danej dĺžky, tak ani jedna z týchto trás nespĺňa podmienku o zdrojoch. Najkratšia trasa spĺňajúca podmienku je dlhá 8 a existuje 6*4, spolu 24 trás danej dĺžky, ktoré prechádzajú bodom so zdrojom.

### *2.2.1 Množstvo zdroja v kroku trasy*

Trasa je postupnosť stavov $s_1$, $s_2$...., $s_n$. Stav s na i-tom mieste označujeme aj ako i-ty krok trasy. Pre každý krok v trase definujeme množstvo zdroja, t.j. aké množstvo zdroja je k dispozícii na prechod z aktuálnej dlaždice/stavu na stav v nasledujúcom kroku. Množstvo zdroja v kroku n označíme $z_n$ a počítame ho ako súčet množstva zdrojov potrebných na prechod od najbližšej predchádzajúcej dlaždice so zdrojom do kroku n odpočítanej od maximálneho množstva zdroja dostupného na pohyb po mape.

Pre krok n, kde $s_n$ nie je dlaždica so zdrojom platí, že

$$z_n = z_{n-1} + S(s_{n-1}, s_n)$$

Kde $S(s_{n-1},s_n)$ je spotreba zdroja pri prechode zo stavu $s_{n-1}$ do stavu $s_n$, t.j. spotreba zdroja pri prechode z dlaždice na n-1 mieste trasy do stavu na n-tom mieste trasy T. Spotreba je vždy kladná hodnota. Pokiaľ stav v kroku n je stav reprezentujúci dlaždicu mapy so zdrojom, tak $z_n = z^{max}$.

Ak označíme $s_1$ stav so zdrojom, teda $z_1=z^{max}$, tak pre stav $s_n \in T$, kde v trase medzi krokmi 1 a n (vrátane kroku n) neprechádza stav so zdrojom, je množstvo zdroja v kroku n dané vzťahom:

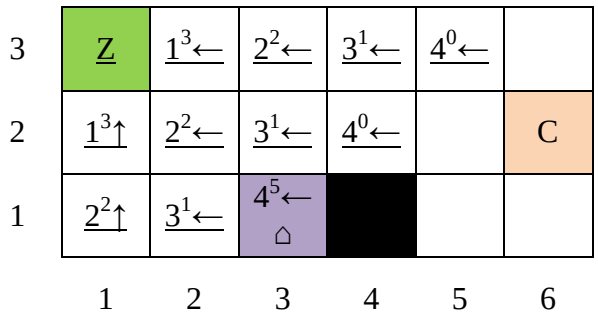
$$z_n = \mathrm{z}^{max} - \sum_{i=1}^{n-1} S(s_i, s_{i+1}) = \mathrm{z}^{max} - S(s_1, s_n).$$

Podmienku na zdroje v trase možno potom zapísať nasledovne: Označme množinu krokov, v ktorých sa prechádza dlaždicou so zdrojom, $S_z$. Stav v kroku i označíme ako $s_i$. Potom musí platiť:

$$\forall x, y, x < y, \forall i, x < i < y, i \notin S_Z: \sum_{j=\mathrm{x}}^{y-1} S(s_j, s_{j+1}) = S(s_x, s_y) \leq z^{max}$$

### *2.2.2 Dopad zdrojov na algoritmy prehľadávania*

Podmienka na zdroje spôsobuje, že klasické slepé a heuristické algoritmy nemusia nájsť trasu a nedosiahnuť cieľ aj v prípadoch, keď trasa existuje. Trasy s menšími nákladmi, ktorým sa minú zdroje, môžu zablokovať trasy, ktoré síce majú väčšie náklady, ale cieľ sa dosiahne bez porušenia podmienky na zdroje.




**Obr. 27 Zablokovanie prehľadávania pri algoritme Rovnomerné náklady**

Na obrázku Obr. 27 je príklad zablokovania prehľadávania. Použitá je notácia „$\underline{\mathrm{N}^{\mathrm{Z}}\mathrm{O}}$“, kde N sú celkové náklady, $^{\mathrm{Z}}$ je množstvo zdrojov, s ktorými sa spracovával stav, a O je operátor pohybu, ktorým je šípka na predka v strome prehľadávania, teda na predchádzajúci bod v trase, ktorá tak má opačný smer ako bol uskutočnení pohyb. Operátormi sú znaky „↑“, „←“, „↓“ a „→“. Podčiarknuté stavy sú spracované a nepodčiarknuté sú v zozname otvorených stavov, pričom sú zoradené podľa celkových nákladov stavu (hodnoty N v $\mathrm{N}^{\mathrm{Z}}\mathrm{O}$).

Dlaždica (3,2) bola spracovaná na trase zo začiatku, keď náklady na jej dosiahnutie boli 3, pričom išlo o 3 dlaždice za sebou v trase bez prechodu zdrojom.

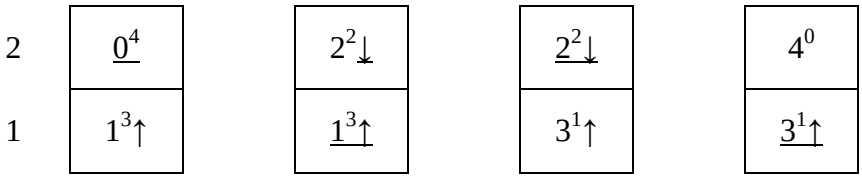
Keď sa začne spracovávať dlaždica (3,1) so zdrojom, do ktorej najnižšie náklady sú 4, a teda sa spracováva neskôr, ako sa spracoval bod (3,2), tak už je (3,2) štandardným

algoritmom zablokované. Trasa, ktorá má väčšie náklady, má však väčšiu zásobu zdrojov a je schopná dosiahnuť cieľ.

K zablokovaniu by došlo aj v situácii, keby sa použil heuristický algoritmus, ktorý by neumožnil aj prehľadávanie spracovaných stavov.

### *2.2.3 Dominancia trás*

Zavedenie zdrojov do hľadania trasy spôsobilo, že je potrebné skúmať aj už raz preskúmané stavy. Nie je však možné skúmať všetky trasy z každého skúmaného bodu, viedlo by to k neefektívnemu prehľadávaniu rovnakých bodov.




**Obr. 28 Neefektívne skúmanie preskúmaných stavov**

Na obrázku Obr. 28 je znázornený príklad neefektívneho prehľadávania medzi dvomi stavmi. Zo stavu 2 sa vytvoril potomok 1, ktorého potomkom je opäť stav 1, atď. Tento cyklus zastaví až minutie zdrojov pohybu, je však zrejmé, že neobmedzené prehľadávania už preskúmaných stavov by viedli k skúmaniu každého stavu niekoľkokrát[6]. Opätovné skúmanie by spustil každý sused, čím by sa ešte viac znásobil počet prehľadávaní. Význam pre nájdenie optimálnej trasy by však mal len nepatrný zlomok.

Nové preskúmanie prechodu dlaždicou má význam len vtedy, keď sa tým zlepší aspoň jedna vlastnosť trasy. Zlepšenie môže byť buď v celkových nákladoch alebo množstve zdrojov. Zlepšenie celkových nákladov z princípu fungovania algoritmov ako Rovnomerné náklady alebo A* nemôže nastať. Opätovné preskúmanie stavu je preto dané zlepšením v množstve zdrojov.

Opätovne skúmať prechod stavom s, ktorý už bol aspoň raz preskúmaný v trase $T_1$ v kroku i, má význam pre trasu $T_2$, kde sa stavom prechádza v kroku j len, ak platí, že

$$s_i^1 \in T_1, s_j^2 \in T_2, s_i^1 = s_j^2, z_i^1 < z_j^2$$

Pokiaľ je splnená táto podmienka a zároveň celkové náklady na prechod do s zo začiatočného stavu po trase $T_1$ sú vyššie ako celkové náklady na prechod do s zo

[6] Počet je závislý od spotreby zdrojov pri prechode zo stavu 1 do stavu 2 a pri prechode zo stavu 2 do stavu 1 a aj od toho, koľko existuje susedov.

začiatočného stavu s po trase $T_2$, tak hovoríme, že trasa $T_2$ dominuje trase $T_1$. Označme celkové náklady v kroku i ako $n_i$. Pokiaľ potrebujeme rozlíšiť trasu, tak pre trasu $T_j$ označíme celkové náklady na prechod do kroku i ako $n_i^j$. Podmienka dominancie je potom nasledovná:

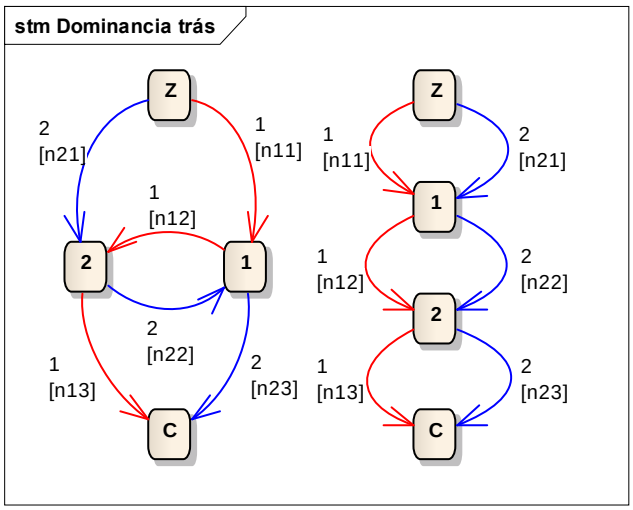
$$T_1 \sqsubseteq T_2 \Leftrightarrow \exists s_i^1 \in T_1, s_j^2 \in T_2, s_i^1 = s_j^2, z_i^1 \geq z_j^2 \wedge n_i^1 \leq n_j^2$$

Silnú dominanciu definujeme nasledovne:

$$T_1 \sqsubset T_2 \Leftrightarrow \exists s_i^1 \in T_1, s_j^2 \in T_2, s_i^1 = s_j^2, z_i^1 \geq z_j^2 \wedge n_i^1 < n_j^2$$

Vzťah dvoch trás môže byť aj taký, že $T_1$ dominuje $T_2$ a $T_2$ dominuje $T_1$. Trasy musia mať aspoň dva spoločné body a dôsledok je potom ten, že ani jedna trasa nie je optimálna.

Nech $T_1$ a $T_2$ sú trasy medzi začiatočným a koncovým bodom. Označme bod, kde $T_1$ dominuje $T_2$ $s_1$ a bod, kde $T_2$ dominuje $T_1$ $s_2$. Umiestnenie bodov na trasách môže byť dvojaké: buď je skorší zo skúmaných dvoch spoločných bodov v trase $T_1$ skorším aj v trase $T_2$, alebo je daný bod v trase $T_2$ neskorší (Obr. 29 Dominancia trás a optimálnosť trasy). Na obrázku sú znázornené aj náklady jednotlivých úsekov, všetky hodnoty „nXX“ sú kladné čísla.




**Obr. 29 Dominancia trás a optimálnosť trasy**

Z dominantnosti v sériovej konfigurácii (na obrázku vpravo), t.j. skorší bod v $T_1$ je skorším aj v $T_2$, vyplýva:

$$n_{11} < n_{21}$$

$$n_{11} + n_{12} > n_{21} + n_{22}$$

z toho vyplýva: $n_{12} > n_{22}$

Ak vytvoríme trasu $T_3$, tak úsek do bodu $s_1$ má zhodný s $T_1$ a úsek z $s_1$ do $s_2$ zhodný s $T_2$ a úsek zo stavu $s_2$ do cieľa má zhodný s tou trasou, ktorá má na tento úsek menšie náklady, tak je zrejmé, že celkové náklady $T_3$ sú nižšie ako $T_1$ a $T_2$:

$$\sout{n_{11}} + n_{22} + \min(n_{13}, n_{23}) < \sout{n_{11}} + n_{12} + n_{13}$$

a súčasne $n_{11} + \sout{n_{22}} + \min(n_{13}, n_{23}) < n_{21} + \sout{n_{22}} + n_{23}$

Aby pri paralelnej konfigurácii, t.j. skorší spoločný skúmaný bod v $T_1$ je neskorším v trase $T_2$ (na obrázku vľavo), platila podmienka dominancie, tak musí platiť:

$$n_{11} < n_{21} + n_{22},$$

$$n_{11} + n_{12} > n_{22}.$$

Pri paralelnom zapojení je možná iba situácia, kde trasa je dominantná len v skoršom zo spoločných skúmaných bodov. Ak by bola trasa $T_1$ dominantná v $s_2$, tak by musela mať menšie náklady aj v $s_1$, čiže $T_2$ by nemohla byť v $s_1$ dominantná. Vzťahy dominancie nie je možné splniť, pokiaľ sú všetky náklady kladné čísla:

$$n_{11} + n_{12} < n_{21}$$

$$n_{11} > n_{21} + n_{22}$$

Z nich je možné odvodiť: $\sout{n_{21}} + n_{22} < n_{11} < \sout{n_{21}} - n_{12}$. Aby ten vzťah platil, muselo by platiť aj $n_{22} < -n_{12}$, čo však nie je možné, pretože $n_{22}$ a aj $n_{21}$ sú väčšie ako 0.

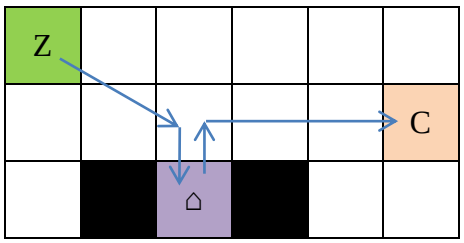
Pri paralelnej konfigurácii je tak vzájomná dominancia možná, len ak je trasa dominantná v skoršom spoločnom bode. To, že ani jedna trasa potom nie je optimálna, dokážeme nasledovne:

Nech $n_{13} < n_{23}$. Potom definujeme trasu $T_3$ tak, že prvý úsek je od začiatku do bodu 2 zhodný s trasou $T_2$ a druhý úsek od bodu 2 do cieľa je zhodný s trasou $T_2$. Platí, že celkové náklady na trasu $T_3$ sú $n_{21} + n_{13}$. V porovnaní s trasou $T_1$ sú menšie náklady, pretože $n_{21} < n_{11} + n_{21}$. Oproti celkovým nákladom trasy $T_2$ sú náklady $T_3$ menšie, pretože $n_{22} > 0$ a $n_{23} > n_{31}$. Z toho vyplýva, že ani trasa $T_1$ a ani trasa $T_2$ nie sú optimálne.

Znamená to, že pokiaľ sa počas hľadania zistí, že skúmané pole (presnejšie potomok spracovávaného stavu) je už preskúmané, ale náklady na dosiahnutie boli vyššie a množstvo zdroja bolo nižšie, tak je možné pôvodnú trasu zrušiť a nahradiť novou.

### 2.2.4 *Duplicitný prechod bodom mapy*

Doteraz opísané algoritmy hľadania trasy (prehľadávania stavového priestoru) predpokladali, že optimálna trasa prechádza každý bodom mapy najviac raz, resp. na trase nie sú slučky[7]. Podmienka na zdroje však spôsobuje, že existujú mapy, v ktorých je potrebné cez jeden bod prejsť viac ako raz. Príkladom je mapa na obrázku Obr. 30.




**Obr. 30 Mapa s nevyhnutným dvojitým prechodom bodu**

Je preto potrebné upraviť algoritmy prehľadávania tak, aby umožnili nielen opätovné preskúmanie už raz spracovaných stavov, ale aj generovanie trasy, kde je viacnásobný prechod určitým stavom.

Aby malo zmysel skúmať aj druhý prechod stavom, tak musí platiť, že v ňom pri druhom prechode musí byť k dispozícii väčšie množstvo zdrojov ako pri prvom. Slučka musí zväčšiť množstvo zdroja. Ak by to neplatilo, tak by určite trasa so slučkou nebola optimálna.

Pokiaľ by sa totiž vynechal úsek medzi prvým a druhým prechodom duplicitného bodu, tak by trasa mala celkové náklady nižšie, keďže náklady na prechod medzi dlaždicami sú vždy kladné. Ak by trasa bez duplicitného prechodu dosiahla cieľ aj nižšími zdrojmi v duplicitnom stave, tak by ho nevyhnutne dosiahla s nižšími nákladmi ako trasa s duplicitným prechodom. Zvýšenie množstva zdroja slučkou však negarantuje dosiahnutie cieľa, iba zväčší množinu dosiahnuteľných bodov.

Formálne možno podmienku, kedy má duplicitný prechod jedným bodom, t.j. zavedenie slučky v trase zmysel skúmať, možno definovať nasledovne[8]:

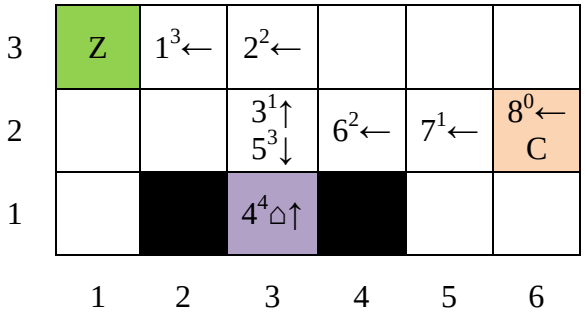
$$\mathrm{i} < \mathrm{j}, s_i \in T, s_j \in T, s_i = s_j \;\Rightarrow z_i < z_j.$$

Ak podmienka neplatí, tak duplicitný prechod stavom nemá význam a slučka, úsek medzi dvomi prechodmi rovnakým stavom, určite nie je súčasťou optimálnej trasy.

[7] Pokiaľ by na trase bola slučka, tak by pri kladných nákladoch na každý prechod medzi dlaždicami automaticky existovala rovnaká trasa len bez slučky, a tá by tak mala nižšie náklady. Z toho vyplýva, že trasa so slučkou nikdy nemôže byť optimálnou trasou.

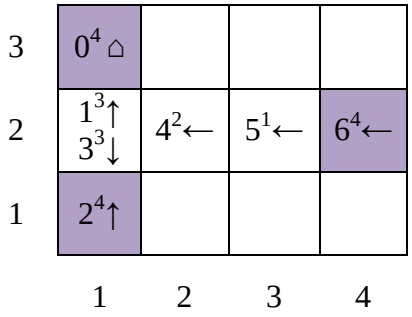
[8] Rovnosť $s_i = s_j$ znamená, že sa jedná o rovnaké dlaždice na mape, ktoré sa v trase vyskytujú na i-tom a j-tom mieste.

V mape na obrázku Obr. 31 je znázornená trasa, ktorá stavom – dlaždicou – (3,2) prechádza dvakrát, raz v treťom kroku a druhýkrát v piatom. Pre množstvo zdroja platí $z_3$=1 a $z_4$=3, podmienka je teda splnená a slučka môže byť súčasťou optimálnej trasy.




**Obr. 31 Príklad trasy s duplicitným prechodom jedného stavu**

Trasa na obrázku Obr. 32 už podmienku zmysluplnosti obchádzky nespĺňa, keďže $z_1 = z_3$, a obchádzka je neefektívna.




**Obr. 32 Príklad trasy so zbytočným duplicitným prechodom jedného stavu**

Podmienka na duplicitný prechod je slabšia ako podmienka dominancie. Druhý prechod stavom má automaticky väčšie celkové náklady, keďže náklady na prechod sú vždy kladné čísla. Pokiaľ teda nemá druhý prechod viac zdroja, tak je dominovaný prvým prechodom a vylúči sa zo zvažovanej trasy.

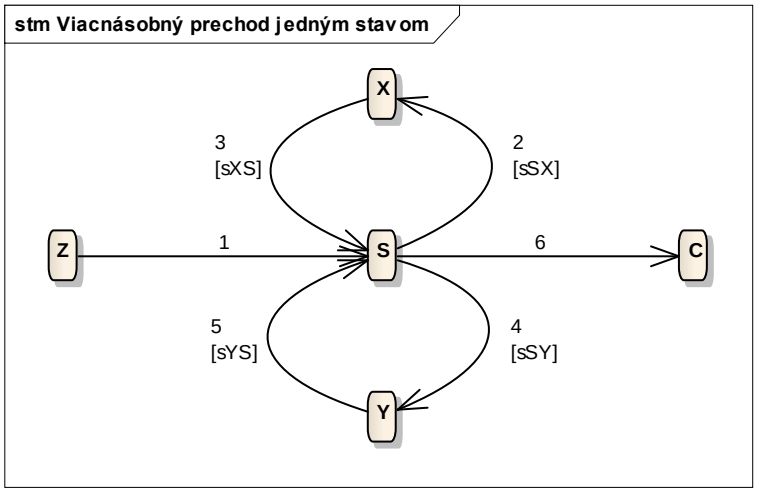
Dôsledok možnosti duplicitných alebo viacnásobných prechodov jedným bodom mapy je taký, že jeden bod mapy môže byť zvažovaný viackrát, a to ako v zozname otvorených, tak i uzavretých.

### *2.2.5 Vlastnosti funkcie spotreba zdroja*

Podmienka na zdroje spôsobuje, že aj v optimálnej trase je možný viacnásobný prechod jedným bodom za podmienky, že v každom ďalšom prechode je k dispozícii viac zdrojov. Podľa vlastností funkcie spotreby môže byť v optimálnej trase maximálne dvojnásobný alebo viac ako dvojnásobný prechod.

Nech trasa T obsahuje tri prechody jedným stavom s v krokoch i, j a k: $s_{i,j,k} \in T, i < j < k,\ s_i = s_j = s_k$ (Obr. 33 Trojnásobný prechod stavom). Nech platia aj podmienky zmysluplnosti duplicitného prechodu pre úseky trasy v krokoch i až j a j až k, t.j. $z_i < z_j$ a $z_j < z_k$.

Na obrázku je použitá notácia pre spotrebu sAB = S(a,b), napr. sSX je spotreba na úseku medzi stavom s a stavom $s_x$. V ďalšom texte budeme zjednodušene písať $S_{sx}$=S(s,$s_x$).




**Obr. 33 Trojnásobný prechod stavom**

Z platnosti podmienok $z_i < z_j$ a $z_j < z_k$ vyplýva, že zvažovaný stav neobsahuje zdroj a tiež, že na úseku v krokoch i až j a j až k trasa prechádza aspoň jedným zdrojom, keďže spotreba je vždy kladné číslo. Označme tieto stavy $s_x$ a $s_y$, pričom i < x < j < y < k, $s_x \neq s_y \neq s$. Pre takúto trasu platí:

$$z_i < z_j < z_k$$

$$z_i \geq S_{sx} \text{ a } z_j \geq S_{sy}$$

$$z_j = z^m - S_{xs} \text{ a } z_k = z^m - S_{ys}$$

Z týchto základných vzťahov je možné odvodiť nasledovné:

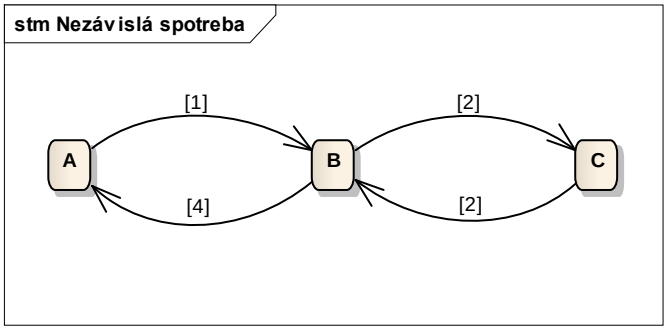
$$z_j = z^m - S_{xs} < z^m - S_{ys} = z_k$$

$$S_{xs} > S_{ys}$$

Slučka cez stav $s_x$ má význam iba vtedy, ak na prechod do stavu $s_y$ v kroku i nie je dosť zdrojov, teda ak $S_{sx} \leq z_i < S_{sy}$. Ak nerovnosť neplatí a vzťahy sú $S_{sy} \leq S_{sx} \leq z_i$, tak prechod slučkou cez stav $s_x$ je zbytočný a len zvýši celkové náklady.

#### *2.2.5.1 Nezávislá spotrebná funkcia pre návrat*

V prípade, že spotreba zdroja na prechod zo stavu $s_b$ do stavu $s_a$ nie je závislá na spotrebe pri prechode $s_a$ do $s_b$, tak je možné v optimálnej trase mať aj viac ako dva prechody jedným stavom. Na obrázku Obr. 34 je príklad takej funkcie spotreby.




**Obr. 34 Príklad nezávislej spotrebnej funkcie**

#### *2.2.5.2 Podmienky pre obmedzenie viacnásobného prechádzania*

Dostatočná podmienka na to, aby v každej optimálnej trase bol najviac dvojnásobný prechod jedným stavom, je, aby spotreba prechodu zo stavu a do stavu b bola rovnaká ako zo stavu b do a.

Táto podmienka je však zbytočne silná a existujú aj iné funkcie medzi S(a,b) a S(b,a), resp. definície spotrebnej funkcie, ktoré zabezpečia, že z jedného bodu bude môcť byť najviac jedna slučka. V dokumente však budeme skúmať model so všeobecnou funkciou spotreby pri prechode medzi stavmi.

## 2.3 Nadmorská výška a terén

Kapitola je venovaná ďalším vlastnostiam mapy ovplyvňujúcim výber optimálnej trasy z hľadiska celkových nákladov:

- nadmorskú výšku, resp. prevýšenie medzi susednými dlaždicami;
- terén.

### *2.3.1 Nadmorská výška*

Každá dlaždica, stav má priradenú výšku, ktorá ovplyvňuje náklady na pohyb. Vplyv môže byť dvojaký. Náklady na prechod medzi stavmi sú modifikované na základe absolútnej výšky a/alebo výškového rozdielu.

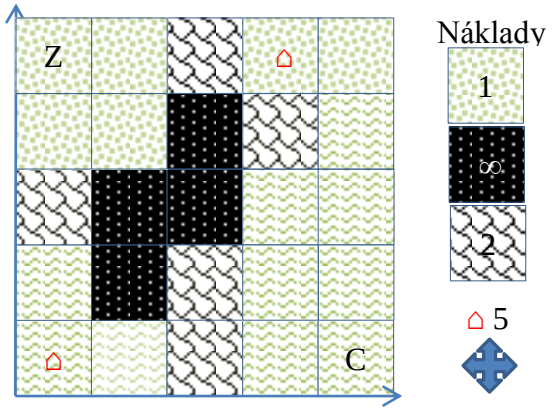
Prevýšenie môže zaviesť do prechodov medzi stavmi nekomutatívnosť (náklady) a prípadne aj spotreba pri prechode medzi stavmi sa môže líšiť podľa smeru pohybu. Táto vlastnosť má výrazný vplyv na vlastnosti trasy, vedie k možnosti viacnásobných slučiek v optimálnej trase.

### 2.3.2 *Terén*

Mapa obsahuje terény rôznej náročnosti pohybu. Základná rýchlosť, resp. minimálne náklady na prechod dlaždice sú rovné jej dĺžke, ktorá je pre jednoduchosť rovná 1. Ďalšie terény majú vyššie náklady, pričom existuje aj nepriechodný terén, t.j. terén s nekonečnými nákladmi.

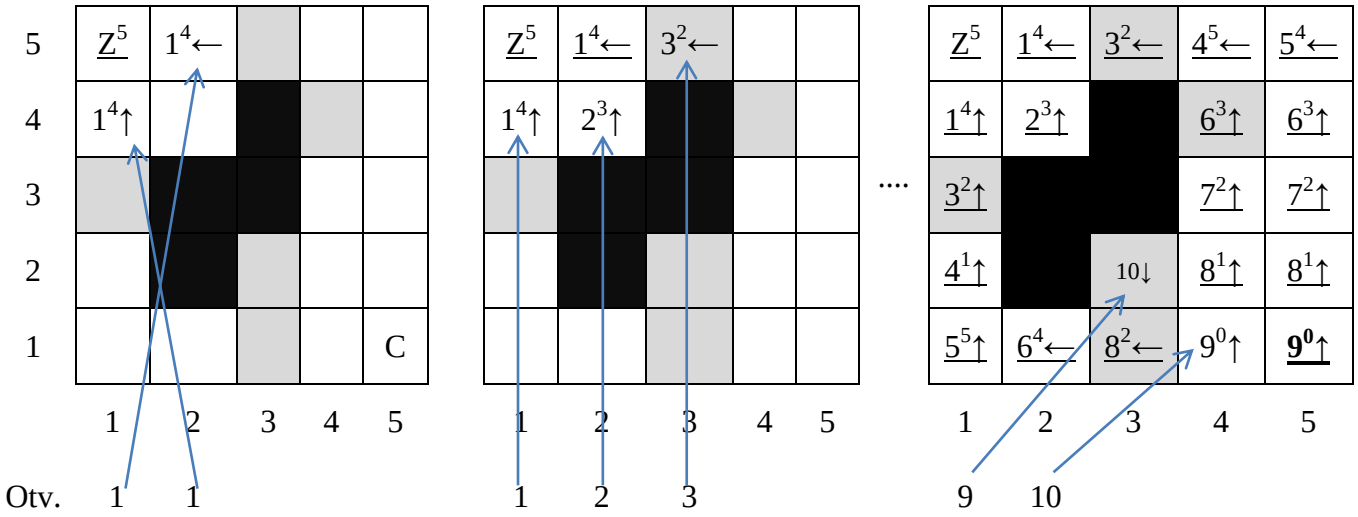
#### *2.3.2.1 Príklad hľadania trasy v mape s rôznymi terénmi*

V mape Obr. 35 je začiatočný stav označený Z a cieľ písmenom C. Použité sú tri typy dlaždíc, čierna je nepriechodná, základný typ má náklady na prechod 1 a náročnejší typ má náklady 2. Všetky dlaždice sú v rovnakej výške. Prechod je povolený len do 4 smerov, takže ide o aplikáciu taxikárskej geometrie. Zdroje sú označené symbolom „⌂“. Začína sa so stavom 5, čo je aj maximálny stav, a každý krok stojí toľko zdrojov, koľko sú náklady na prechod. Na vyhľadanie trasy sa použije algoritmus Rovnomerné náklady.




**Obr. 35 Jednoduchá mapa**

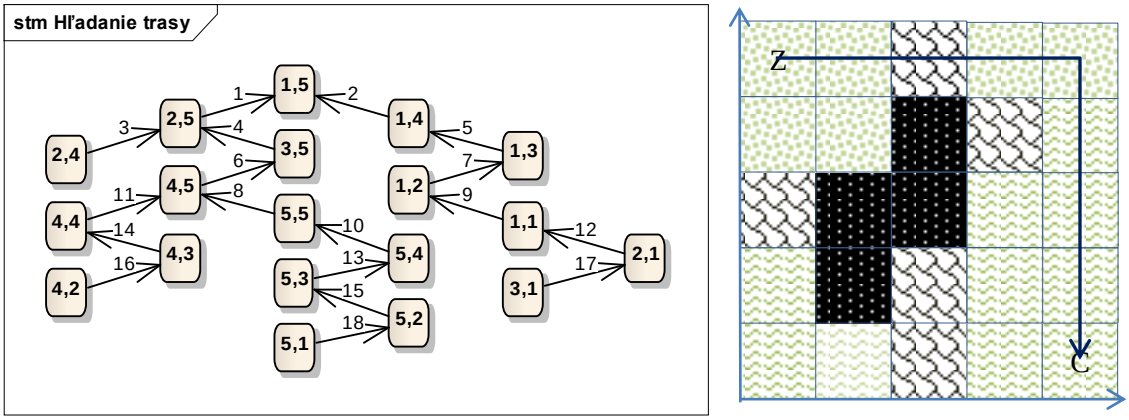
Mapa prehľadávania v rôznych momentoch hľadania trasy je na obrázku Obr. 36. Číslo vyjadruje celkové náklady a šípka smer do predchádzajúceho stavu. Podčiarknuté hodnoty reprezentujú spracované stavy, nepodčiarknuté sú otvorené stavy.




**Obr. 36 Mapa prehľadávania počas hľadania trasy**

Krajná ľavá mapa reprezentuje stav po spracovaní začiatočného stavu, stredná mapa zobrazuje stav po spracovaní bodu (2,5) a krajná po spracovaní bodu (5,1), t.j. cieľa.

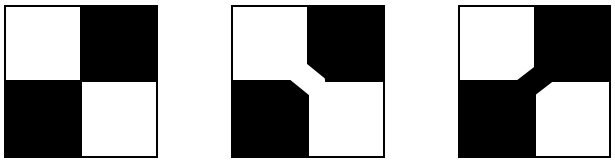
Po skončení prehľadávania sa dosiahol cieľ s celkovými nákladmi 9 a v zozname otvorených stavov sú dlaždice (3,2) a (4,1). Na príklade je vidieť ako terén ovplyvňuje zaraďovanie stavov do zoznamu otvorených.




**Obr. 37 Strom prehľadávania mapy a nájdená cesta**

## *2.3.3 Dopad doplnkových vlastností na 8-smerovú mapu*

Pre mapy s povolenými 8 smermi pohybu je potrebné riešiť prechod vrcholom (Obr. 38 Priechod vrcholom dlaždice). Mapa tvorená len dlaždicami neurčuje, ako sa správa vrchol. Je prechod v tomto prípade možný alebo nie je? Ak je povolený, aký vplyv majú dlaždice, ktoré majú spoločný vrchol so začiatočnou a cieľovou dlaždicou prechodu?



**Obr. 38 Priechod vrcholom dlaždice**

Možností riešenia je viacero:

- Určia sa pravidlá vyhodnocovania (napr. prechod je vždy povolený, vždy zakázaný, východným smerom je povolený; definuje sa pravidlo o vlastníckom bode – t.j. dlaždice, ktorá určuje vlastnosti vrcholu, apod.);
- V mape sa presne určia vlastnosti vrcholu, pričom tie nemusia byť zhodné s vlastnosťami ani jednej z dlaždíc, ktoré ho obsahujú.

Druhé uvedené riešenie výrazne zvyšuje zložitosť reprezentácie mapy, ale umožňuje realistickejšie modelovať terén, keďže sa odstránia neprirodzené pravidelnosti v mape.

## 2.4 Príklady možných scenárov hľadania trasy

Na priblíženie preberanej problematiky uvedieme niektoré možné použitia preberaných vlastností na modelovanie reálnych situácií.

### *2.4.1 Automobil*

V tomto scenári je zdrojom palivo alebo elektrická energia. Pre jednoduchosť sa nepočíta s množstvom paliva, stačím ak trasa neobsahuje viac ako N ťahov bez navštívenia dlaždice so zdrojom, a teda spotreba pri prechode medzi stavmi je konštantná hodnota 1.

Rôzne terény predstavujú rôzne kvalitné cesty, na diaľnici je najjednoduchší pohyb. V automobile sa zvyšujú náklady len na cestu hore, čiže len pri pohybe na bod s vyššou nadmorskou výškou. Prechod na nižšie položený bod neovplyvní náklady.

Scenár môže pracovať s taxikárskou geometriou, kde nie sú povolené prechody cez vrcholy a kde nekrajný bod mapy má štyroch susedov.

### *2.4.2 Cestovateľ*

Zdrojom v tomto scenári je voda alebo jedlo. Cestovateľ putuje po krajine, kde sú vodné zdroje vzácne, napríklad austrálske vnútrozemie alebo púšte s oázami.

Terén má rôznu úroveň schodnosti. Pre výškové rozdiely platí, že nielen stúpanie, ale aj klesanie zvyšuje náklady. Pohyb bude možný aj cez vrcholy dlaždíc, nekrajný bod mapy má osem susedov. Diagonálny prechod bude 1,42-krát nákladnejší ako prechod cez hranu.

## 2.5 Rozšírenie algoritmov vyhľadávania

Pre nájdenie trasy v rastrovej mape s podmienkou na zdroje je potrebné počítať aj so situáciami, keď trasa viackrát prechádza jedným bodom. Pri prehľadávaní sa nedá jednoducho označiť dlaždica za preskúmanú a uzavretú. Je však možné využiť pravidlo o dominancii na vyraďovanie neperspektívnych trás.

Na vyhľadanie trasy je možné použiť upravené štandardné algoritmy. Namiesto jednoduchej polohy, resp. dlaždice však za stav bude považovaná kombinácia polohy a hodnoty zdroja pre pohyb.

```
DECLARE
  zOtvorené  STROM
  zUzavreté  STROM
  sSpracovavany PRVOK
ZAČIATOK
  zOtvorené.PRIDAJ(začiatočnýStav)
  KÝM zOtvorene.NieJePrazdny TAK
     sSpracovavany = zOtvorene.VyberPrvy
     AK (sSpracovany je sCielovy) TAK VRÁT(sSpracovany)

     PRE KAŽDÝ STAV sStav V (sSpracovanany.Stav.GenerujPotomkov)
       /* generuj potomkov okrem nákladov počíta aj spotrebu */
       AK zUzavrete.EXISTUJE_DOMINUJUCI_STAV_PRE(sStav) ALEBO
         zOtvorene.EXISTUJE_DOMINUJUCI_STAV_PRE(sStav) TAK
           DALŠÍ /* začne sa spracovávať ďalší potomok */

       zUzavreté.ODSTRÁN_DOMINOVANÉ_STAVOM(sStav)
       zOtvorene.ODSTRÁN_DOMINOVANÉ_STAVOM(sStav)

       /* špecifické výpočty potrebné na zaradenie potomka do zoznamu otvorených
          Zaradenie na určité miesto z zozname otvorených */
       zOtvorene.PRIDAJ(sStav) /* konkrétna metóda podľa algortimu */

     zUzavreté.PRIDAJ(sSpracovanany)
  KONIEC CYKLU
  VRÁT(NIČ)
KONIEC
```

**Alg. 11 Úprava algoritmov, ktoré udržujú zoznam spracovaných stavov**

Pri spracovaní stavu sú potomkami také susedné dlaždice, do ktorých je možné prejsť so zdrojmi dostupnými v spracovávanom stave. Pre každého potomka sa otestuje, či neexistuje v zozname preskúmaných a otvorených stavov stav, ktorý by potomka dominoval a tiež sa vylúčia z otvorených a spracovaných stavov tie stavy, ktoré sú dominované potomkom.

Podobný princíp definovali Natasha Alechina a Brian Logan pre hľadanie trasy s obmedzeniami (Alechina, 1998) v algoritme ABC.

Takto je možné upraviť algoritmy ako Prehľadávanie s rovnomernými nákladmi, A* a ďalšie algoritmy, ktoré majú zoznam otvorených a uzavretých stavov.

## 2.6 Analyzované algoritmy a heuristiky

Cieľom práce je porovnať algoritmy a heuristiky pri hľadaní trasy v rastrovej mape s podmienkou na dostupnosť zdrojov.

### *2.6.1 Algoritmy*

Pre hľadanie trasy použijeme slepé aj heuristické algoritmy, pričom algoritmy je potrebné upraviť tak, aby vedeli nájsť trasu aj pri podmienke na dostatok zdrojov pohybu. Budeme skúmať tieto základné algoritmy:

- Prehľadávanie s rovnomernými nákladmi;
- Chamtivé hľadanie;
- A*;
- Prehľadávanie najskôr do šírky.

### *2.6.2 Heuristiky*

Pre heuristické algoritmy budeme skúmať viaceré heuristiky. Všetky budú založené na vzdialenosti skúmanej dlaždice od cieľa. Použije sa euklidovská a manhattenská vzdialenosť. Do heuristík sa v niektorých prípadoch započíta aj nadmorská výška a/alebo množstvo dostupných zdrojov.

V tabuľke Tab. 8 sú uvedené testované heuristické funkcie. Písmeno a je aktuálna skúmaná dlaždica a c je cieľ.

**Tab. 8 Skúmané heuristiky**

| | Heuristika | Výpočet |
|---|---|---|
| 1 | Euklidovská vzdialenosť | $\sqrt{(a_x - c_x)^2 + \left(a_y - c_y\right)^2}$ |
| 2 | Taxikárska vzdialenosť | $\lvert a_x - c_x \rvert + \lvert a_y - c_y \rvert$ |
| 3 | Taxikárska vzdialenosť a výškový rozdiel | $\lvert a_x - c_x \rvert + \lvert a_y - c_y \rvert + \lvert v_a - v_c \rvert / (v_{max} + 1)$ |
| 4 | Taxikárska vzdialenosť a množstvo zdrojov | $\lvert a_x - c_x \rvert + \lvert a_y - c_y \rvert + (z_{max} - z_a)/z_{max}$ |
| 5 | Taxikárska vzdialenosť, množstvo zdrojov a výškový rozdiel | $\lvert a_x - c_x \rvert + \lvert a_y - c_y \rvert + (z_{max} - z_a) / (z_{max} + v_{max} + 1) + \lvert v_a - v_c \rvert / (z_{max} + v_{max} + 1)$ |
| 6 | Taxikárska vzdialenosť a neškálované množstvo zdrojov | $\lvert a_x - c_x \rvert + \lvert a_y - c_y \rvert -$ |

Príspevok výškového rozdielu a množstva zdrojov, okrem poslednej uvedenej heuristiky, je prepočítaný do rozsahu <0,1). Je to z dôvodu, že vzdialenosť je primárna heuristika a príspevok ostatných častí len pomáha vybrať medzi trasami, ktoré majú rovnakú hodnotu heuristickej funkcie.

### *2.6.3 Metriky*

Základnými sledovanými metrikami budú údaje o kvalite trasy a aj o pamäťovej a výpočtovej náročnosti. Sledované údaje sú uvedené v tabuľke:

**Tab. 9 Sledované metriky**

| Metrika | Výpočet |
|---|---|
| Úspešnosť | Či algoritmus našiel na mape cestu, resp. koľkokrát našiel cestu počas testovania. |
| Celkové náklady | Celkové náklady pre prechod trasou . |
| Dĺžka trasy | Počet dlaždíc v trase. |
| Analyzované trasy | Počet trás, ktoré boli počas hľadania vygenerované a spracované, vrátane dominovaných trás. |
| Veľkosť mapy hľadania | Počet trás v mape pri skončení prehľadávania. |
| Maximálny počet trás v bode | Najvyšší počet trás v mape hľadania na jednom bode pri skončení prehľadávania. |

Výsledky štatistík budú spriemerované z viacerých behov, pričom pôjde o stovky behov.

## 2.7 Dizajn aplikácie

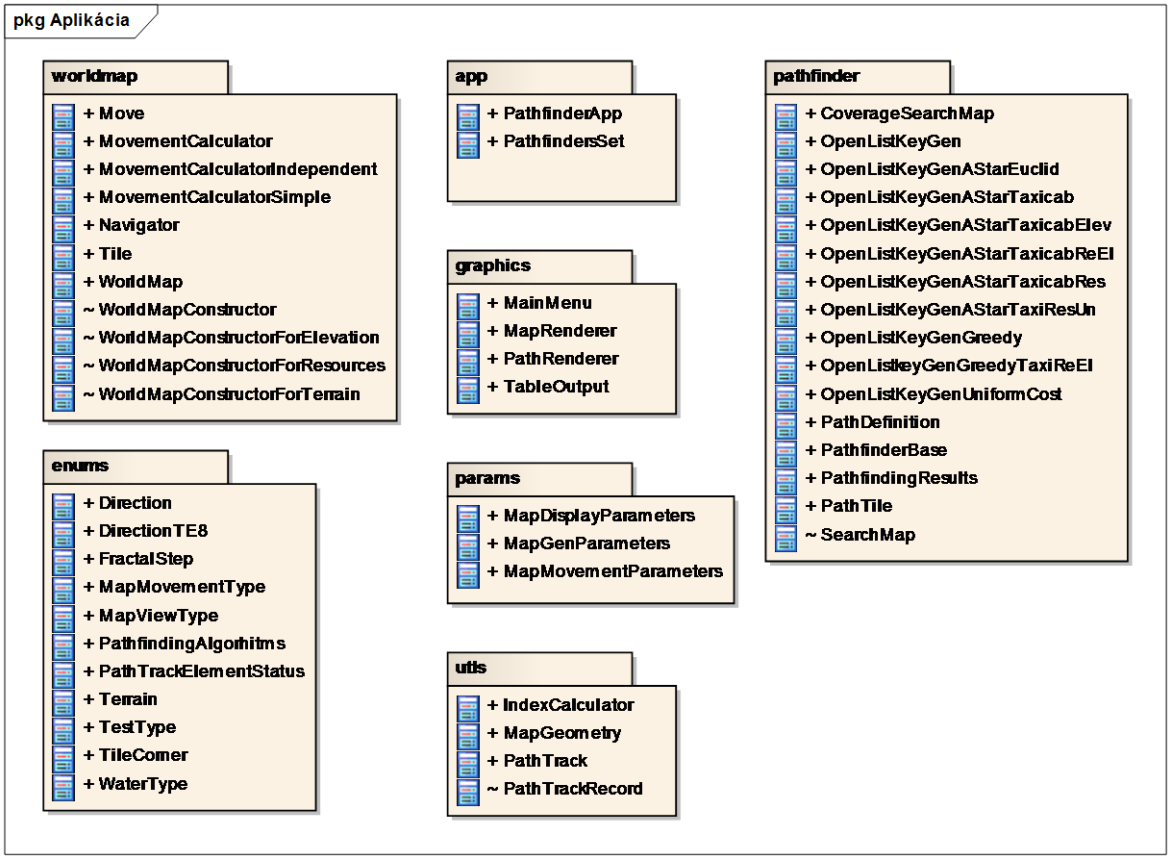
Aplikácia na testovanie algoritmov a metrík je implementovaná v jazyku Java. Na generovanie náhodných čísiel sú použité knižnice a triedy z Montrealskej univerzity (SSJ). Na zobrazovanie sa používajú štandardné knižnice Swing a AWT.

Program je rozdelený na balíky (Obr. 39 Štruktúra balíkov aplikácie na porovnanie algoritmov prehľadávania ).

### *2.7.1 Balík worldmap*

Balík „worldmap“ obsahuje implementáciu mapy, navigácie po mape, výpočtu nákladov a spotreby. Základom je dvojrozmerné pole dlaždíc. Dlaždica, trieda Tile obsahuje informácie o výške, teréne a prítomnosti zdroja.

Trieda „Worldmap“ obsahuje pole dlaždíc, základné informácie o mape a parametre, ktoré sa používajú na generovanie mapy.




**Obr. 39 Štruktúra balíkov aplikácie na porovnanie algoritmov prehľadávania**

#### *2.7.1.1 Pohyb po mape*

Výpočet nákladov a spotreby je implementovaný triedou „MovementCalculator“, pričom trieda mapy má vždy asociovaný práve jeden objekt tejto triedy alebo tried z nej odvodených. Riešenie má za účel umožniť zmenu funkcií jednoduchou náhradou používaného objektu. Implementované sú tri typy pohybu (Tab. 10 Implementované triedy pre výpočet nákladov a spotreby pri pohybe po mape). Na rozšírenie typov pohybu stačí odvodiť novú triedu z triedy „MovementCalculator“ alebo z nej odvodených tried a pri vytváraní mapy zadať objekt tejto triedy objektu triedy „WorldMap“.

**Tab. 10 Implementované triedy pre výpočet nákladov a spotreby pri pohybe po mape**

| Trieda | Náklady | Spotreba |
|---|---|---|
| MovementCalculator | Konštanta 1 | Konštanta 1 |
| MovementCalculator Simple | Náklady na pohyb cieľovej dlaždice vynásobené modifikátorom podľa veľkosti zmeny výšky, pričom zmena smerom hore zväčší náklady viac ako zmena smerom dole | Ako náklady |
| MovementCalculator Independent | Náklady na pohyb cieľovej dlaždice | Veľkosť zmeny výšky, ale len pri pohybe nahor, inak 0 |

Implementované sú tri typy terénu. Základný terén, kde je pohyb najmenej nákladný a náklady na prechod sú rovné 1. Náročný terén má náklady na pohyb rovné 4. Posledný typ terénu je nepriechodný terén.

Dopĺňanie terénu je limitované najmä nutnosťou mať grafickú reprezentáciu povrchu. Bez zobrazovania mapy je doplnenie typov terénov jednoduché, stačí rozšíriť enumeráciu typov a určiť všetkým typom pravdepodobnosť výskytu pri generovaní a náročnosť prechodu.

#### *2.7.1.2 Generovanie mapy*

Generovanie mapy je rozčlenené do troch tried: začína sa generovaním výšky (trieda „WorldMapConstructionForElevation“), pokračuje generovaním terénu (trieda „WorldMapConstructorForTerrain“) a nakoniec sa vygenerujú zdroje (trieda „WorldMapConstructorForTerrain“).

Na generovanie výšok dlaždíc mapy je použitý rekurzívny fraktálový algoritmus (Olsen, 2004), upravený tak, aby susedné dlaždice mali rozdiel vo výškach najviac jedna. Pri teréne je implementovaný algoritmus, ktorý vytvára rôzne veľké súvislejšie plochy rovnakých terénov. Zdroje sú generované tak, aby pokryli mapu, ale zároveň neboli rozmiestnené pravidelne.

### *2.7.2 Balík pathfinder*

V balíku sú implementované vyhľadávacie algoritmy a heuristiky. Balík obsahuje aj pomocné triedy na zber štatistík a výmenu údajov medzi triedami. Základom sú triedy „PathfinderBase“ a „SearchMap“. Trieda „PathfinderBase“ obsahuje cyklus cez zoznam otvorených stavov, t.j. dlaždíc: vyberie prvý záznam zo zoznamu, skontroluje, či sa nedosiahol cieľ, vygeneruje potomkov a nechá ich spracovať v triede „SearchMap“.

**Tab. 11 Triedy generujúce kľúč do zoznamu otvorených stavov**

| Trieda | Algoritmus a heuristika |
|---|---|
| OpenListKeyGen | Najskôr do hĺbky |
| OpenListKeyGenUniformCost | Rovnomerné náklady |
| OpenListKeyGenGreedy | Chamtivé hľadanie s heuristiku: Euklidovská vzdialenosť k cieľu |
| OpenListKeyGenGreedyTaxiReEl | Chamtivý hľadanie s heuristikou: Taxikárska vzdialenosť k cieľu, normalizovaný rozdiel kapacity a množstva zdroja v stave a normalizovaný výškový rozdiel medzi cieľom a stavom |
| OpenListKeyGenAStarEuclid | A* s heuristikou: Euklidovská vzdialenosť |
| OpenListKeyGenAStarTaxicab | A* s heuristikou: Taxikárska vzdialenosť k cieľu |
| OpenListKeyGenAStarTaxicabElev | A* s heuristikou: Taxikárska vzdialenosť k cieľu a normalizovaný výškový rozdiel medzi cieľom a stavom |
| OpenListKeyGenAStarTaxicabRes | A* s heuristikou: Taxikárska vzdialenosť k cieľu a normalizovaný rozdiel kapacity a množstva zdroja v stave |
| OpenListKeyGenAStarTaxicabReEl | A* s heuristikou: Taxikárska vzdialenosť k cieľu, normalizovaný rozdiel kapacity a množstva zdroja v stave a normalizovaný výškový rozdiel medzi cieľom a stavom |
| OpenListKeyGenAStarTaxicabResUn | A* s heuristikou: Taxikárska vzdialenosť k cieľu a (nenormalizovaný) rozdiel kapacity a množstva zdroja v stave |

Vygenerovanie potomkov zabezpečí trieda „Navigator“ skúmanej mapy, keďže táto trieda nájde také susedné dlaždice, ktoré sú priechodné a vypočíta pre každého suseda

náklady a cenu prechodu. Pre nové trasy, vygenerovaných potomkov, sa vypočíta aj hodnota, s ktorou sa zaradia do zoznamu otvorených v prípade, ak sa budú zapisovať.

Trieda „SearchMap“ implementuje strom prehľadávania pre hľadanie trasy, t.j. ide reálne o pole rovnakých rozmerov, ako má skúmaná mapa. V každom bode poľa je však pole skúmaných stavov, t.j. trás. Pri spracovaní nového stavu (trasy) sa skontroluje dominancia: odstránia sa dominované stavy, ak sú tie aj v zozname otvorených stavov, tak aj z neho alebo sa skončí spracovanie nového stavu, pokiaľ je dominovaný aspoň jedným existujúcim stavom v danom bode mapy.

Pokiaľ nový stav nie je dominovaný žiadnym existujúcim stavom, tak sa zapíše do mapy prehľadávania a doplní sa aj do zoznamu otvorených stavov podľa vygenerovaného kľúča. Na generovanie kľúča sa použije objekt triedy „OpenListKeyGen“ alebo triedy z neho odvodenej. Pri vytváraní objektu triedy „PathfinderBase“ sa určí, aký spôsob výpočtu sa použije tak, že sa ako parameter vloží vhodný objekt na výpočet kľúča. Trieda má jedinú metódu: generateKey(PathTile), kde trieda „PathTile” reprezentuje stav, resp. vygenerovanú trasu. Implementované triedy sú uvedené v tabuľke Tab. 11.

### *2.7.3 Balík app*

Obsahuje triedu „PathfinderApp“ zabezpečujúcu biznis logiku aplikácie. Generovanie máp, spúšťanie simulácii, poskytovanie informácií grafickému používateľskému rozhraniu, vrátanie miešania obrazových údajov mapy a trasy, atď. Na obsluhu množiny objektov pre hľadanie trasy (objektov triedy „PathfinderBase“) je vytvorená trieda „PathfidersSet“.

### *2.7.4 Balík graphics*

Balík obsahuje triedy pre používateľské rozhranie a zobrazovanie informácií. Trieda „MainMenu“ vytvára okno a menu aplikácie a spracováva voľby používateľa. Trieda „MapRenderer“ generuje obraz mapy a trieda „PathRenderer“ nájdenej trasy a stromu prehľadávania. Trieda „TableOutput“ je určená na zobrazenie výsledkov hľadania trasy v tabuľke.

### *2.7.5 Balíky enums, params a utils*

Balík enums obsahuje definície enumerácií používaných v aplikácii. Niektoré obsahujú aj určité riadiace informácie. Napríklad enumerácia „Directions“, ktorá obsahuje svetové strany (východ, západ, sever a juh), obsahuje aj údaje o zmene súradníc pri pohybe daným smerom: juh je zmena súradnice y o -1 a zmena x o 0 atď.

Balík params obsahuje parametre pre mapu, pohyb po mape a pre zobrazovanie. Pre mapu sa tu evidujú rozmery mapy, počet výšok, pravdepodobnosti používané pri generovaní mapy. Pre pohyb sa v balíku definujú premenné o nákladoch na pohyb po jednotlivých terénoch, kapacitu zdroja, atď. Trieda pre parametre zobrazenia obsahuje informácie p veľkosti zobrazovanej dlaždice, špecifikáciu farieb pre nadmorské výšky, atď.

Balík utils obsahuje triedy, ktoré sa používajú pri generovaní mapy.

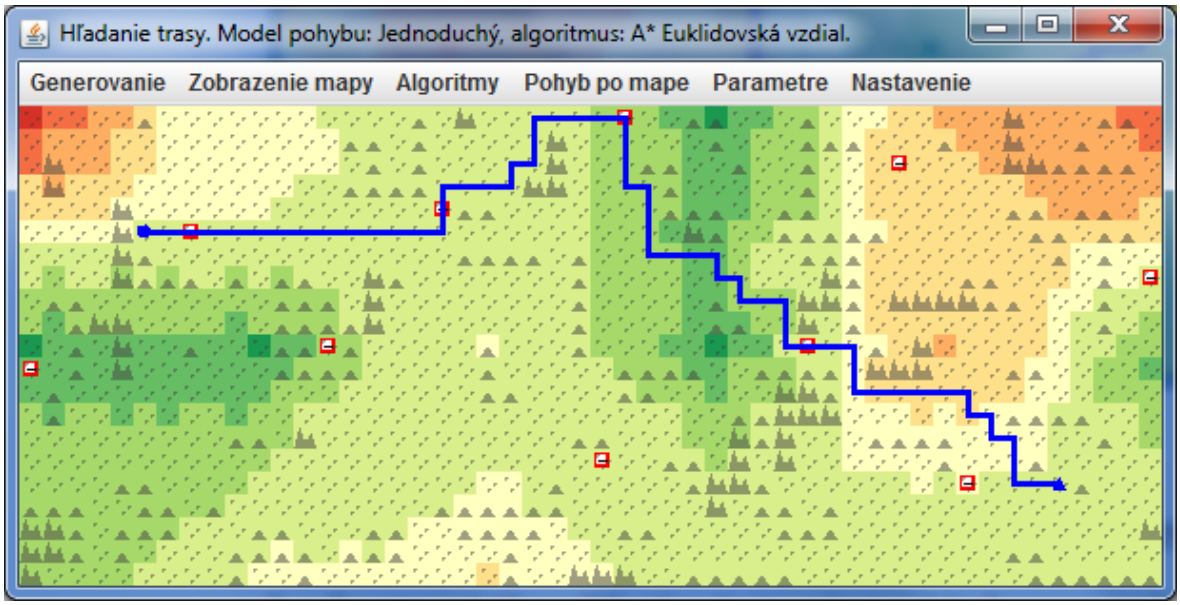
# 3 Výsledky práce a diskusia

Program sa ovláda v jednom okne, kde je zobrazená mapa s nájdenou cestou a prístup k všetkým možnostiam nastavenia (Obr. 40 Program na porovnávanie algoritmov na hľadanie trasy). Na hornom okraji je uvedený model pohybu a algoritmus, ktorý našiel aktuálne zobrazenú trasu. Počas simulácie je na hornom okraji uvedené percento dokončenia.

## 3.1 Aplikácia

Aplikácia je implementovaná v jazyku Java, na jej spustenie nie je potrebná inštalácia, stačí mať k dispozícii java runtime verzie 7. Program sa spustí príkazom „java –jar jbeno_pathfinding.jar“ alebo dvojklikom na súbor.

Program sa ovláda v jednom okne, kde je zobrazená mapa s nájdenou cestou a prístup k všetkým možnostiam nastavenia (Obr. 40 Program na porovnávanie algoritmov na hľadanie trasy). Na hornom okraji je uvedený model pohybu a algoritmus, ktorý našiel aktuálne zobrazenú trasu. Počas simulácie je na hornom okraji uvedené percento dokončenia.




**Obr. 40 Program na porovnávanie algoritmov na hľadanie trasy**

### *3.1.1 Menu Generovanie*

Menu Generovanie obsahuje dve možnosti:

- Generovanie mapy,
- Simulácia.

Pri výbere Generovanie mapy sa vytvorí nová mapa podľa aktuálne zadaných údajov a spustí sa vyhľadávanie trasy všetkými definovanými algoritmami.

Voľba Simulácia spustí simuláciu zadanej dĺžky.

### *3.1.2 Menu Zobrazenie mapy*

Aplikácia podporuje tri typy zobrazenia mapy:

- Terén,
- Výška,
- Povrch.

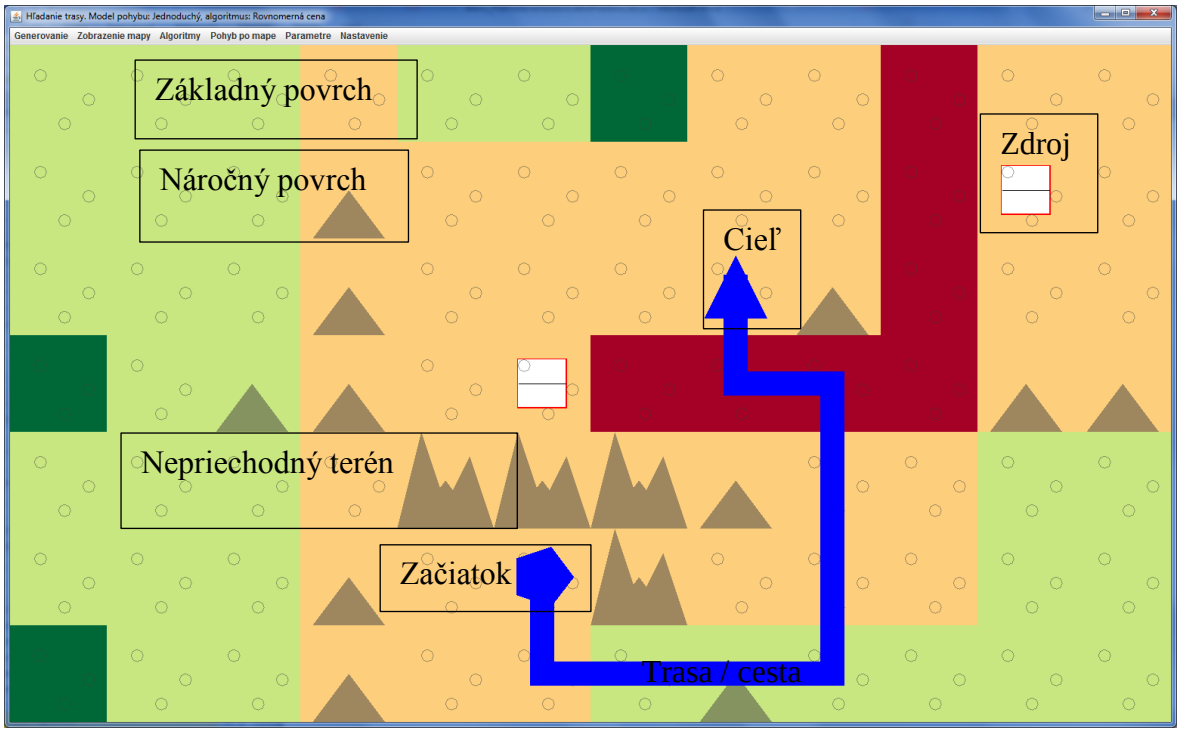
#### *3.1.2.1 Voľba Terén*

Základné zobrazenie je zobrazenie typu Terén. Výšky sú na mape rozlíšené farebnými kódmi a terén grafickými značkami. Grafické zobrazenie je škálovateľné.

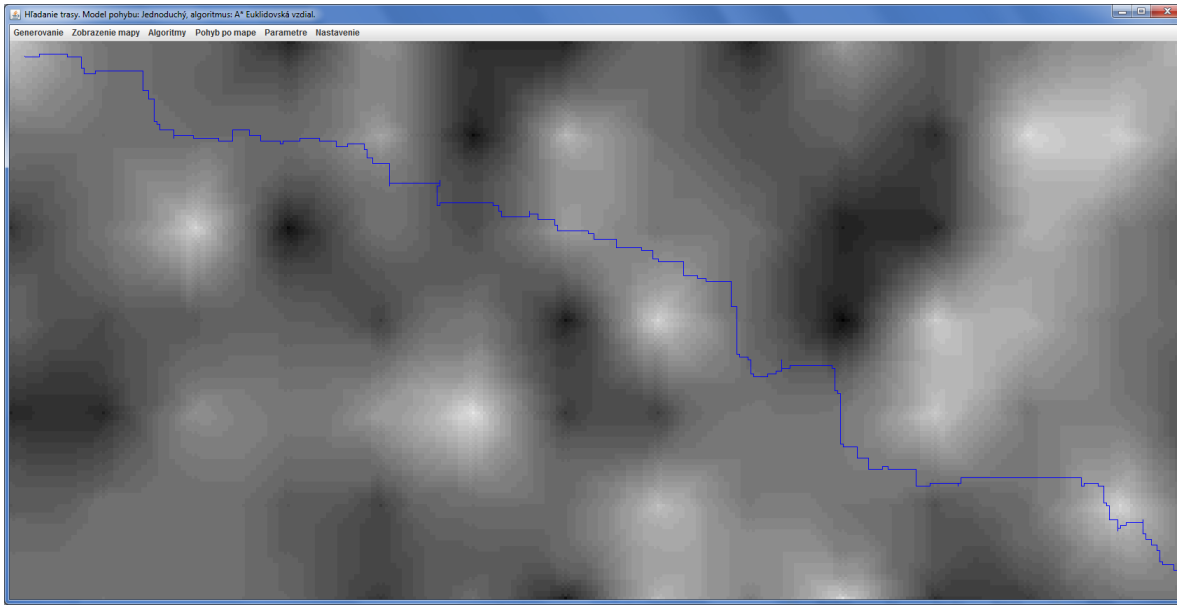
Na obrázku Obr. 41 je použitá veľkosť dlaždice 128 bodov a sú doplnené popisy pre elementy mapy: tri základné terény (základný, náročný a nepriechodný), prítomnosť zdroja na dlaždici, začiatočný a cieľový bod trasy a nájdená trasa. Mapa na obrázku obsahuje len 4 výškové úrovne.

V móde Výška sa mapa zobrazí v odtieňoch šedej, kde najtmavšie, čierne dlaždice majú najmenšiu výšku a s rastúcou výškou dlaždice sa dlaždica zobrazuje svetlejšie. Biele dlaždice sú najvyššie. Na obrázku Obr. 42 je v šedej škále zobrazená mapa v rozmeroch 450x221 dlaždíc veľkosti 4 obrazových bodov a s 34 úrovňami výšky.

Výška mapy sa udáva buď ako počet diskrétnych výšok alebo ako najvyššia povolená hodnota výšky. Platí, že počet výšok = maximálna výška + 1, pretože výšky sú celé čísla od 0 do najvyššej výšky vrátane.




**Obr. 41 Zobrazenie mapy a nájdenej trasy**




**Obr. 42. Mapa zobrazená v šedej škále reprezentujúcej výšku bodov**

Posledný typ zobrazenia je zobrazenie povrchu. Ide o zobrazenie ako v móde pre terén, ale bez uvedenia grafických značiek terénov. Zobrazená je tak farebná mapa, kde farby sú určené podľa výšky dlaždice.

#### 3.1.2.2 *Výber farieb*

Farby pre jednotlivé výšky boli vybrané podľa návrhu na stránke http://www.ColorBrewer.org (Brewer, 2012). Prebraná je schéma s 11 hodnotami (Obr. 43 Základná farebná škála pre zobrazenie výšok v mape (Brewer, 2012)).

**Obr. 43 Základná farebná škála pre zobrazenie výšok v mape (Brewer, 2012)**

Pokiaľ je použitý iný počet výšok, schéma sa buď zužuje alebo rozširuje. Na generovanie farieb sa používa vážený súčet jednotlivých zložiek základných farieb.

Generovanie farieb reálne použitých na zobrazenie mapy sa vykoná tak, že sa mapuje celočíselná škála povolených výšok mapy do intervalu 0 až 10 v množine racionálnych čísel. Výška i pri maximálnej výške $v_m$ sa mapuje na číslo $i_b$:

$$i_b = 10\frac{\text{í}}{v_m}$$

Pokiaľ je $i_b$ celé číslo, tak farba i sa znázorňuje farbou i. Pokiaľ nie je $i_b$ celé číslo, tak získa z najbližších farieb základnej schémy. Najbližšie farby možno získať funkciami „floor“ a „ceiling“, t.j. ako najvyššie celé číslo menšie ako x a najnižšie celé číslo väčšie ako x. Matematicky možno definovať pre x funkcie „floor“ a „ceiling“ nasledovne (Porubský, 2012):

$$x \in R; floor(x) = \lfloor x \rfloor = max\{m \in Z | \, m < x\}$$

$$x \in R; ceiling(x) = \lceil x \rceil = min\{m \in Z | \, m > x\}$$

Váha pre farbu základnej škály s indexom $\lfloor x \rfloor$ je daná vzťahom $w_{fx} = x - \lfloor x \rfloor$ a pre farbu s indexom $w_{cx} = \lceil x \rceil - x$. Výsledná farba pre výšku i je potom daná výpočtom váhovaného priemeru pre každú farebnú zložku: červená, zelená a modrá. Napr. modrá zložka ($m_i$) farby pre výšku i sa počíta z modrých zložiek bázických farieb (cib = $\lceil i_b \rceil$):

$$m_i = w_{cib} m_{cib} + w_{fib} m_{fib}$$

### 3.1.3 *Menu Algoritmy*

Menu sa používa na

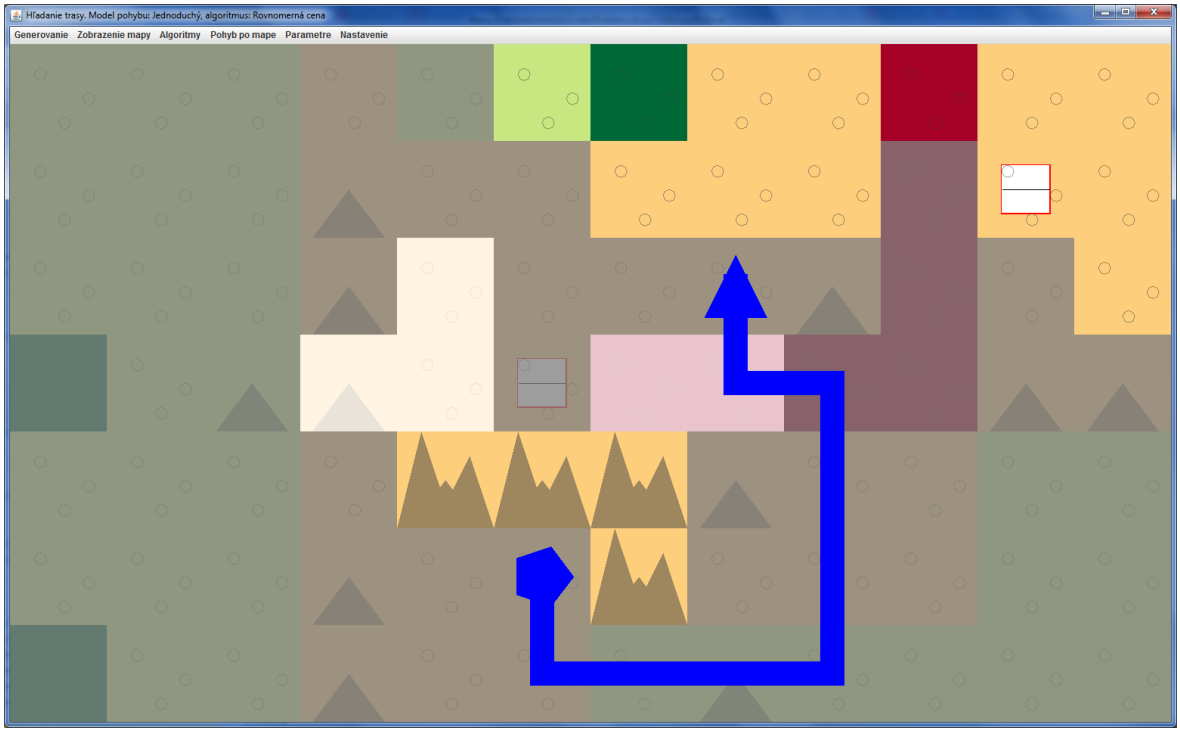
- výber algoritmu, ktorým vygenerovaná trasa sa má zobraziť na v hlavnom okne,
- výber, či sa má zobraziť aj mapa prehľadávania aktuálne zvoleného algoritmu,
- zobrazenie výsledkov.

#### *3.1.3.1 Voľby algoritmov*

Algoritmy sa volia kliknutím na názov algoritmu. Na použitie zvoleného algoritmu je však potrebné vygenerovať novú mapu alebo spustiť simuláciu.

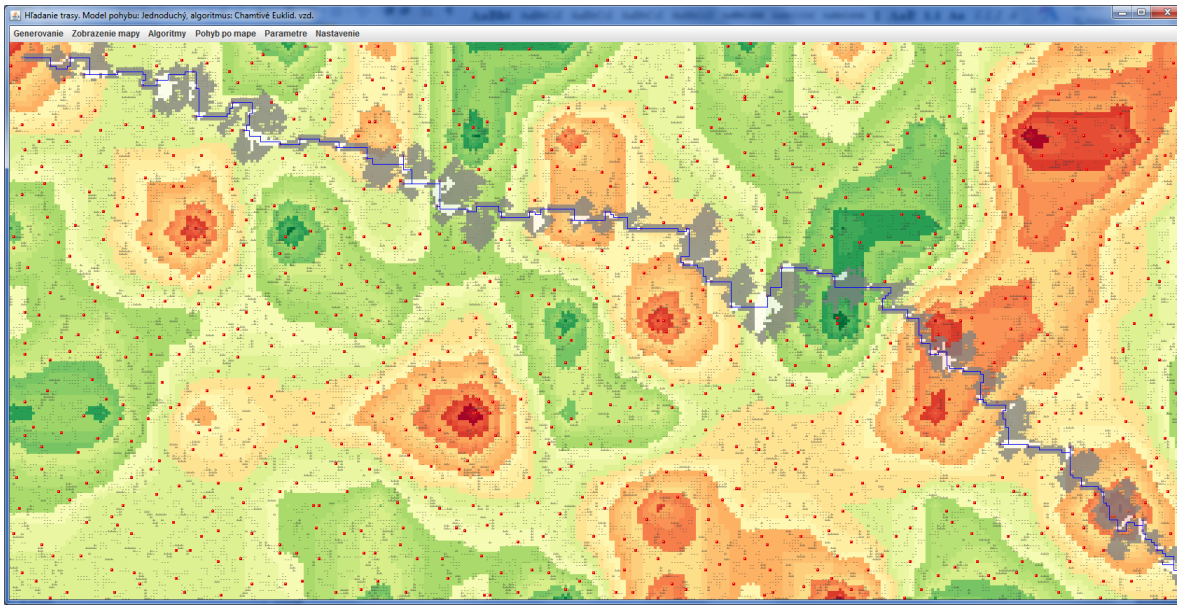
#### *3.1.3.2 Voľba Zobraziť mapu prehľadávania*

Mapa prehľadávania (Obr. 44 Mapa prehľadávania) zobrazí stav na konci prehľadávania. Pre každý jej bod sa určí jas, resp. farba na šedej škále podľa toho, koľko stavov – zvažovaných trás – je na ňom. Body mapy, ktoré mapa prehľadávania neobsahuje resp. nie je v ňom ani jedna zvažovaná trasa, sú úplne priehľadné a zobrazuje sa tak cez ne mapa (sveta). Ostatné body, kde je aspoň jedna zvažovaná trasa majú nastavenú slabú priehľadnosť a mapa sveta pod mapou prehľadávania je čiastočne vidieť aj terén.



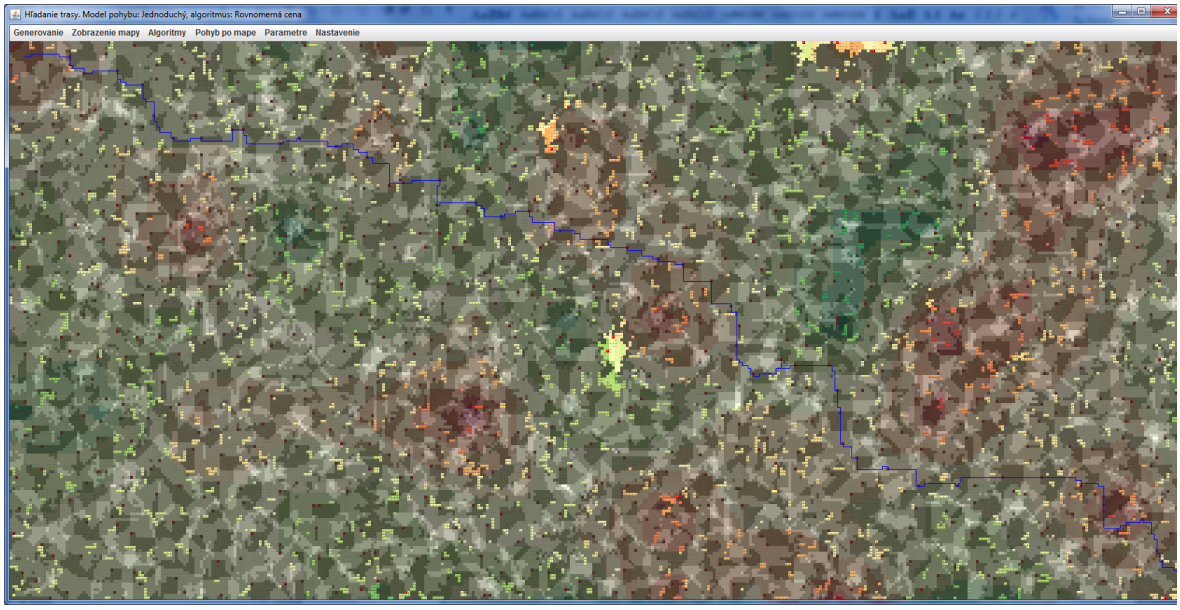

**Obr. 44 Mapa prehľadávania**

Z mapy prehľadávania sa jednoducho dá zistiť náročnosť konkrétneho algoritmu na výpočtové a pamäťové zdroje.




**Obr. 45 Mapa prehľadávania Chamtivého prehľadávania**

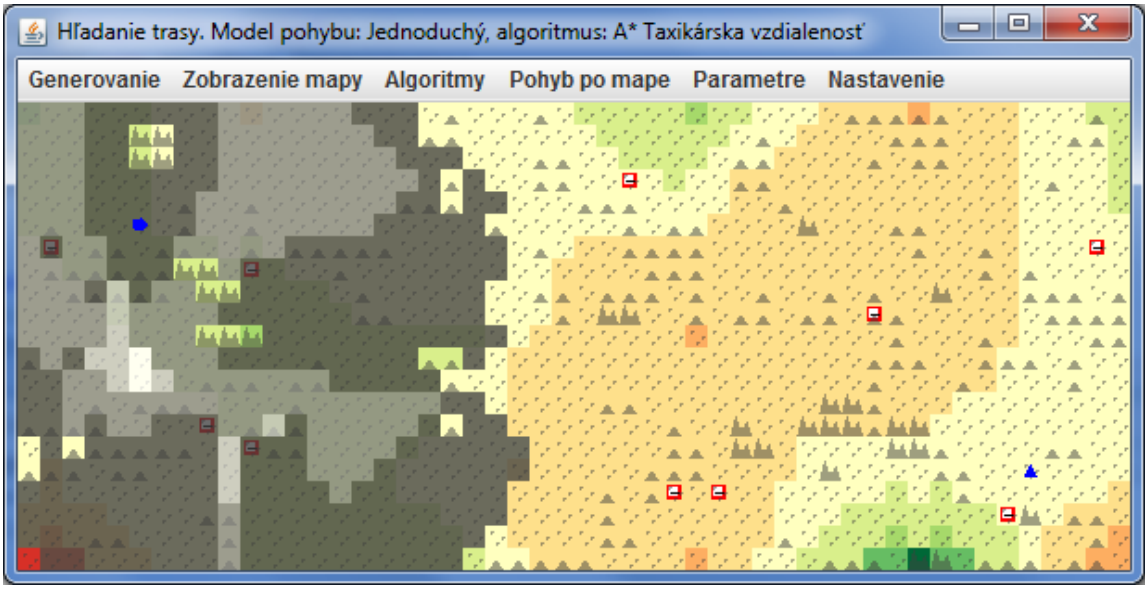
Na obrázkoch je porovnanie mapy prehľadávania na rovnakej mape pre Chamtivé prehľadávanie a Rovnomerné náklady (Obr. 45 Mapa prehľadávania Chamtivého prehľadávania, Obr. 46 Mapa prehľadávania algoritmu Rovnomerných nákladov). Už na pohľad je zjavné, že algoritmus Rovnomerné náklady má oveľa väčšie nároky na zdroje ako Chamtivé prehľadávanie.




**Obr. 46 Mapa prehľadávania algoritmu Rovnomerných nákladov**

V prípade, že algoritmus nenájde trasu, je možné mapou prehľadávania identifikovať, do akej vzdialenosti sa algoritmus dostal (Obr. 47 Mapa prehľadávania pri neúspešnom hľadaní trasy). Všetky implementované algoritmy používajú zoznam spracovaných stavov, v tomto prípade mapu prehľadávania, a v prípade neúspechu mapy prehľadávania všetkých algoritmov pokrývajú rovnakú časť mapy.

Pokiaľ by sa implementovali algoritmy ako Horolezecké prehľadávanie alebo obojsmerné prehľadávanie, tak ich mapy prehľadávania by sa líšili od ostatných algoritmov.




**Obr. 47 Mapa prehľadávania pri neúspešnom hľadaní trasy**

### *3.1.3.3 Menu Pohyb po mape*

Menu obsahuje výber z implementovaných spôsobov výpočtu nákladov a spotreby. Pri výbere z možností: Konštantný, Jednoduchý a Nezávislý sa voľba prejaví až pri najbližšom generovaní mapy alebo simulácii.

### *3.1.3.4 Menu Parametre*

Menu umožňuje zmeniť parametre generovania mapy alebo jej zobrazenia. Nastaviť možno:

- veľkosť dlaždice v obrazových bodoch pri jej renderovaní,
- maximálna výška dlaždice v mape,
- kapacita zdroja pri hľadaní trasy,
- dĺžka simulácie.

Zmeny sa prejavia až pri najbližšom generovaní mapy alebo pri spustení simulácie. Okrem dĺžky simulácie sú všetky zmeny parametrov realizované cez dialóg výberu z preddefinovaných možností.

#### *3.1.3.5 Menu Nastavenie*

Menu obsahuje jedinú možnosť, a to zmenu maximálnej povolenej veľkosti okna aplikácie. Nastavenie sa prejaví ihneď po zatvorení okna vygenerovaním novej mapy. Pri niektorých nastaveniach môže aplikácia nereagovať aj niekoľko desiatok sekúnd, keďže sa prepočítavajú trasy všetkými implementovanými algoritmami.

Nastavuje sa len šírka okna (v mape sa tento údaj označuje ako dĺžka mapy) a výška okna (v mape šírka) sa určí ako polovica šírky.

#### *3.1.3.6 Vzťah veľkosti okna, maximálnej výšky a veľkosti dlaždice*

Zo spôsobu generovania mapy vyplýva nutnosť mať dĺžku mapy určenú ako násobok najvyššej povolenej výšky. Nech $m_d$ je dĺžka mapy v dlaždiciach a $m_s$ je jej šírka, h je najvyššia povolená výška, s je dĺžka/veľkosť dlaždice v obrazových bodoch, w je šírka okna v obrazových bodoch a $w_m$ je najväčšia povolená šírka okna. Pre šírku okna tak platí, že:

$$m_d s = w \leq w_m$$

$$m_d = hn \;\; a \; m_s = 1 + \left\lfloor \frac{hn}{2} \right\rfloor, n \in N, n > 1$$

Hodnota n, z ktorej sa určí dĺžka a šírka mapy, sa určí nasledovne:

$$n = max\left\{ i \in N \mid \left\lfloor \frac{w_m}{2hi} \right\rfloor > 0 \wedge shi \leq w_m \right\}$$

Skutočná šírka okna je tak menšia alebo rovná ako vybraná hodnota z menu. Pri výbere najvyššej povolenej výšky a veľkosti dlaždice sa môže zobraziť chyba v prípade, ak zadané hodnoty nespĺňajú podmienku pre maximálnu šírku okna.

## 3.2 Testy algoritmov

Testy algoritmov sa vykonajú na štvorcových mapách s prechodom len cez hrany. Zdroje sú generované po jednom do neprekrývajúcich sa štvorcov pokrývajúcich celú mapu s hranou 10 dlaždíc. Veľkosť mapy je daná počtom výškových úrovní pri okne veľkosti 1800 obrazových bodov a veľkosti dlaždice 4 body. Veľkosti tabuliek sú:

- 450x226 pre 4 výšky;
- 450x221 pre 11 výšok a

- 429x199 pre 34 výšok.

Každý test bude spočívať v spustení 100 alebo 1000 behov, t.j. vygeneruje sa 100 resp. 1000 máp a na nich sa všetky algoritmy spustia hľadanie trasy medzi rovnakými bodmi. Dĺžka simulácie je obmedzená výpočtovými a pamäťovými možnosťami dostupných počítačov. Najmä pri veľkých mapách (450x221) sú algoritmy ako Najskôr do hĺbky alebo Rovnomerné náklady veľmi náročné na zdroje.

Algoritmy sa testujú za rôznych nastavení:

- spôsobu pohybu po mape;
- kapacity zdroja;
- počtu výškových úrovní.

Algoritmy budú v tabuľkách s výsledkami testov z dôvodu šetrenia miestom označované podľa tabuľky. (Tab. 12 Označenie algoritmov)

**Tab. 12 Označenie algoritmov**

| Algoritmus | Označenie |
|---|---|
| Rovnomerné náklady | RN |
| Najskôr do šírky | DŠ |
| A* Euklidovská vzdialenosť | A*E |
| A* Taxikárska vzdialenosť | A*T |
| A* Taxik. vzdial. a výška | A*TV |
| A* Taxik. vzdial. a zdroje | A*TZ |
| A* Taxi. vzd., zdroje a výška | A*TZV |
| A* Taxi. vzd. a zdroje neškál. | A*TZVn |
| Chamtivé Euklidovská vzdial.. | ChE |
| Chamtivé Taxi.v., zdr. a výška | ChTZV |

## 3.3 Výstupy simulácií

Kapitola obsahuje výsledky jednotlivých experimentov. V tabuľkách sú uvedené výsledky pre všetky algoritmy a sledované metriky. V tabuľkách je zvýraznený najlepší výsledok z pohľadu celkových nákladov. Pokiaľ je viacero trás s rovnakými výsledkami, tak je z nich ešte viac zvýraznený algoritmus, ktorý má najmenší počet analyzovaných trás.

Názvy kapitol sú odvodené od nastavenia aplikácie počas simulácie, obsahuje tak model pohybu, kapacitu zdroja najväčšiu výšku v mape.

### *3.3.1 Scenáre s jednoduchým modelom pohybu*

Jednoduchý model pohybu počíta náklady na základe náročnosti terénu a túto hodnotu následne modifikuje v prípade, že došlo k prechodu na dlaždicu inej výšky. Pri prechode nižšie sa náklady navýšia o 25% pre každú výškovú úroveň zmeny a pri prechode vyššie sa zvýšia až o 50% za každú výškovú úroveň[9]. Spotreba pri prechode medzi dlaždicami je zhodná s nákladmi na daný prechod.

Vzhľadom na to by mali mať výhodu heuristiky počítajúce aj výškou resp. s rozdielom výšky spracovávaného stavu a cieľa.

#### *3.3.1.1 Scenár: Jednoduchý 15, 10*

Pre scenár s jednoduchým modelom pohybu, 11 úrovňami výšky a s kapacitou zdroja 15 sa na mapách najväčších rozmerov ani raz z 1000 pokusov nepodarilo nájsť cestu. Keďže zdroje sú umiestňované do štvorcov s hranou 10 dlaždíc, tak priemerne je taxikárska vzdialenosť medzi zdrojmi 15: 4 susedné zdroje cez hrany sú priemerne vzdialené 10 a 4, diagonálne sú priemerne 20.

Ak sa však do nákladov, a teda aj spotreby, započíta terén, tak je malá šanca, že medzi dvomi zdrojmi existuje trasa s nákladmi nižšími ako je kapacita. Pri väčších mapách tak prudko klesá pravdepodobnosť existencie trasy medzi začiatočným a koncovým bodom, ktorá by spĺňala podmienku na zdroje.

**Tab. 13 Výsledky algoritmov v scenári Jednoduchý, 15, 10 pre rôzne veľkosti mapy**

| Veľkosť mapy | Úspešné | Úspešné v % |
|---|---|---|
| 450x221 | 0 | 0% |
| 110x51 | 2 | 0,2% |
| 50x21 | 43 | 4,30% |
| 40x21 | 105 | 10,5% |
| 20x21 | 830 | 83% |

Pri menších rozmerom je pravdepodobnejšia mapa, kde je možné nájsť cestu aj s kapacitou 15 pri jednoduchom type pohybu. V tabuľke je uvedený počet nájdených trás pri 1000 generovaniach (Tab. 13). Výsledky pre menšie rozlíšenia ovplyvňuje fakt, že trasa sa hľadá z bodu (5,5) a cieľ je na súradniciach (dĺžka mapy – 5, šírka mapy - 5), takže pri

[9] Keďže však algoritmus generovania výšok je vytvorený tak, aby susedné dlaždice malo rozdiel najviac jenu úroveň tak náklady a aj spotreba sa modifikujú iba o 25 resp. 50%

mape (20,21) je vzdialenosť cieľa a zdroja v taxikárskej geometrii rovná 10, teda menej ako sú náklady a spotreba, aj s jedným prechodom cez náročnejší terén.

#### *3.3.1.2 Scenár: Jednoduchý, 20, 3*

Simulácia mala len 100 opakovaní. Aj napriek nízkemu počtu výškových úrovní dosiahol najlepší výsledok algoritmus A* s heuristikou, ktorá pracuje aj s výškovým rozdielom spracovávaného bodu a cieľa. Dva najlepšie výsledky (A*TZ a A*TZV) počítajú v heuristickej funkcii aj so zdrojmi.

**Tab. 14 Výsledky algoritmov v scenári Jednoduchý, 20, 3**

| Algoritm. | Úspešné | Náklady | Dĺžka | Analyz. | V mape | Max trás |
|---|---|---|---|---|---|---|
| RN | 87 | 844,186782 | 762 | 674367 | 191530 | 6 |
| DŠ | 87 | 872,597701 | 733 | 5741214 | 191736 | 6 |
| A*E | 87 | 844,186782 | 762 | 596546 | 170149 | 6 |
| A*T | 87 | 844,186782 | 761 | 678066 | 192082 | 6 |
| A*TV | 87 | 844,186782 | 762 | 515487 | 147284 | 6 |
| A*TZ | 87 | 844,186782 | 762 | 514702 | 147493 | 6 |
| A*TZV | 87 | 844,186782 | 762 | 514530 | 147461 | 6 |
| A*TZVn | 87 | 845,652299 | 762 | 576640 | 149489 | 6 |
| ChE | 87 | 1188,442530 | 911 | 72923 | 8167 | 3 |
| ChTZV | 87 | 1262,097700 | 1033 | 129821 | 9761 | 4 |

#### *3.3.1.3 Scenár : Jednoduchý, 20, 10*

Na rozdiel od scenárov jednoduchého modelu, dosahuje najlepšie výsledky heuristika používajúca primárne vzdialenosť a aj výšku, resp. prevýšenie, a dostupné zdroje.

**Tab. 15 Výsledky algoritmov v scenári Jednoduchý, 20, 10**

| Algoritm. | Úspešné | Náklady | Dĺžka | Analyz. | V mape | Max trás |
|---|---|---|---|---|---|---|
| RN | 84 | 821,083333 | 748 | 657695 | 186014 | 6 |
| DŠ | | | | | | |
| A*E | 84 | 821,083333 | 748 | 580050 | 164735 | 6 |
| A*T | 84 | 821,083333 | 748 | 661743 | 186554 | 6 |
| A*TV | 84 | 821,083333 | 748 | 495375 | 140840 | 6 |
| A*TZ | 84 | 821,083333 | 748 | 494232 | 141102 | 6 |
| A*TZV | 84 | 821,083333 | 748 | 493793 | 141014 | 6 |
| A*TZVn | 84 | 821,919643 | 748 | 554931 | 143142 | 6 |
| ChE | 84 | 1169,059524 | 906 | 67196 | 8137 | 3 |
| ChTZV | 84 | 1268,886905 | 1067 | 160552 | 11318 | 4 |

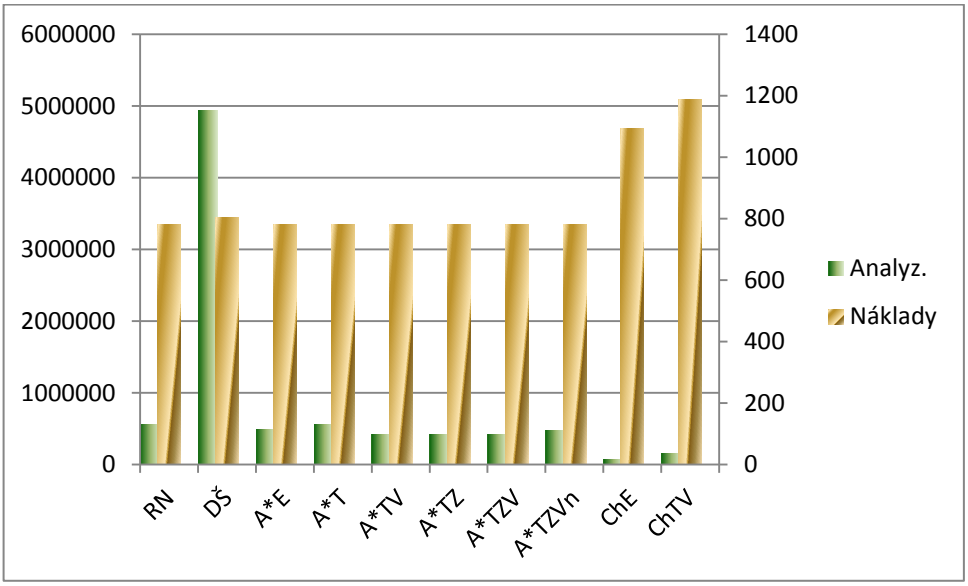
### *3.3.1.4 Scenár: Jednoduchý, 20, 33*

Výsledky testu sú skreslené chybami zaokrúhľovania, keďže algoritmus Prehľadávanie s rovnomernými nákladmi nemá najmenší priemerné minimálne celkové náklady na prechod trasou. Pre potreby štatistiky však budeme považovať hodnoty odlišujúce sa o jednu tisícu za zhodné

**Tab. 16 Výsledky algoritmov v scenári Jednoduchý, 20, 33**

| Algoritm. | Úspešné | Náklady | Dĺžka | Analyz. | V mape | Max trás |
|---|---|---|---|---|---|---|
| RN | 855 | 779,444 | 708 | 559705 | 158144 | 6 |
| DŠ | 855 | 804,858 | 682 | 4940621 | 158425 | 6 |
| A*E | 855 | 779,444 | 708 | 492126 | 139645 | 6 |
| A*T | 855 | 779,450 | 708 | 564933 | 159015 | 6 |
| A*TV | 855 | 779,443 | 708 | 422435 | 119888 | 6 |
| A*TZ | 855 | 779,446 | 708 | 421282 | 120125 | 6 |
| A*TZV | 855 | 779,444 | 708 | 420570 | 119972 | 6 |
| A*TZVn | 855 | 780,242 | 709 | 471999 | 122009 | 6 |
| ChE | 855 | 1094,720 | 845 | 67619 | 7630 | 3 |
| ChTZV | 855 | 1186,210 | 992 | 153972 | 10233 | 4 |

Pre názornosť je uvedený aj graf porovnávajúci celkové náklady nájdených trás a množstvo skúmaných (analyzovaných) trás (Obr. 48 Výsledky simulácie pre model: Jednoduchý, 20, 30). Základné charakteristiky majú všetky simulácie zhodné alebo veľmi podobné.




**Obr. 48 Výsledky simulácie pre model: Jednoduchý, 20, 30**

Najhoršie výsledky z pohľadu kvality trast dosahujú chamtivé algoritmy, majú však aj o jeden až dva rády lepšie výsledky v oblasti množstva vygenerovaných trás.

Prehľadávanie do šírky má slabé výsledky ako v kvalite mapy, tak i počte vygenerovaných stavov. Algoritmus Prehľadávanie s rovnomernými nákladmi dosahuje najmenej nákladné trasy, ale potrebuje na to preskúmať viac stavov.

#### *3.3.1.5 Scenár: Jednoduchý, 30, 10*

Pre model s kapacitou 30, 11 výškových úrovniach a jednoduchom modeli pohybu sa cesta cez mapu našla cesta v 99% prípadov.

**Tab. 17 Výsledky algoritmov v scenári Jednoduchý, 30, 10**

| Algoritm. | Úspešné | Náklady | Dĺžka | Analyz. | V mape | Max trás |
|---|---|---|---|---|---|---|
| RN | 990 | 711,694 | 670 | 831278 | 223285 | 7 |
| DŠ | 990 | 739,796 | 651 | 6008280 | 223386 | 7 |
| A*E | 990 | 711,694 | 670 | 744360 | 201078 | 7 |
| A*T | 990 | 711,695 | 670 | 834079 | 223358 | 7 |
| A*TV | 990 | 711,694 | 670 | 553034 | 150797 | 7 |
| A*TZ | 990 | 711,699 | 670 | 551664 | 151545 | 7 |
| A*TZV | 990 | 711,698 | 670 | 550775 | 151379 | 7 |
| A*TZVn | 990 | 715,503 | 672 | 1068375 | 157639 | 7 |
| ChE | 990 | 1119,130 | 759 | 15102 | 4052 | 2 |
| ChTZV | 990 | 1112,340 | 851 | 14790 | 4169 | 2 |

Chamtivé algoritmy potrebovali na nájdenie trasy až o dva rády menej analyzovaných trás ako algoritmus Prehľadávanie najskôr do šírky.

#### *3.3.1.6 Scenár: Jednoduchý, 2680, 10*

Pri nákladoch 2680 sa výsledky blížia štandardným výsledkom bez obmedzenia zdrojmi. Celkové náklady sú najnižšie zo všetkým simulácii s jednoduchým modelom pohybu po mape.

**Tab. 18 Výsledky algoritmov v scenári Jednoduchý, 2680, 10**

| Algoritm. | Úspešné | Náklady | Dĺžka | Analyz. | V mape | Max trás |
|---|---|---|---|---|---|---|
| RN | 100 | 678,4200 | 653 | 1085768 | 281637 | 10 |
| DŠ | 100 | 680,6450 | 651 | 6832293 | 281683 | 10 |
| A*E | 100 | 678,4200 | 653 | 981677 | 256608 | 10 |
| A*T | 100 | 678,4375 | 653 | 1089557 | 281702 | 10 |
| A*TV | 100 | 678,4200 | 653 | 591646 | 160275 | 9 |
| A*TZ | 100 | 678,4200 | 653 | 584079 | 161299 | 9 |
| A*TZV | 100 | 678,4200 | 653 | 583918 | 161260 | 9 |
| A*TZVn | 100 | 696,1275 | 662 | 3218303 | 165311 | 8 |
| ChE | 100 | 1135,9725 | 653 | 2522 | 1530 | 2 |
| ChTZV | 100 | 997,7950 | 677 | 2647 | 1833 | 2 |

### 3.3.2 Scenáre s nezávislým a konštantným modelom pohybu

Scenáre s nezávislým alebo konštantným pohybom sú menej citlivé na kapacitu zdroja alebo počet výškových úrovní mapy.

#### 3.3.2.1 Scenár: Nezávislý, 20, 10

Na rozdiel od scenárov jednoduchého modelu, dosahuje najlepšie výsledky heuristika používajúca primárne vzdialenosť a aj výšku, resp. prevýšenie, a dostupné zdroje, ale zlepšenie oproti heuristike bez zdroja je len minimálne.

**Tab. 19 Výsledky algoritmov v scenári Nezávislý, 20, 10**

| Algoritm. | Úspešné | Náklady | Dĺžka | Analyz. | V mape | Max trás |
|---|---|---|---|---|---|---|
| RN | 97 | 746,103093 | 702 | 840362 | 227687 | 8 |
| DŠ | 97 | 812,762887 | 661 | 8846100 | 227440 | 9 |
| A*E | 97 | 746,103093 | 702 | 744027 | 202610 | 8 |
| A*T | 97 | 746,134021 | 702 | 850835 | 227833 | 8 |
| A*TV | 97 | 746,103093 | 702 | 606189 | 160977 | 8 |
| A*TZ | 97 | 746,103093 | 702 | 588474 | 161564 | 8 |
| A*TZV | 97 | 746,103093 | 702 | 588100 | 161480 | 8 |
| A*TZVn | 97 | 747,896907 | 702 | 746800 | 166336 | 8 |
| ChE | 97 | 1275,41237 | 800 | 24937 | 4882 | 4 |
| ChTZV | 97 | 1480,69072 | 941 | 54313 | 6524 | 5 |

#### 3.3.2.2 Scenár: Nezávislý, 20, 33

Podľa očakávania malo zvýšenie počtu výšok dopad na celkové náklady nájdených trás, keďže pri vyššej spotrebe je potrebné častejšie prechádzať cez dlaždice so zdrojom.

**Tab. 20 Výsledky algoritmov v scenári Nezávislý, 20, 33**

| Algoritm. | Úspešné | Náklady | Dĺžka | Analyz. | V mape | Max trás |
|---|---|---|---|---|---|---|
| RN | 97 | 704,123711 | 659 | 713350 | 193188 | 8 |
| DŠ | 97 | 767,391753 | 621 | 7031619 | 192960 | 8 |
| A*E | 97 | 704,123711 | 660 | 631861 | 172044 | 8 |
| A*T | 97 | 704,154639 | 659 | 721658 | 193376 | 8 |
| A*TV | 97 | 704,123711 | 660 | 516093 | 137615 | 8 |
| A*TZ | 97 | 704,123711 | 660 | 504261 | 138326 | 8 |
| A*TZV | 97 | 704,123711 | 660 | 503085 | 138053 | 8 |
| A*TZVn | 97 | 705,525773 | 660 | 638792 | 141913 | 8 |
| ChE | 97 | 1201,78351 | 753 | 26107 | 4717 | 4 |
| ChTZV | 97 | 1371,14433 | 864 | 44759 | 5722 | 5 |

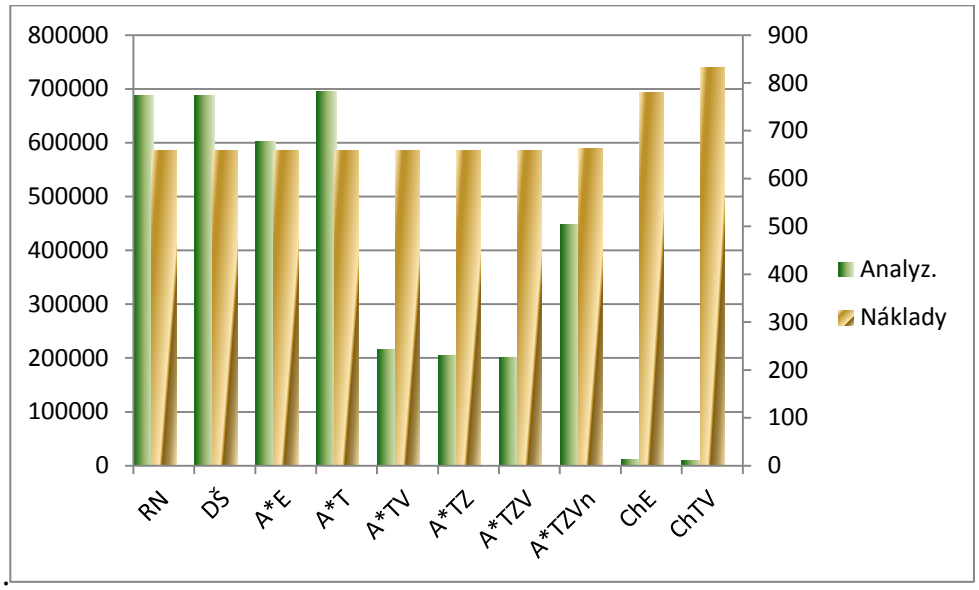
### *3.3.2.3 Scenár: Konštantný, 20, 3*

Konštantný model pohybu ignoruje terén a aj výšku dlaždíc mapy. Pohyb je tak obmedzovaný len nepriechodnými prekážkami. Z toho dôvodu našli trasu minimálnych nákladov takmer všetky algoritmy, a to aj Prehľadávanie do hĺbky.

**Tab. 21 Výsledky algoritmov v scenári Konštantný, 20, 3**

| Algoritm. | Úspešné | Náklady | Dĺžka | Analyz. | V mape | Max trás |
|---|---|---|---|---|---|---|
| RN | 100 | 659,32 | 659 | 686856 | 181939 | 5 |
| DŠ | 100 | 659,32 | 659 | 686856 | 181939 | 5 |
| A*E | 100 | 659,32 | 659 | 602872 | 161004 | 5 |
| A*T | 100 | 659,32 | 659 | 695805 | 181973 | 5 |
| A*TV | 100 | 659,32 | 659 | 215047 | 56313 | 3 |
| A*TZ | 100 | 659,32 | 659 | 203561 | 56601 | 3 |
| A*TZV | 100 | 659,32 | 659 | 201592 | 56049 | 3 |
| A*TZVn | 100 | 661,88 | 661 | 448164 | 79392 | 5 |
| ChE | 100 | 779,26 | 779 | 11892 | 3956 | 2 |
| ChTZV | 100 | 831,22 | 831 | 9878 | 3694 | 2 |

Optimálnu trasu nenašli len chamtivé algoritmy a algoritmus A* s neprípustnou heuristickou funkciou. Prípustné heuristiky využívajúce taxikársku vzdialenosť, výšku a/alebo zdroje však dokázali nájsť trasu s podstatne nižšími nákladmi na výpočtové pamäťové zdroje




**Obr. 49 Porovnanie výkonu algoritmov pre scenár Konštantný, 20, 3**

## 3.4 Vyhodnotenie výsledkov

### *3.4.1 Efektívnosť algoritmov*

Algoritmy prehľadávania priestoru v tejto kapitole porovnáme z hľadiska ich efektívnosti pri nájdení trasy s najmenšími nákladmi. V tabuľke (Tab. 22) sú označené písmenom O algoritmy, ktoré v danom scenári našli minimálnu trasu. Písmenom A je potom označený ten z nich, ktorý má najmenší počet analyzovaných trás.

V hornom stĺpci tabuľky je model pohybu a pod ním spresnenie scenára vo forme: „kapacita zdroja; maximálna výška“.

**Tab. 22 Porovnanie algoritmov z hľadiska ich úspešnosti**

| Algoritmus | Jednoduchý model pohybu | | | | | | Nezávislý model p. | | | Kon. |
|---|---|---|---|---|---|---|---|---|---|---|
| | 15;10 | 20,3 | 20;10 | 30;10 | 20;33 | ∞;10 | 20;3 | 20;10 | 20;33 | 20;3 |
| RN | | O | O | O | O | O | O | O | O | O |
| DŠ | | | | | | | | | | O |
| A*E | | O | O | O | O | O | O | O | O | O |
| A*T | | O | O | | O | | O | | | O |
| A*TV | | O | O | A | A | A | A | O | O | O |
| A*TZ | | O | O | | | O | | O | O | O |
| A*TZV | | A | A | O | | O | | A | A | A |
| A*TZVn | | | | | | | | | | |
| ChE | | | | | | | | | | |
| ChTZV | | | | | | | | | | |

Algoritmus A* dokázal svoje kvality a v sledovaných experimentoch našiel najkratšiu trasu a to s menšími nákladmi ako ostatné algoritmy. V rôznych situáciách však boli použité dve mierne odlišné heuristické funkcie. V oboch prípadoch išlo o taxikársku vzdialenosť k cieľu a výškový rozdiel. Pri použití nezávislého modelu pohybu sa najlepšie osvedčila heuristika, ktorá k týmto dvoch veličinách pridala aj spotrebované množstvo zdroja v spracovávanom stave.

### *3.4.2 Vzťah celkových nákladov k typu pohybu*

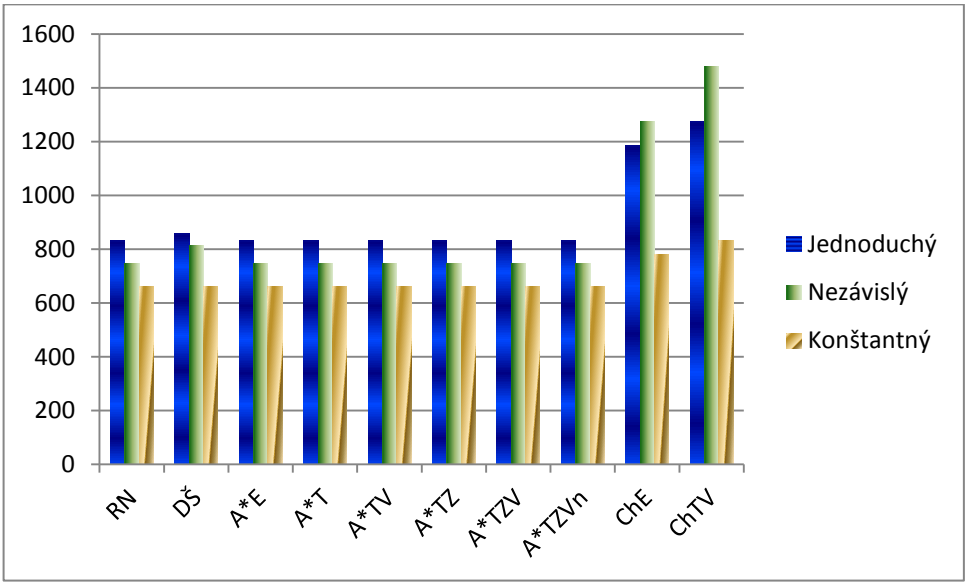
Výsledky v tabuľke Tab. 23 sú pre mapu 450x221 s 11 úrovňami výšky a kapacitou 20. Pri konštantnom pohybe sa neberie do úvahy terén (okrem nepriechodného) a ani nadmorská výška. Nájdená trasa je tak vždy najkratšia. Pri nezávislom pohybe sa zdroj spotrebováva len pri prechode na vyššiu úroveň, takže na

veľkej mape s 11 úrovňami výšky má lepšie výsledky ako jednoduchý pohyb. V taxikárskej geometrii trasa môže často krát obísť vyšší terén bez toho, že by to ovplyvnilo celkové náklady.

**Tab. 23 Porovnanie nákladov na trasu pre rôzne typy pohybu po mape**

| Algoritmus | | Jednoduchý | Nezávislý | Konšt. |
|---|---|---|---|---|
| RN | Rovnomerné náklady | 831,204545 | 746,103093 | 659,32 |
| DŠ | Najskôr do šírky | 857,545455 | 812,762887 | 659,32 |
| A*E | A* Euklidovská vzdialenosť | 831,204545 | 746,103093 | 659,32 |
| A*T | A* Taxikárska vzdialenosť | 831,204545 | 746,134021 | 659,32 |
| A*TV | A* Taxik. vzdial. a výška | 831,204545 | 746,103093 | 659,32 |
| A*TZ | A* Taxik. vzdial. a zdroje | 831,204545 | 746,103093 | 659,32 |
| A*TZV | A* Taxi. vzd., zdroje a výška | 831,204545 | 746,103093 | 659,32 |
| A*TZVn | A* Taxi. vzd. a zdroje neškál. | 832,000000 | 747,896907 | 661,88 |
| ChE | Chamtivé Euklidovská vzdial. | 1185,545460 | 1275,41237 | 779,26 |
| ChTZV | Chamtivé Taxi.v., zdr. a výška | 1275,159090 | 1480,69072 | 831,22 |

V grafe je viditeľná výhodnosť jednotlivých algoritmov a heuristík pre jednotlivé modely pohybu po mape (Obr. 50 Vzťah medzi celkovými nákladmi nájdenej trasy a modelom pohybu). Chamtivé algoritmy, ktoré vždy dosahujú z pohľadu kvality nájdenej trasy najslabšie výsledky, majú horšie výsledky v pomere k ostatným algoritmom pri nezávislom a jednoduchom pohybe. Pri konštantnom pohybe sú celkové náklady nájdenej trasy bližšie k minimu nájdenému ostatnými algoritmami.




**Obr. 50 Vzťah medzi celkovými nákladmi nájdenej trasy a modelom pohybu**

### 3.4.3 *Vzťah dĺžky trasy k počtu výšok na mape*

Všetky testované algoritmy s rastom počtu výškových úrovní nachádzali menej nákladné trasy. Tento jav môže súvisieť aj s menšími rozmermi mapy, keďže s pre väčšie maximálne výšky sa vytvoria menšie mapy.

**Tab. 24 Vzťah medzi počtom výšok a celkovými nákladmi**

| Algoritmus | | 34 výšok | 11 výšok | 3 výšky |
|---|---|---|---|---|
| RN | Rovnomerná cena | 779,444 | 821,083333 | 844,186782 |
| DŠ | Najskôr do šírky | 804,858 | | 872,597701 |
| A*E | A*, euklidovská vzdial. | 779,444 | 821,083333 | 844,186782 |
| A*T | A*, taxikárska vzdial. | 779,450 | 821,083333 | 844,186782 |
| A*TV | A*, taxi.vzdial.a výška | 779,443 | 821,083333 | 844,186782 |
| A*TZ | A*, taxi.vzdial.a zdroje | 779,446 | 821,083333 | 844,186782 |
| A*TZV | A*, taxi.vzd., zdr. a výš. | 779,444 | 821,083333 | 844,186782 |
| A*TZVn | A*, taxi.vz., zdroje 2 | 780,242 | 821,919643 | 845,652299 |
| ChE | Chamtivé, euklid. vzd. | 1094,720 | 1169,059524 | 1188,442530 |
| ChTZV | Chamtivé, taxi.v., zdr. a výš. | 1186,210 | 1268,886905 | 1262,097700 |

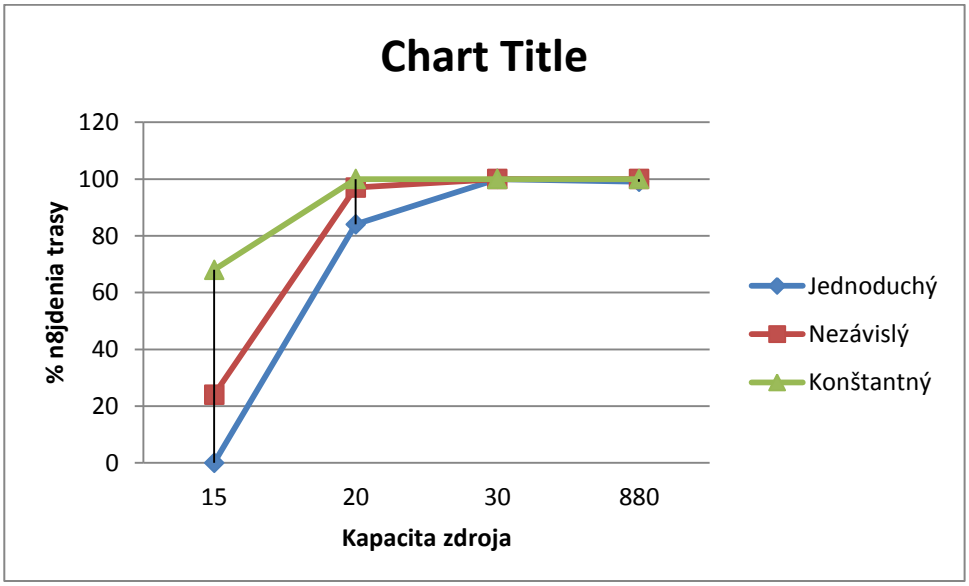
### 3.4.4 *Vzťah modelu pohybu a kapacita zdroja*

Podmienka na zdroje výrazne ovplyvňuje vlastnosti trasy. Ako sa však zvyšuje kapacita, tak klesá potreba neefektívnych odbočiek na doplnenie zdroja. V tabuľke sú uvedené výsledky 100 behov generovania trasy algoritmom A*, s heuristikou založenou na taxikárskej vzdialenosti a výške, pre jednoduchý model pohybu s 11 výškovými úrovňami v mape s rozmermi 450x221.

**Tab. 25 Vzťah modelu pohybu a kapacity zdroja**

| Pohyb | Kap. | Úspešnosť (%) | Náklady | Analyz. |
|---|---|---|---|---|
| Konštantný | 15 | 68 | 747,352941 | 359564 |
| Jednoduchý | 15 | 0 | | |
| Nezávislý | 15 | 24 | 1120,958333 | 493141 |
| Konštantný | 20 | 100 | 655,040000 | 213022 |
| Jednoduchý | 20 | 84 | 821,083333 | 495375 |
| Nezávislý | 20 | 97 | 746,103093 | 606189 |
| Konštantný | 30 | 100 | 651,070000 | 186461 |
| Jednoduchý | 30 | 100 | 711,537500 | 555338 |
| Nezávislý | 30 | 100 | 678,540000 | 534334 |
| Konštantný | 880 | 100 | 651,000000 | 227273 |
| Jednoduchý | 880 | 99 | 678,323232 | 595509 |
| Nezávislý | 880 | 100 | 653,890000 | 246123 |

Najcitlivejší na kapacitu je jednoduchý model pohybu, kde je spotreba najväčšia. Konštantný model pohybu je dokáže nájsť trasu v 2/3 máp aj s kapacitou 15 zdrojov. Podľa predpokladov s rastom kapacity rastie aj percento úspešnosti generovania.




**Obr. 51. Úspešnosť nájdenia trasy v závislosti od modelu pohybu a kapacity zdroja**

# Záver

V práci sme ukázali ako sa používajú algoritmy prehľadávania stavového priestoru na hľadanie trasy v mape a ako ich je potrebné upraviť, aby pracovali aj pri obmedzených zdrojoch k pohybu. Algoritmy sme podrobili skúmaniu za rôznych nastaveniach modelu.

Výsledky porovnania potvrdili, že pre hľadanie trasy na mape je veľmi dôležité definovať vhodnú heuristiku. Neinformované prehľadávanie do šírky má exponenciálnu náročnosť ako na zdroje tak i na pamäť a pre väčšie mapy je prakticky nepoužiteľné.

Vhodnosť heuristiky je daná vlastnosťami mapy. Najdôležitejšou zložkou je vzdialenosť skúmaného bodu, resp. dlaždice k cieľu tak, ako aj v pri modeli mapy bez obmedzenia zdrojov. Z výsledkov vyplýva, že pri štvorcovej mape s pohybom iba cez hrany je výhodnejšie použiť manhattenskú vzdialenosť oproti euklidovskej, keďže výsledné trasy sú rovnako nákladné, ale taxikárska vzdialenosť sa podstatne jednoduchšie počíta a menší je aj počet analyzovaných trás.

V testovaných heuristických funkciách bola aj funkcia nespĺňajúca podmienku prípustnosti a výsledky len potvrdili teóriu, keďže daná heuristika nachádzala trasy, ktoré mali horšie celkové náklady ako algoritmus A* s prípustnou heuristickou funkciou alebo ako Prehľadávanie s rovnomernými nákladmi.

Pokiaľ je pri riešení problému podstatná rýchlosť nájdenia trasy a prípustné sú aj neoptimálne trasy (aspoň do určitej miery), tak je výhodné použiť Chamtivý algoritmus s vhodnou heuristickou funkciou. Oproti algoritmu A* má Chamtivé hľadanie o jeden až dva rády menšie nároky na výkon a pamäť.

Pre mapy, kde má výška výrazný vplyv na náklady a spotrebu pri pohybe medzi dlaždicami, je výhodné rozšíriť heuristiku o funkciu výšky, resp. rozdielu výšky skúmaného a cieľového stavu.

Množina heuristík a algoritmov testovaná súčasnou verziou programu je pomerne malá a bolo by zaujímavé doplniť do porovnania viacero ďalších dôležitých algoritmov ako paprskové či obojsmerné hľadanie. Tiež je určitým zjednodušením porovnávanie len na štvorcových mapách s prechodmi iba cez hrany. Dizajn programu je dostatočne flexibilný, aby sa bez väčších problémov dal rozšíriť ako o nové algoritmy, tak i o nové typy máp, 8-smerovú a/alebo šesťuholníkovú.

# Zoznam použitej literatúry


ADAM, G. 2011. *Introduction to Artificial Intelligence. State Space Search* [online]. Prezentácia. [cit. 2012-05-30] Dostupné na internete: <http://www.atoutfox.org/modules/articles/pdf/0000000788.pdf>.

ALECHINA, N. – LOGAN, B. 1998. State space search with prioritised soft constraints. In *Proceedings of the {ECAI-98} Workshop. Decision theory meets artificial intelligence: qualitative and quantitative approach* [online]. Nottingham : University of Nottingham, 1998. s. 33 - 42, [cit. 2012-05-14] Dostupné na internete: < www.cs.nott.ac.uk/~nza/papers/Alechina+Logan:99a.pdf>

AMIT P. 2003. *Amit's thoughts on path-finding and A-star*. [online]. [cit. 2012-04-25] Dostupné na internete: <http://theory.stanford.edu/~amitp/GameProgramming/>.

BAŠIĆ, B. D. – ŠNAJDER, J. 2012. *Artificial Intelligence. State Space Search* [online]. Prednášky. Zahreb : University of Zagreb Faculty of Electrical Engineering and Computing (FER). [cit. 2012-05-11]. Dostupné na internete: www.fer.unizg.hr/_download/repository/AI-2-StateSpaceSearch.pdf.> 51 s.

BAŠIĆ, B. D. – ŠNAJDER, J. 2012. *Artificial Intelligence. Heuristic Search* [online]. Prednášky. Zahreb : University of Zagreb Faculty of Electrical Engineering and Computing (FER). [cit. 2012-05-11]. Dostupné na internete: www.fer.unizg.hr/_download/repository/AI-3-HeuristicSearch.pdf.> 32 s.

BEEKER, E. R. 2004. An Algorithm for Finding the Shortest Sailing Distance from Any Maritime Navigable Point to a Designated Port. In *Mathematical and Computer Modelling* [online]. 2004, vol. 39, no. 6-8, p. 641 - 647 [cit. 2012-05-05]. Dostupné na internete: <http://www.sciencedirect.com/science/article/pii/S0895717704905456>. ISSN 0895-7177.

BJŐRNSSON, Y. - HALLDÓRSON, K. 2006. Improved Heuristics for Optimal Pathfinding on Game Maps [online]. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. Reykjavik : Reykjavik University, 2006 [cit. 2012-05-18]. Dostupné na internete: <www.ru.is/faculty/yngvi/pdf/BjornssonH06.pdf>. s. 9 – 14.

BJŐRNSSON, Y. et al. 2003. Comparison of Different Grid Abstractions for Pathfinding on Maps. In *IJCAI'03 Proceedings of the 18th international joint conference on Artificial intelligence* [online]. San Francisco : Morgan Kaufmann Publishers, 2003 [cit. 2012-04-29]. Dostupné na internete: <webdocs.cs.ualberta.ca/~emarkus/publications/ijcai2003.ps>. p. 1511-1512.

BJŐRNSSON, Y. et al. 2005. Fringe Search: Beating A* at Pathfinding on Game Maps. In *Proceedings of the 2005 IEEE Symposium on Computational Intelligence and Games (CIG05)* [online]. Essex : Essex University, 2005 [cit. 2012-05-17]. Dostupné na internete: <http://www.cs.ualberta.ca/~games/pathfind/publications/cig2005.pdf>. p. 8.

BOTEA, A. - MÜLLER, M. – SCHAEFFER, J. 2004. Near Optimal Hierarchical Path-Finding. In *Journal of Game Development* [online]. 2004, vol. 1, issue 1 [cit. 2012-05-30]. Dostupné na internete: <http://abotea.rsise.anu.edu.au/data/hpastar.pdf >. ISSN 1543-9399. 30 s.

BREWER, C. A. 200x. http://www.ColorBrewer.org, prvý prístup 4.6.2012, <http://www.personal.psu.edu/cab38/ColorBrewer/ColorBrewer_updates.html>.

BRIMKOV, V. E. – BARNEVA, R. P. 2004. Analytical Honeycomb Geometry for Raster and Volume Graphics [online]. In *Communication and Information Technology Research Technical Report*. 2004, Vol. 151 [cit. 2012-05-30]. Dostupné na internete: <https://researchspace.auckland.ac.nz/handle/2292/2815>. 33 s.

CABALLERO, D. 2006. Taxicab Geometry: some problems and solutions for square grid-based fire spread simulation. In *International Conference on Forest Fire Research* [online]. D. X. Viegas, 2006. [cit. 2012-06-01] Dostupné na internete: <taxicabgeometry.net/docs/mirror/20061127_Caballero_Taxicab.pdf >.

FAYYOUMI, E. Structure and Strategies for State Space Search [online]. Prednášky podľa Luger, G. 2004. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Pearson Education. 2004. [cit. 2012-05-06] Dostupné na internete: <http://www.eis.hu.edu.jo/ACUploads/10749/Chapter3_Structure%20and%20strategies%20for%20state%20space%20search.pdf>.

GOODWIN, S. D. – MENON - S. R. - PRICE, G. 2009. *Pathfinding in Open Terrain* [online]. Windsor : Centre for Interactive Digital Entertainment Research. 2009 [cit. 2012-

05-12]. Dostupné na internete: <http://www.stanford.edu/~smenon/professional_files/publications/pathfinding_in_open_terrain.pdf >. 8 s.

KLAUS, B. – HORN, P. 2002. Saddle Points in 2D. In Arjan Kuijper : *The deep structure of Gaussian scale space images* [online]. Enschede : PrintPartners Ipskamp B.V. 2002, p. 23-28 [cit. 2012-06-02]. Dostupné na internete: <http://igitur-archive.library.uu.nl/dissertations/2002-0904-155155/full.pdf.>. ISBN 90–393–3061–1.

LANCOT, M. et al. 2006. Path-finding for large scale multiplayer computer games [online]. In *Proceedings of the 2nd Annual North American Game-On Conference (GameOn'NA 2006).* [cit. 2012-04-28] Dostupné na internete: <http://gram.cs.mcgill.ca/papers/lanctot-06-path-finding.pdf>. 8 s.

MACKLEM, G. 2003. *Computer Landscape Generation and Smoothing. Masters Comprehensive Exam* [online]. [cit. 2012-05-12] Dostupné na internete: <www.gantless.com/programs/macklem.pdf >. 29 s.

NÁVRAT, P. a kol. 2001. *Umelá Inteligencia*. Bratislava : Vydavateľstvo STU, 2001. 393 s. ISBN 80-227-1645-6.

OLSEN, J. 2004. *Realtime Procedural Terrain Generation. Realtime Synthesis of Eroded Fractal Terrain for Use in Computer Games* [online]. Department of Mathematics And Computer Science (IMADA) University of Southern Denmark. 2004. [cit. 2012-05-17] Dostupné na internete: <http://web.mit.edu/cesium/Public/terrain.pdf>. 20 s.

PORUBSKÝ, Š. 2012. *Integer rounding functions* [online]. Retrieved 2012/6/16 from Interactive Information Portal for Algorithmic Mathematics. Praha : Institute of Computer Science of the Czech Academy of Sciences. [cit. 2012-04-14] Dostupné na internete: http://www.cs.cas.cz/portal/AlgoMath/NumberTheory/ArithmeticFunctions/IntegerRoundingFunctions.htm

SSJ : *A Java library for a stochastic simulation API specification* [online]. Dostupné na internete: <http://www.iro.umontreal.ca/~simardr/ssj/doc/html/overview-summary.html>.

ŠTULLER, J. 2006. *Znalostné systémy. Riešenie úloh a využívanie znalostí* [online]. Prednášky. Praha: ÚI AC ČR, 2006. [cit. 2012-05-02] Dostupné na internete: www.cs.cas.cz/.../3%20Znalostné%20systémy.ppt.

TOZOUR P. 2008. *Fixing Pathfinding Once and For All* [online]. [cit. 2012-05-3] Dostupné na internete: <http://www.ai-blog.net/archives/000152.html>.

WILLIAMS, B. C. 2003. *Problem Solving as State Space Search* (slides adapted from Tomas Lozano Perez, Russell and Norvig AIMA) [online]. [cit. 2012-04-20] Dostupné na internete: http://stuff.mit.edu/afs/athena/course/16/16.410/www/lectures_fall04/l3_uninformed_search.pdf. 87 s.

YANG, F. et al. 2008. A General Theory of Additive State Space Abstractions. In *Journal of Artificial Intelligence Research* [online]. 2008, no.32, p. 7-28 [cit. 2012-05-12]. Dostupné na internete: <www.jair.org/media/2486/live-2486-3915-jair.pdf >. ISSN 1076 - 9757.