Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Tomáš Novella

Grid-Based Path Planning

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Tomáš Balyo

Studijní program: Informatika

Studijní obor: Obecná informatika

Praha 2013

Poděkování.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V .... dne .... Podpis autora

Název práce: Název práce

Autor: Jméno a příjmení autora

Katedra: Název katedry či ústavu, kde byla práce oficiálně zadána

Vedoucí bakalářské práce: Jméno a příjmení s tituly, pracoviště

Abstrakt:

Klíčová slova:

Title:

Author: Jméno a příjmení autora

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Jméno a příjmení s tituly, pracoviště

Abstract:

Keywords:

Obsah

Úvod

0.1 Motivácia

Hľadanie najkratšej cesty je jedným z elementárnych problémov teórie grafov. Dôležitosť tohto problému je zrejmá, najmä keď si uvedomíme jeho všestranné využitie, napríklad v umelej inteligencii, v počítačovych hrách a podobne. Nanešťastie riešenia a algoritmy využívané v komerčných hrách sú closed-source a teda obecne nie známe.

0.2 Stručný popis práce, ciele

Na nájdenie najkratšej cesty v obecných grafoch využívame Dijkstrov algoritmus, prípadne jeho nadstavby, ktoré si následne rozoberieme.

Cieľom práce je hľadanie cesty v špeciálnych grafoch - tzv. herných mapách. Zadanie tohto problému sa od všeobecneho hľadania najkratšej cesty v obecných grafov líši v dvoch veciach. Jednak grafy herných máp predstavujú akúsi mriežku - a teda arita vrcholov je maximálne 8 a jednak pri riešení tejto úlohy máme k dispozícii čas na predspracovanie mapy a vytvorenie pomocných dátových štruktúr.

V práci sme naimplementovali vlastný algoritmus a porovnali ho s doterajšímy známymi. TODO?? Počas práce sme prišli na zaujímavé zefektívnenie algoritmov [snad na nieco prijdem :))]a dúfame v jeho rozšírenie do hernej sféry.

ASK?? ake su vlastne ciele? mam vymysliet vzbrusu novy algoritmus?

V prvej kapitole si zadefinujeme kľúčové termíny a popíšeme problem. Na konci kapitoly spomenieme súťaž, ktorej sa daný algoritmus zúčastnil a popíšeme jej podmienky. Druhá kapitola sa pokúsime rozobrať doterajšie zistenia a algoritmy používané na riešenie obdobných problémov. V tretej kapitole popíšeme náš algoritmus a vo štvrtej kapitole ho porovnáme s ostatnými algoritmami a uvedieme výsledky.

1. Zadanie problému a cieľové požiadavky

1.1 Úvodné definície a značenia

Na začiatok si zaveďme niektoré dôležité pojmy teórie grafov. Budú sa týkať obecnej teórie a úlohu so všetkými jej špecifikami si ozrejmíme v nasledujúcej kapitole.

Definícia. *Graf G je usporiadaná dvojica (V, E), kde V označuje množinu vrcholov(vertices) a* $E \subseteq V \times V$ *označuje množinu hrán (edges). Značíme G = (V, E).*

Definícia. *Ohodnotený graf (G, w) je graf s spolu s reálnou funkciou (tzv. ohodnotením)* $w : E(G) \to \mathbb{R}$*, kde w je funkcia, ktorá každej hrane priradí reálne číslo, takzvanú* cenu*, alebo* dĺžku *hrany.*

Teraz keď už vieme, čo je to graf, skúsme si zadefinovať najkratšiu cestu. Začnime najprv obecne cestou.

Definícia. *Cesta P z vrcholu* v_0 *do vrcholu* v_n *v grafe G je postupnosť* $P = (v_0, e_1, v_1, \ldots, e_n, v_n)$*, pre ktorú platí* $e_i = \{v_{i-i}, v_i\}$ *a taktiež* $v_i \neq v_j$ *pre každé* $i \neq j$*.*

Všimnime si, že v ceste nenavštívime žiaden vrchol dvakrát a teda cesta neobsahuje kružnice.

TODO?? dlzka hrany - cena cesty - zadefinuj funkciu d(u, v)

Definícia. *Cena cesty P z vrcholu* v_0 *do vrcholu* v_n *v ohodnotenom grafe (G, w) je súčet cien hrán, ktoré sa na ceste nachádzajú.*

Definícia. *Najkratšia cesta P z vrcholu* v_0 *do vrcholu* v_n *v ohodnotenom grafe (G, w) je cesta s najnižsou cenou.*

1.2 Herná mapa

Po zavedení kľúčových pojmov sa dostávame k samotnému zadaniu úlohy. Ako sme už spomínali, problém budeme riešit na tzv. herných mapách. Čo je herná mapa a v čom sa od obecného grafu odlišuje?

Zjednodušene povedané, je to mapa s ktorou sa stretávame v najrôznejších hrách, ako je Warcraft, Startcraft, Dragon Age a podobne.

Ide v podstate o špeciálny a dosť obmedzený typ grafu. Vizuálne si ju môžme predstaviť ako graf v ktorom sú vrcholy rozostúpené v tvare mriežky a hrana je stále medzi dvojicami susedných vrcholov vo všetkých ôsmych smeroch. Cena vodorovnej alebo zvislej hrany je 1 a cena šikmej hrany je $\sqrt{2}$.

Skúsme si teraz nadefinovať hernú mapu formálne.

Definícia. *Herná mapa rozmeru m*n je ohodnotený graf v ohodnotením w s m*n vrcholmi očíslovanými od $v_{1,1}$ až po $v_{m,n}$ s jednoduchými hranami j v tvare* $\{v_{a,b}, v_{a,b+1}\}, \{v_{a,b}, v_{a+1,b}\}$*, kde* $w(j) = 1$ *a šikmými hranami s v tvare* $\{v_{a,b}, v_{a+1,b+1}\}, \{v_{a,b}, v_{a+1,b-}$ *kde* $w(s) = 1.4142$.

Poznámka 1. *Herná mapa je v tvare obdĺžniku a teda budem kvôli stručnosti pre hernú mapu používať pojem obdĺžnik.*

Definícia. *Herná podmapa p rozmeru a*b hernej mapy P rozmeru m*n TODO?? definuje ernu podmapu ako suvisly podgraf blablabla ASK?? ako zjednodusit??? neprijemne komplikovana definicia....*

FIXME?? pridat obrazok

1.3 GPPC: Grid-Based Path Planning Competition

Algoritmus navrhnutý a naprogramovaný v tejto práci bol zaradený do súťaže **GPPC**, ktorá sa koná približne 2 krát ročne.

1.3.1 Špecifiká súťaže, limity

Herné mapy budú mať rozmery maximálne 2048 × 2048. Súťaž bude rozdelená do dvoch fáz — fázy predspracovania mapy(pre-processing) a fázy testovania. Na predspracovanie mapy bude vyhradený čas maximálne 30 minút a program si svoje dáta uloží na disk do súboru o veľkosti maximálne 50MB. A potom vo fáze testovania budé dostávať požiadavky na nájdenie najkratšej cesty.

Úlohou súťaže je naimplementovať tieto tri funkcie.

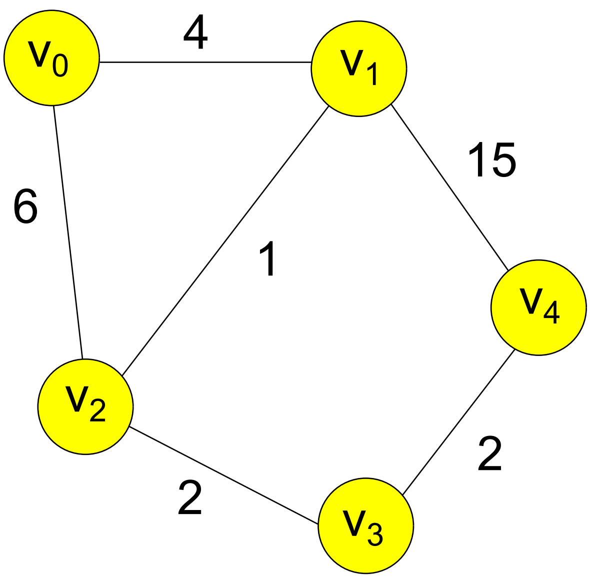
```
void PreprocessMap(std::vector &bits, int width, int height,
                   const char *filename);

void *PrepareForSearch(std::vector &bits, int width, int height,
                       const char *filename);
bool GetPath(void *data, xyLoc s, xyLoc g, std::vector &path);
```

1.3.2 Kritériá súťaže, hodnotenie programov

Programy sa nebudú porovnávať v jednej metrike a podľa jedného kritéria, nakoľko sa programy dalo zoptimalizovať podľa viacerých kritérií. Metriky sú nasledovné:

- Celkový čas na nájdenie cesty.
- Čas na nájdenie prvých 20 tich krokov.
- Dĺžka cesty (zohľadnená suboptimalita).
- Maximálny čas vrátenia hocitktorej časti cesty.

Testovací počítač má 12 GB RAM pamäti a dva 2.4 Ghz Intel Xeon E5620 procesory.


v0
v1
v2
v3
v4
4
6
1
15
2
2

2. Prehľad algoritmov

Na hľadanie najkratších ciest v grafe poznáme mnoho algoritmov, ktoré vieme rozdeliť do troch skupín.

- Point To Point Shortest Path - hľadáme najkratšiu cestu medzi dvoma zadanými bodmi
- Single Source Shortest Path - pre daný vrchol v hľadáme najkratšiu cestu do všetkých vrcholov grafu.
- All Pairs Shortest Path - skúmame najkratšiu cestu medzi všetkými dvojicami vrcholov.

Napriek tomu, že sú tieto problémy na obecných grafoch NP-ťažké, na herných mapách, kde majú všetky vrcholy kladnú cenu, vieme nájsť riešenie v polynomiálnom čase. V práci sa ďalej budeme zaoberať riešením prvého problému (Point to Point Shortest Path).

V tejto kapitole si následne popíšeme algoritmy, ktoré sú použiteľné na všetkých grafoch s nezápornými dĺžkami hrán.

2.1 Dijkstrov algoritmus

Medzi základné algoritmy patrí Dijkstrov algoritmus, ktorý je asymptoticky optimálny (TODO?? for sure?).

Pri hľadaní cesty z vrcholu s do vrcholu t prechádzame postupne vrcholy zo stúpajúcou vzdialenosťou od s, až dokým sa nedostaneme k cieľovému vrcholu t. Graficky si beh algoritmu môžme predstaviť ako kruh so stredom v bode s so zväčšujúcim sa polomerom. Algoritmus napísaný v pseudokóde je nasledovný:

```
d(s) <- 0
d(*) <- $\infty$
```

3. Nový algoritmus: NovellA*

3.1 Zlepšenie výkonu v niektorých prípadoch

Nie všetky cesty sú ale také kľukaté. V mnohých prípadoch, napríklad keď medzi počiatočným a koncovým bodom neleží žiadna prekážka, sú cesty veľmi priamočiare. To sa pokúsime využiť na zlepšenie výkonu algoritmu v niektorých prípadoch. Predstavme si, že máme obdlžníkovú mapu bez prekážoch a hľadáme najkratšiu cestu medzi bodmi $s = (x_s, y_s), t = (x_t, y_t)$. V tomto prípade vieme nájsť najkratšiu cestu veľmi jednoducho. Algoritmus: vstup:
$s = (x_s, y_s), t = (x_t, y_t)$
vystup:
path ... usporiadaná množina súradníc po ktorých vedie cesta
beh algoritmu:

```
path = (xs, ys)

TODO?? nejde diakritika??
# skopirujem suradnice pociatocneho vrcholu
(x1, y1) = (xs, ys)
while (x1, y1) <> (x2, y2):
        if y1 < y2:
                ++y1
        else:
                --y1

        if x1 < x2:
                ++x1
        else:
                --x1
        path.append((x1,y1))
```

Teda, jednoducho povedané: keď sa počiatočný a koncový bod líšia v jednej súradnici, tak sa posúvame priamočiaro, keď sa líšia v oboch, tak sa posúvame šikmo.

Pokiaľ si zadefinujeme $dx := |x_t - x_s|$ a $dy := |y_t - y_s|$, tak počet vrcholov, ktorými cesta prechádza vieme zhora odhadnúť, ako $\max(dx, dy)$. Jej vzdialenosť vieme zistiť v čase $O\big(\max(dx, dy)\big)$ Na zistenie vzdialenosti v každom kroku nám stači konštantná pamäť.

TODO?? Poznamka, ze nepotrebujem na to obdlzniky, mozem robit aj komplikovanejsie utvary, ale by to sa blbo hladalo... ledaze...

Skúsme to teda nejak využiť. Pokiaľ vieme, že počiatočný aj koncový bod ležia v jednom obdĺžniku, tak máme problém vyriešený. Jediným problémom ostalo takéto obdĺžniky nájsť.

ASK?? alebo radsej neutralne napiast, hladanie obdlznikov?

3.2 Hľadáme obdĺžniky

Pre ľahšie vyjadrovanie si zaveďme definíciu *čistej mriežkovej mapy*. Intuitívne ju môžme vnímať ako mriežkovú mapu bez prekážok.

Definícia. *Mriežková podmapa je čistá pokiaľ medzi každými dvoma susednými vrcholmi existuje hrana.*

3.2.1 Proporcie obdĺžnikov

Dôležitou otázkou je, aké proporcie obdĺžnikov hľadáme. Predstavme si tieto dva prípady. Nech sme mapu dekomponovali na dva obdĺžniky, ktorých celkový obsah je 10. V prvom prípade je obsah prvého 9 a druhého 1, v druhom prípade sú obsahy 6 a 4. Chceme maximalizovať pravdepodobnosť toho, že pri voľbe dvoch náhodných bodov budú obe v rovnakom obdĺžniku.

Úlohu vieme zobecniť na klasickú pravdepodobnostno-optimalizačnú úlohu. Majme k ekvivalenčných tried na množine s n prvkami. Ako zvoliť ekvivalenčné triedy, aby pri voľbe dvoch náhodných prvkov bola pravdepodobnosť toho, že oba prvky budú v tej istej ekvivalenčnej triede čo najvyššia? (Poznámka: ekvivalenčnú triedu predstavuje obdĺžnik a množinu predstavuje množina vrcholov grafu.) Alternatívny pohľad poskytuje pozerať sa na úlohu, ako na farbenie guličiek čo najmenej farbami.

Zapíšme túto úlohu formálne. Množinu prvkov nazvime Prv, označme ekvivalenčné triedy $ek_1 \dots ek_k$, množinu týchto ekvivalenčných tried označme ako Ek, veľkosť triedy ek_i označme k_i a zaveďme si funkciu $f\colon Prv \to Ek$ ktorá roztriedi prvky do ekvivalenčných tried.

Označme výberový priestor $\Omega = \{(x_a, x_b) | x_a, x_b \in Prv, a \neq b\}$ Udalosťou A_i nazveme jav, v ktorom oba prvky patria do tej istej ekvivalenčnej triedy i, teda $A_i = \{(x_a, x_b) | x_a, x_b \in Prv, a \neq b, f(x_a) = f(x_b) = i\}$ Jav $A = \bigcup_{i \in Ek} A_i$ teda nastáva práve vtedy, keď oba vybrané prvky patria do rovnakej triedy.

Úlohou je teda rozvrhnúť vytvoriť funkciu f tak, aby pravdepodobnosť $P[A]$ bola čo najvyššia. Keďže udalosti A_i sú nezlučiteľné, môžme písať $P[A] = P[\bigcup_{i \in Ek} A_i] = \sum_{i \in Ek} P[A_i]$.

Ak si pravdepodobnosť každého javu rozpíšeme, dostaneme $\sum_{i \in Ek} P[A_i] = \sum_{i=1}^{|Ek|} \frac{\binom{k_i}{2}}{\binom{|Prv|}{2}}$.

nakoľko chceme nejak rozvrhnúť prvky v triedach ek_i, a menovateľ je len konštanta, môžme ho vynechať.

Maximalizujeme teda hodnotu výrazu $\sum_{i=1}^{|Ek|} \binom{k_i}{2} = \sum_{i=1}^{|Ek|} \frac{k_i!}{(k_i-2)!2!} = \sum_{i=1}^{|Ek|} \frac{k_i(k_i-1)}{2}$. Po vyškrtnutí konštanty a roznásobení sme dostali nasledujúcu optimalizačnú úlohu: maximalizovať $\sum_{i=1}^{|Ek|} k_i^2 - k_i$ za podmienok $\sum_{k=1}^{|Ek|} k_i = n$, kde $k_i \in \mathbb{N}_0$.

Sumu si vieme rozpísať ako $\sum_{i=1}^{|Ek|} k_i^2 - k_i = \sum_{i=1}^{|Ek|} k_i^2 + \sum_{i=1}^{|Ek|} -k_i$ druhá suma sa nasčíta $-n$, čo je konštanta, takže nám stačí maximalizovať $\sum_{i=1}^{|Ek|} k_i^2$.

Teraz nám už len zostáva použiť nerovnosť $(a+b)^2 \geq a^2 + b^2$, z ktorej jasne vyplýva, že potrebujeme spraviť ľubovoľné k_i čo najväčšie. Ekvivalenčné triedy musia teda byť čo najväčšie a problém sa redukuje na problém hľadania obdĺžnikov s najväčším možným obsahom. V programe tento problém rieši trieda Colorizator.

3.2.2 Nájdenie najväčšieho obdĺžnika

Alebo ináč - nájdenie najväčšej čistej podmapy je problém, ktorý vieme algoritmicky vyriešiť v lineárnom čase v závislosti od počtu vrcholov. TODO?? mozno prepisem na problem najdenia najvacsej podmatice. + hovorit o tom ako o farbeni mriezky

TODO??popis algoritmus

4. Porovnanie algoritmu NovellA* s ostatnými algoritmami.

4.1 Vstupné dáta

Častým problémom pri vzájomnom porovnávaní algoritmov je nájsť tetsovaciu vyorku, ktor...

Na porovnávanie využijeme benchmark

Obrázek 4.1: Logo MFF UK

4.1.1 Ukázka LaTeXu

V této krátké části ukážeme použití několika základních konstrukcí LaTeXu, které by se vám mohly při psaní práce hodit.

Třeba odrážky:

- Logo Matfyzu vidíme na obrázku. 4.1
- Tato subsekce má číslo 4.1.1.
- Odkaz na literaturu [**?**].

Druhy pomlček: červeno-černý (krátká), strana 16–22 (střední), 45 − 44 (minus), a toto je — jak se asi dalo čekat — vložená věta ohraničená dlouhými pomlčkami. (Všimněte si, že jsme za `a` napsali vlnovku místo mezery: to aby se tam nemohl rozdělit řádek.)

„České uvozovky.“

Definícia. *Strom je souvislý graf bez kružnic.*

Veta 1. *Tato věta neplatí.*

Důkaz. Neplatné věty nemají důkaz. □

Závěr

Seznam použité literatury

Seznam tabulek

Seznam použitých zkratek

Přílohy