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

Slávna Eulerova úloha siedmych mostov v Kaliningrade [1] sa považuje za prvú prácu, ktorá zaviedla teóriu grafov. Úlohou je prejsť po týchto siedmych mostoch tak, aby sme po každom šlo práve raz. Od tej doby sa využitie teórie grafov značne rozšírilo a v dnešnej dobe patrí medzi významné a rozpracované teórie. V modernej dobe je jedným z jej najdôležitejších problémov hľadanie najkratšej cesty. Najčastejšie sa s nimi stretávame pri GPS navigacii. Medzi najvýznamnešie práce považujeme práce od Dijkstru [2] a Floyd-Warshalla.

S narastajúcim fenoménom počítačových hier a umelej inteligencie sa do povedomia dostal špeciálny typ grafu – mriežkový graf, využívaný ako herná mapa. V hrách trebalo často nájsť cestu pre počítačom ovládanú postavičku z miesta A do miesta B. Nakoľko je ale väčšina hier komerčná, algoritmy využívané v hrách boli a sú taktiež komerčne. Dôsledkom toho nie sú publikované a porovnané rôzne prístupy a algoritmy na vyhľadávanie v mriežkových mapách. A keď už aj sú, tak práce používajú rôzne mapy na bechmarking a teda neexistuje žiadna globálna porovnávacia štúdia týchto prístupov. Súťaž *Grid-Based Path Planning Competition* [4] sa snaží tento problém vyriešiť tým, že ktorá porovnáva rôzne algoritmy na veľkej množine máp použitých v známych počítačových hrách.

Cieľom tejto práce je spraviť prehľad doterajších prístupov k tomuto problému a prispieť vlastným algoritmom do súťaže.

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

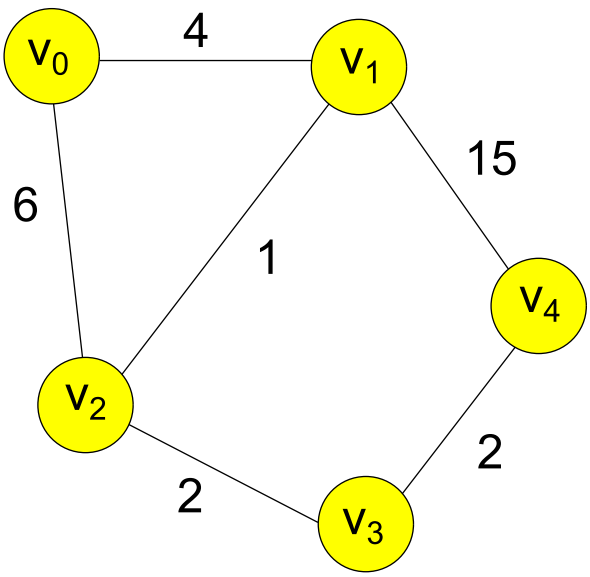
1.1 Úvodné definície a značenia

Na začiatok si zaveďme niektoré dôležité pojmy z 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.*

ukazka na 1.1



Obr. 1.1: Ohodnotený graf

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.*

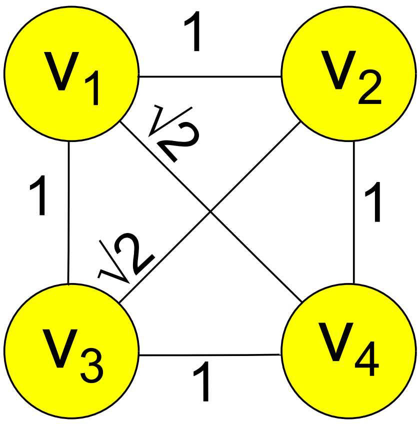
Definícia. *Najkratšia cesta P z vrcholu v_0 do vrcholu v_n v ohodnotenom grafe (G, w) je cesta s najnižšou cenou.*

1.2 Mriežkový graf

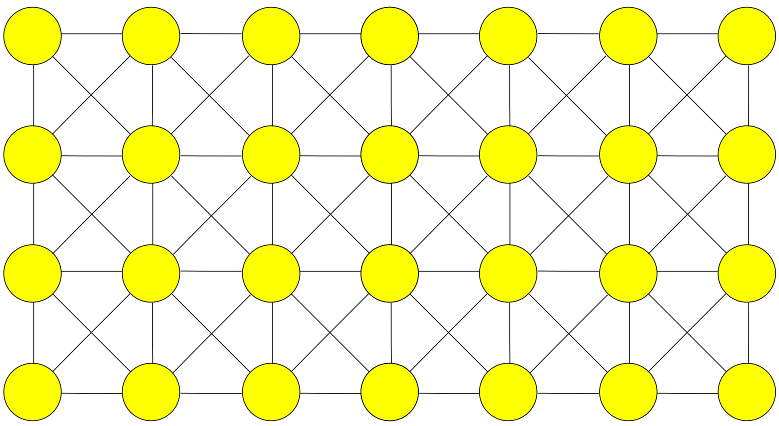
Po zavedení kľúčových pojmov sa dostávame k samotnému zadaniu úlohy. Ako sme už spomínali, problém budeme riešit na tzv. mriežkových grafoch. Čo je mriežkový graf a v čom sa od obecného grafu odlišuje?

Zjednodušene povedané, je to mapa s ktorou sa stretávame v najrôznejších hrách, ako je Warcraft, Startcraft, Dragon Age a podobne. ASK?? je to "mapa"dat do uvodzoviek

Ide o špeciálny a dosť obmedzený typ grafu. Vizuálne si ho môžme predstaviť ako konečný 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 $\sqrt{2}$.



Obr. 1.2: Mriežkový graf 2x2



Obr. 1.3: Mriežkový graf bez označenia vrcholov a dĺžok hrán

Zadefinujme si teraz mriežkový graf formálne.

Definícia. *Mriežkový graf rozmerov $m*n$ 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* $\{v_{a,b}, v_{a+1,b+1}\}$, $\{v_{a,b}, v_{a+1,b+1}\}$, *kde* $w(s) = \sqrt{2}$.

FIXME?? pridat obrazky

Poznámka 1. *Mriežkový graf sa dá reprezentovať ako matica m*n nad telesom* $\mathbb{Z}_2$, *kde jednotky predstavujú vrcholy. Príklad maticovej reprezentácie grafu rozmerov 4x5 je na obrázku 1.4.*

Obr. 1.4: Maticová reprezentácia mriežkového grafu

$$G = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

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 hocictorej časti cesty.

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

2. Prehľad algoritmov

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

- Point To Point Shortest Path(P2PSP) - hľadajú najkratšiu cestu medzi dvoma zadanými bodmi
- Single Source Shortest Path(SSSP) - pre daný vrchol v hľadajú najkratšiu cestu do všetkých vrcholov grafu.
- All Pairs Shortest Path (APSP)- skúmajú najkratšiu cestu medzi všetkými dvojicami vrcholov.

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

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

2.1 Dijkstrov algoritmus

Medzi základné algoritmy typu SSSP patrí Dijkstrov algoritmus 1 popísaný už v roku 1959. Patrí medzi relaxačné algoritmy a jeho činnosť si vieme predstaviť ako posúvanie vlny po grafe TODO??WTf did I just write?

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. Vizuálne si beh algoritmu môžme predstaviť ako kruh so stredom v bode s so zväčšujúcim sa polomerom.

Algoritmus je nasledovný 1. ?? rozlisujeme tri stavy vrcholov, co je halda blablabla ASK??ako odkazovat?

FIXME ?? kolize H a h()

Veta 1. *V dijkstrovom algoritme uzatvárame každý dosiahnuteľný vrchol práve raz.*

ASK?? citovat dijkstru a maresa ako dokaz?

2.1.1 Zložitosť

popis len obecne pomocou casu na ExtractMin a Insert.

2.2 Lineárny Dijkstrov algoritmus

Zložitosť vyššiepopísaného algoritmu 1 závisí od haldy, ktorú použijeme. Pri použití binárnej haldy bude $O(blablabla)$.TODO??

Nakoľko pre mriežkový graf platia ....??

Algoritmus 1 Dijkstra: nájdi najkratšiu cestu medzi dvoma bodmi s a t

Vstup: graf G

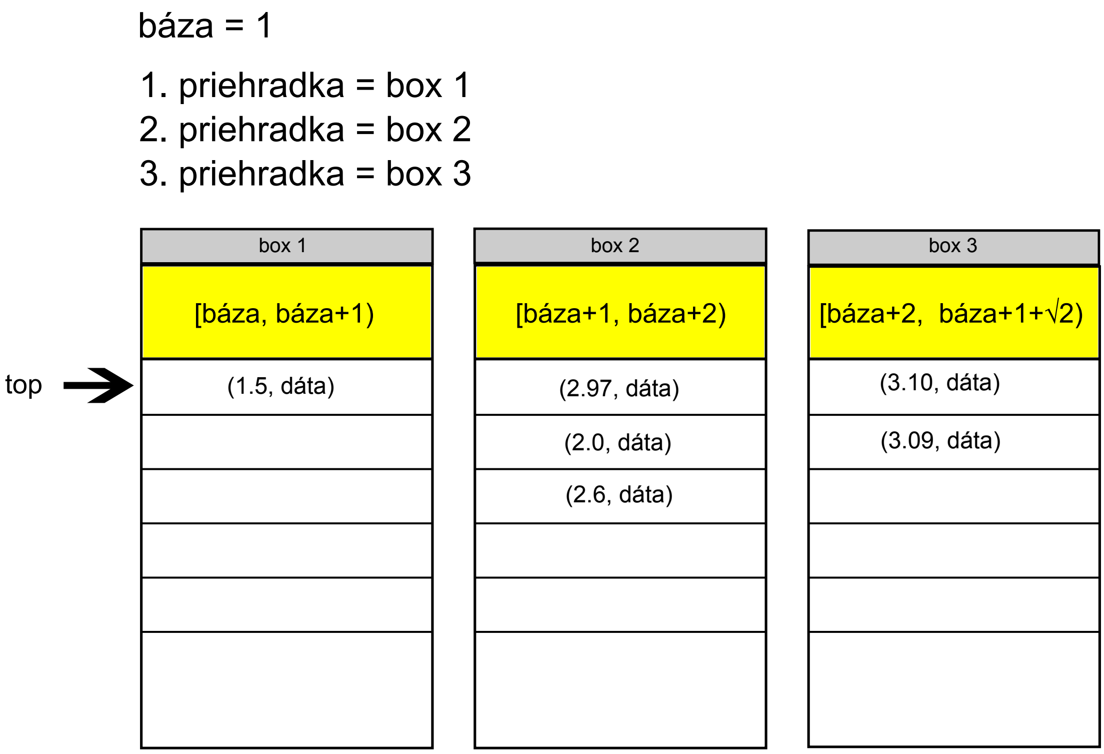
Výstup: hodnotiaca funkcia h obsahujúca najkratšie cesty z vrcholu s do vrcholov grafu

```
h(*) ← ∞
stav(*) ← NENAVŠTÍVENÝ
{pridám počiatok}
h(s) ← 0
stav(s) ← OTVORENÝ
Heap H
Insert(H, s)
while H not empty do
   // vyberiem v – najbližší otvorený vrchol
   v ← ExtractMin(H)
   while stav(v) ≠ OTVORENÝ do
      v ← ExtractMin(H)
   end while
   // zrelaxujeme vrchol v
   for all e, e = (v, u) do
      if h(u)>h(v) + l(v, u) then
         Insert(H, v)
         stav(u) ← OTVORENÝ
         h(u) ← h(v) + l(v, u)
      end if
   end for
end while
```

Veta 2. *Pokiaľ sme v Dijkstrovom algoritme uzavreli vrchol u so vzdialenosťou d_u a najkratšia hrana v grafe má dĺžku ϵ, tak môžme taktiež uzavrieť všetky vrcholy v so vzdialenosťami $d_v \in (d, d + \epsilon)$.*

Dôkaz. Do haldy vieme pridávať len vrcholy so vzdialenosťami aspoň $d+\epsilon$ (kratšia hrana tam už nie je), ale tie už cestu k vrcholom so vzdialenosťami $d_v \in (d, d+\epsilon)$ skrátiť nemôžu. □

Keďže veľkosť najkratšej hrany $\epsilon = 1$, tak v štruktúre budeme mať priehradky, ktorých rozsah je ostro menší ako ϵ. Na konštrukciu lineárneho Dijkstrovho algoritmu potrebujeme priehradkovú haldu. Tá vie vybrať „jeden z najmenších" prvkov. v konštantnom čase a rovnako ako aj vložiť prvok.

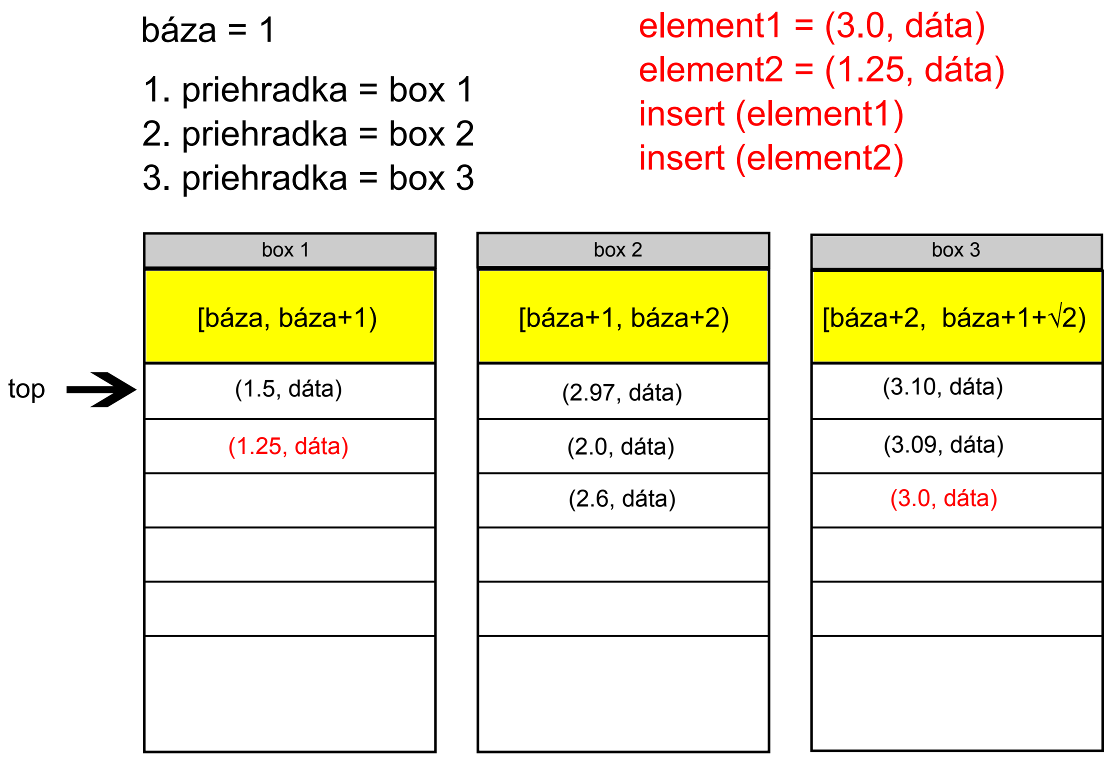


Obr. 2.1: Priehradky

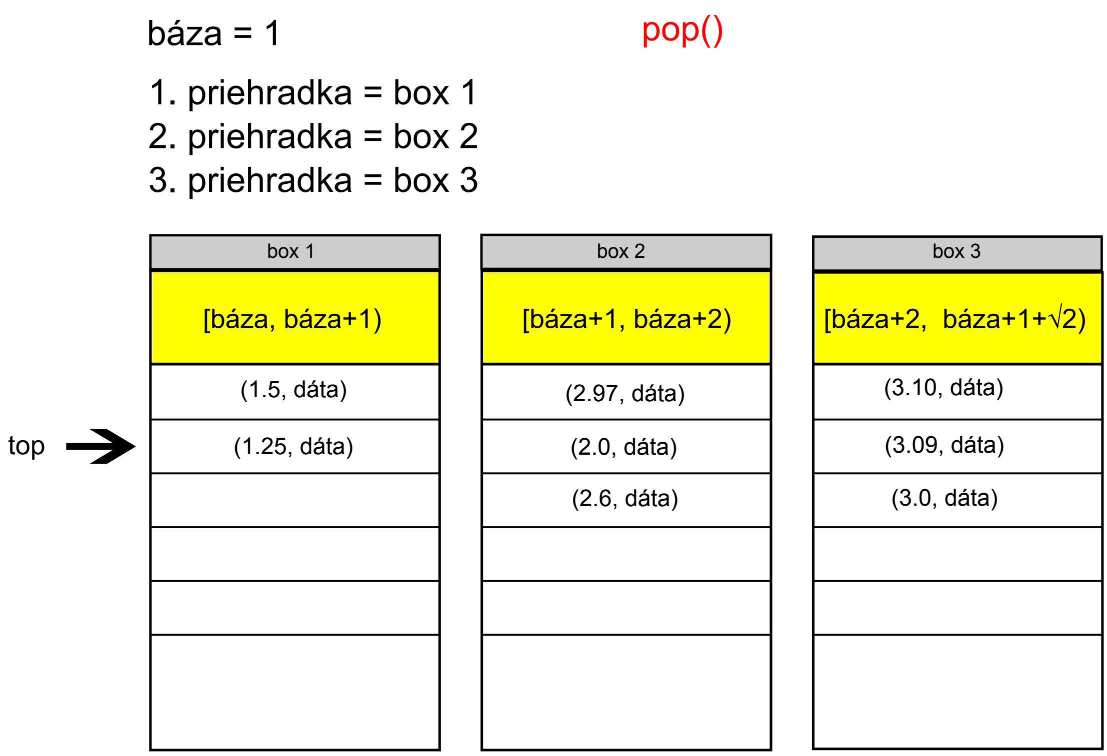
blablabla

2.3 A*

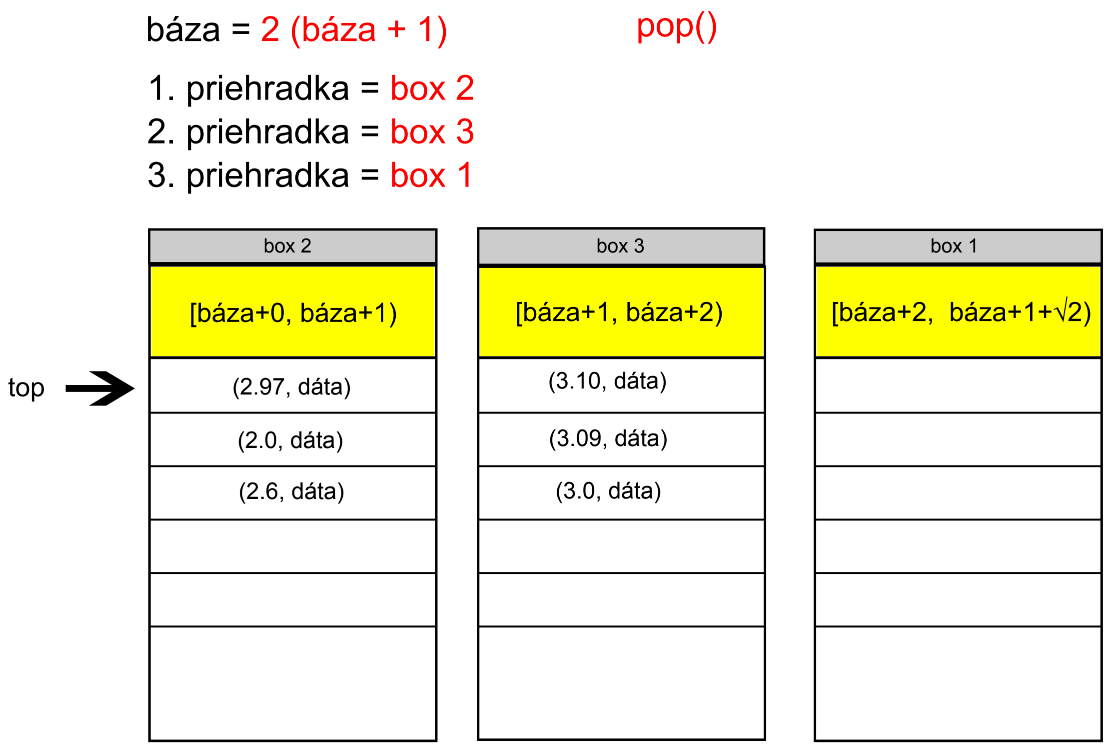
dalsi algoritmus do zbierky je a* [3].



Obr. 2.2: Insert



Obr. 2.3: Insert



Obr. 2.4: Insert

3. Nový algoritmus: NovellA*

3.1 Zlepšenie výkonu v niektorých prípadoch

Nie všetky najkratšie cesty ale musia obchádzať veľa prekážok. V mnohých prípadoch nemusí medzi počiatočným a koncovým bodom ležať žiadna prekážka, a teda cesty sú veľmi priamočiare. To sa pokúsime využiť na zlepšenie výkonu algoritmu.

Pre ľahšie vyjadrovanie si zaveďme definíciu *mriežkového grafu bez prekážok*.

Definícia. *Mriežkový graf je bez prekážok pokiaľ medzi každými dvoma susednými vrcholmi existuje hrana.*

Kvôli lepšej prehľadnosti budeme mriežkový graf bez prekážok nazývať aj *obdĺžnik*. Intuitívne, kvôli jeho vizuálnej interpretácii.

Majme mriežkový graf bez prekážok a hľadajme najkratšiu cestu medzi bodmi $s = (x_s, y_s), t = (x_t, y_t)$. V tomto prípade vieme nájsť najkratšiu cestu veľmi jednoducho.

Algoritmus 2 Nájdi najkratšiu cestu medzi dvoma bodmi s a t na mriežkovom grafe bez prekážok

---

Vstup: $s = (x_s, y_s), t = (x_t, y_t)$
Výstup: *path*

path.append((x_s, y_s)) {pridám počiatok}
while $x_s \neq x_t \vee y_s \neq y_t$ **do**
    if $x_s < x_t$ **then**
        $x_s \leftarrow x_s + 1$
    else if $x_s > x_t$ **then**
        $x_s \leftarrow x_s - 1$
    end if
    if $y_s < y_t$ **then**
        $y_s \leftarrow y_s + 1$
    else if $y_s > y_t$ **then**
        $y_s \leftarrow y_s - 1$
    end if
    path.append((x_s, y_s))
end while

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.

3.2 Hľadáme obdĺžniky

3.2.1 Proporcie obdĺžnikov

Dôležitou otázkou je, na akých vlastnostiach obdĺžnikov záleží. Uvažujme nasledujúci motivačný príklad.

Príklad 1. *Majme na mape nájdené dva obdĺžniky, ktorých celkový obsah je 10. Predstavme si tieto dva prípady. V prvom prípade je obsah prvého 9 a druhého 1, v druhom prípade sú obsahy 6 a 4. Chceme maximalizovať pravdepodobnosť toho, aby pri voľbe dvoch náhodných bodov boli obe v rovnakom obdĺžniku.*

Úlohu vieme zobecniť na klasickú pravdepodobnostno-optimalizačnú úlohu.

Príklad 2. *Majme k ekvivalenčných tried na množine s n prvkami. Ako zvoliť ekvivalenčné triedy tak, aby pri voľbe dvoch náhodných prvkov bola pravdepodobnosť toho, že oba prvky budú v tej istej ekvivalenčnej triede čo najvyššia?*

Poznámka 2. *Ekvivalenčnú triedu predstavuje obdĺžnik a množinu predstavuje množina vrcholov grafu. Alternatívne sa môžeme na úlohu pozerať ako na problém farbenia guličiek čo najmenším počtom farieb.*

Zapíšme túto úlohu formálne.

Majme n-prvkovú množinu $Prv = \{x_1, \dots, x_n\}$, k-prvkovú množinu ekvivalenčných tried $Ek = \{ek_1, \dots, ek_k\}$, veľkosť triedy $\|ek_i\|$ označme k_i a zaveďme 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 ek_i, teda $A_i = \{(x_a, x_b) | x_a, x_b \in Prv, a \neq b, f(x_a) = f(x_b) = ek_i\}$ Jav $A = \bigcup_{i=1}^{k} A_i$ teda nastáva práve vtedy, keď oba vybrané prvky patria do rovnakej triedy.

Úlohou je teda navrhnúť 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 $\sum_{i\in Ek} P[A_i] = \sum_{i=1}^{k} \frac{\binom{k_i}{2}}{\binom{|Prv|}{2}}$.

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

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

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

Teraz nám už len zostáva použiť nerovnosť $(a+b)^2 \geq a^2 + b^2$ ktorá platí pre $a, b \geq 0$, z ktorej jasne vyplýva, že potrebujeme spraviť ľubovoľné k_i čo najväčšie. Ekvivalenčné triedy musia teda byť čo najväčšie a problém sa transformuje na problém hľadania obdĺžnikov s najväčším možným obsahom. V programe tento problém rieši trieda Colorizator.

3.2.2 Nájdenie najväčšej jednotkovej podmatice

Ako sme si v úvode povedali, mriežkovú mapu vieme reprezentovať ako maticu a teda problém môžeme ekvivalentne zapísať ako problém hľadania najväčšej jednotkovej podmatice. Tento problém má riešenie v čase lineárnom od počtu vrcholov a teda nájdenie k najväčších jednotkových matíc trvá $\mathrm{O}\big(k * n\big)$, kde n je počet vrcholov matice.

Slovný popis algoritmu 3: V prvom prechode maticou si u každého vrcholu zapamätáme počet jedničiek naľavo od neho, vratane daneho vrcholu. Tento prechod trvá lineárny čas.

V druhom prechode treba prejsť zaradom všetky stĺpce zľava doprava

Algoritmus 3 Nájdenie najväčšej jednotkovej podmatice v matici mxn

Vstup: matica M rozmerov mxn nad telesom $\mathbb{Z}_2$

Výstup: pravý dolný roh podmatice aj s jej rozmermi

```
for all prvok p, p ∈ M do
  if p = 0 then
    nalavoOdPrvku(p) ← 0
  else
    nalavoOdPrvku(p) ← najdlhšia súvislá postupnosť jednotiek končiaca
    prvkom p
  end if
end for
for stĺpec s, s ∈ M  do
  Vytvor nový zásobník dvojíc (riadok, nalavoOdPrvku)
  for riadok r, r ∈ M  do
    p ← (s, r)
    while Zásobník je neprázdny do
      vyberiem prvok top z vrcholu zásobníka
      if top . nalavoOdPrvku > nalavoOdPrvku(p) then
        prvokZoZasobnika ← (top.r, s)
        DlzkaSekvencie(prvokZoZasobnika) = r - prvokZoZasobnika.r
      else
        Vložím prvok prvokZoZasobnika do zásobníka
        break
      end if
    end while
    Do zásobníka vložím dvojicu (r, nalavoOdPrvku(p))
  end for
  while Zásobník je neprázdny do
    vyberiem prvok top z vrcholu zásobníka
    prvokZoZasobnika ← (top.r, s)
    DlzkaSekvencie(prvokZoZasobnika) = m - prvokZoZasobnika.r
  end while
end for
```

4. Porovnanie algoritmu NovellA* s ostatnými algoritmami.

4.1 Vstupné dáta

Častým problémom pri vzájomnom porovnávaní algoritmov je nájsť tetsovaciu vyorku, ktor... [5]

Na porovnávanie využijeme benchmark

TODO?? bibliografia styl - priezvisko,meno - alebo naopak???

5. T-maps - Vizuálna predstava

Pre názornejšiu predstavu behu algoritmu sme navrhli softvér **T-maps**, ktorý beh algoritmov ilustruje graficky.

5.1 Použitie

Program T-maps má funčnosť veľmi podobnú programu Goole Maps. Na začiatku do neho nahrajeme mapu, nad ktorou algoritmus beží a taktiež dáta tohto algoritmu (prehľadané vrcholy a najkratšiu cestu) a program tieto hodnoty graficky znázorní. Medzi možnosti programu patrí export mapy a viditeľného poľa do súboru. TODO?? uzivatelska dokumentacia Mozno popis zaujimavej casti

Obr. 5.1: Logo MFF UK

5.1.1 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. 5.1
- Tato subsekce má číslo 5.1.1.
- Odkaz na literaturu [6].

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 3. *Tato věta neplatí.*

Dôkaz. Neplatné věty nemají důkaz. □

[LGS12]

Závěr

Zoznam použitej literatúry

[1] L. Euler. *Solutio problematis ad geometriam situs pertinentis.* Commentarii academiae scientiarum Petropolitanae, 8:128–140, 1741.

[2] E. W. Dijkstra. *A note on two problems in connexion with graphs.* Numerische Mathematik, 1(1):269–271, 1959.

[3] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. Correction to a formal basis for the heuristic determination of minimum cost paths. *SIGART Bull.*, (37):28–29, December 1972.

[4] N. Sturtevant. *GPPC: Grid-Based Path Planning Competition.* Dostupné z: `http://http://movingai.com/GPPC/`

[5] N. Sturtevant. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games*, 4(2):144 – 148, 2012.

[6] Lamport, Leslie. *LaTeX: A Document Preparation System.* 2. vydání. Massachusetts: Addison Wesley, 1994. ISBN 0-201-52983-1.

[Sh97] Shor, Peter W. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer.* SIAM J. on Computing, 1997. Pages 1484–1509.

[LGS12] Landais, Gregory, and Sendrier. *CFS Software Implementation.* Cryptology ePrint Archive, Report 2012/132, 2012.

Seznam tabulek

Seznam použitých zkratek

Přílohy