SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Pronalazak najkraćeg puta algoritmom A*

Marko Lazarić
Voditelj: *Doc. dr. sc. Marko Čupić*

Zagreb, svibanj 2019.

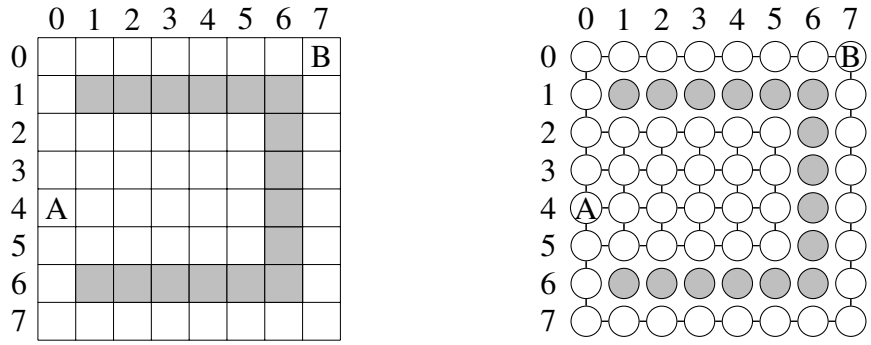
SADRŽAJ

1. Uvod

Velik broj problema može se modelirati grafom u kojem vrhovi predstavljaju stanja, a bridovi prijelaze između tih stanja, takav graf se naziva prostor stanja (engl. *state space*). Rješavanje problema se onda svodi na pretraživanje prostora stanja, odnosno pronalazak najkraćeg puta između početnog stanja i stanja koje predstavlja rješenje problema.

Zbog svoje široke primjenjivosti, razni algoritmi su osmišljeni za efikasno rješavanje tog problema. U ovom seminaru razmatrat će se prednosti i mane različitih algoritama za pronalazak najkraćeg puta između dva vrha u grafu, počevši od jednostavnijih naivnih algoritama, do složenijih za implementaciju, ali efikasnijih informiranih algoritama s naglaskom na algoritam A* i njegovu primjenu na pronalazak najkraćeg puta u cjelobrojnoj rešetci.

U svrhu jednostavne vizualizacije rada algoritama, koristit će se cjelobrojna rešetka u kojoj sivi kvadratići predstavljaju neprolazna polja, dok bijeli predstavljaju polja kroz koja se može proći. Moguće je kretati se u 4 osnovna smjera. "A" predstavlja početno stanje (vrh), a "B" završno.



Slika 1.1: Primjer jednostavne cjelobrojne rešetke i njezinog prostora stanja.

1.1. Definicija problema

Opisani problem definiran je s pet komponenti [3]:

1. Početno stanje (engl. *initial state*)
2. Moguće akcije (engl. *actions*) – opis mogućih akcija u nekom stanju.
3. Model prijelaza (engl. *transition model*) opis što svaka akcija znači.
4. Provjera riješenosti (engl. *goal test*) – provjera je li neko stanje rješenje.
5. Funkcija troška prijelaza između stanja (engl. *step cost*)

Za problem cjelobrojne rešetke, početno stanje je stanje sa slovom "A". Moguće akcije su: gore, dolje, lijevo i desno. Model prijelaza je intuitivan: gore povećava y koordinatu za jedan, dolje ju smanjuje, a lijevo i desno povećavaju, odnosno smanjuju x koordinatu. Provjera riješenosti provjerava sadrži li stanje slovo "B". Trošak neposrednog prijelaza između stanja je uvijek jednak 1. Definicija tog problema implementirana je razredom `RectangularGrid`.

1.2. Kriteriji uspoređivanja algoritama

Algoritme za rješavanje navedenog problema možemo uspoređivati po 4 kriterija [3]:

1. Potpunost – potpuni algoritam će uvijek pronaći rješenje, ako rješenje postoji.
2. Optimalnost – optimalni algoritam će uvijek pronaći optimalno rješenje, ako rješenje postoji.
3. Vremenska složenost – koliko dugo traje izvođenje algoritma u ovisnosti s veličinom ulaza.
4. Memorijska složenost – koliko memorije treba algoritam u ovisnosti s veličinom ulaza.

Pri analizi vremenske i memorijske složenosti korisne su sljedeće oznake: b faktor grananja (najveći broj sljedbenika iz nekog stanja) i d dubina rješenja s najmanjom dubinom.

2. Naivni algoritmi

Naivni algoritmi (engl. *uniformed algorithms*) nemaju dodatne informacije o stanjima, pa smatraju da svaki prijelaz ima jednaku vjerojatnost da dovede do najkraćeg puta. Njihove prednosti su jednostavna implementacija i generalnost, a mana im je velika memorijska i vremenska složenost.

2.1. Pretraživanje u širinu

Pretraživanje u širinu (engl. *breadth-first search*) je jednostavan algoritam za pronalazak najkraćeg puta u grafu s uniformnim težinama bridova. Algoritam je potpun i optimalan, a njegova vremenska i memorijska složenost je $O(b^d)$. [3]

Za pohranu vrhova koje treba obraditi koristi red (engl. *queue*), dok za pohranu već obrađenih vrhova koristi skup (engl. *set*). U svakom koraku širi se u svim mogućim smjerovima, što dovodi do velike memorijske i vremenske složenosti. Implementiran je metodom `bfs` u razredu `SearchAlgorithms`.

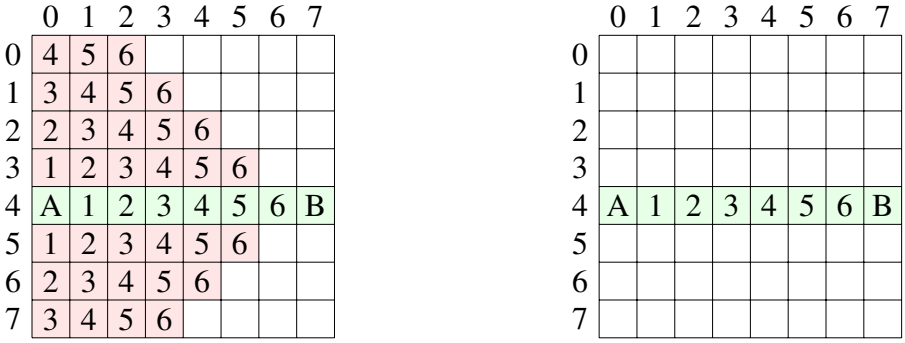
Pseudokod

Na slici 2.1 je prikazano izvođenje algoritma. Crvenom bojom su označena polja koje je algoritam obradio, ali nisu dovela do najkraćeg puta, dok je zelenom označen najkraći put. Brojevi u poljima označavaju udaljenost od početnog polja. Pretraga u širinu je obradila 41 polja, dok bi optimalno rješenje obradilo samo 8.

2.2. Pretraživanje s jednolikom cijenom

Pretraživanje s jednolikom cijenom (engl. *uniform-cost search*) je jednostavan algoritam za pronalazak najkraćeg puta u grafu s pozitivnim težinama bridova. Algoritam je potpun i optimalan, a njegova vremenska i memorijska složenost je $O\left(b^{1+\lfloor C^*/\epsilon \rfloor}\right)$ gdje C^* predstavlja cijenu optimalnog rješenja, a ϵ minimalnu težinu svih bridova. [3]

Za pohranu vrhova koristi prioritetni red (engl. *priority queue*), te hash tablicu radi



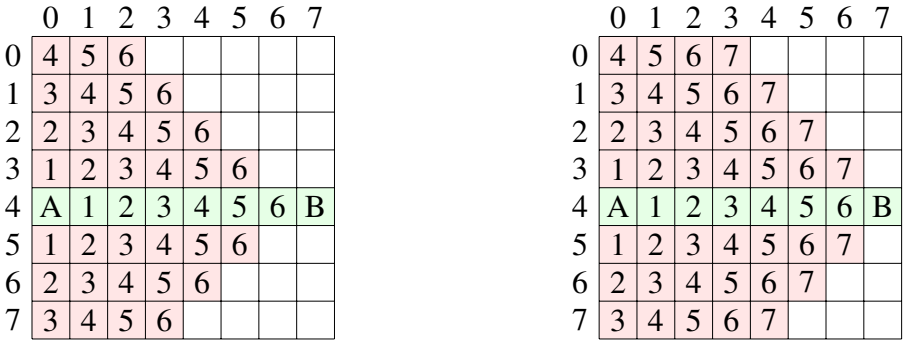
Slika 2.1: Usporedba pretraživanja u širinu i optimalnog rješenja.

efikasne provjere je li pronašao bolji put do nekog vrha koji se već nalazi u prioritetnom redu. Implementiran je metodom `ucs` u razredu `SearchAlgorithms`.

Prioritetni red koristi evaluacijsku funkciju (engl. *evaluation function*) za određivanje prioriteta, pri tome se manji broj smatra većim prioritetom. Označava se s $f(n)$. Konkretno za pretraživanje s jednolikom cijenom evaluacijska funkcija je jednaka cijenom puta od početnog vrha do tog vrha. Oznaka za cijenu puta od početka do tog vrha je $g(n)$, pa za pretraživanje s jednolikom cijenom vrijedi $f(n) = g(n)$.

Pseudokod

Za grafove s uniformnim težinama bridova, pretraživanje s jednolikom cijenom obradi strogo više vrhova od pretraživanja u širinu jer provjerava je li vrh rješenje pri obradi vrha. Posljedica toga je da algoritam obrađuje sve vrhove koji imaju istu cijenu, prije nego što obradi njihove sljedbenike.



Slika 2.2: Usporedba pretraživanja u širinu i pretraživanja s jednolikom cijenom.

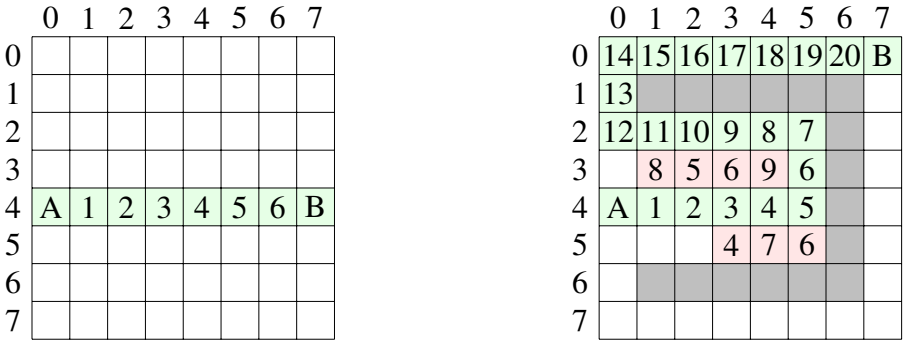
3. Informirani algoritmi

Informirani algoritmi (engl. *informed algorithms*) koriste dodatne informacije o problemu kako bi smanjili prostor stanja kojeg trebaju pretražiti. Te informacije se najčešće ugrađuju u heurističku funkciju (engl. *heuristic function*) koja se označava s $h(n)$, a predstavlja aproksimaciju najmanje cijene prijelaza od stanja n do stanja koje predstavlja rješenje. Njihova prednost je manja memorijska i vremenska složenost, a mana im je specijalizacija za pojedini problem.

3.1. Pretraživanje "najbolji prvi"

Pretraživanje "najbolji prvi" (engl. *greedy best-first-search*) je jednostavan pohlepan informiran algoritam koji nije optimalan, a potpun je jedino ako koristimo listu posjećenih stanja. [3] [1]

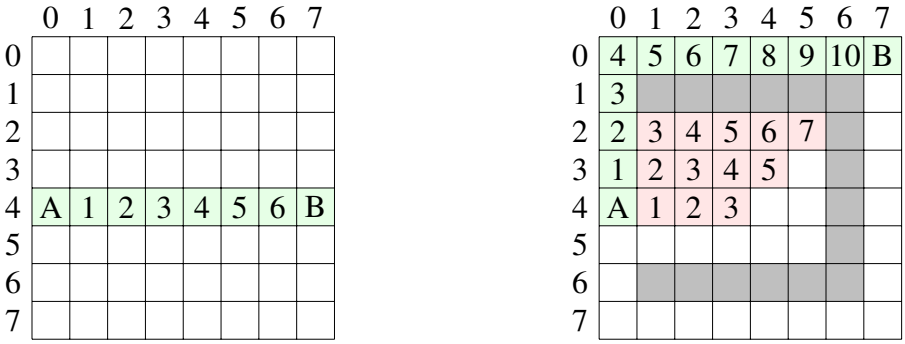
Algoritam je ekvivalentan pretraživanju s jednolikom cijenom, osim što za evaluacijsku funkciju koristi heurističku funkciju $f(n) = h(n)$. To znači da će prvo evaluirati stanja koja su bliža rješenju, no taj lokalni optimum neće nužno dovesti do optimalnog rješenja, kao što je prikazano na slici 3.1.



Slika 3.1: Rezultat pohlepnog pretraživanja.

3.2. Algoritam A*

Algoritam A* je ekvivalentan pretraživanju s jednolikom cijenom, osim što za evaluacijsku funkciju koristi sumu cijene puta od početka do tog vrha i heurističke funkcije $f(n) = g(n) + h(n)$. Uz određene uvjete na heurističku funkciju, algoritam je potpun i optimalan. [3]



Slika 3.2: Rezultat algoritma A*.

4. Primjena algoritma A* za pronalazak najkraćeg puta u cjelobrojnoj rešetci

5. Zaključak

Zaključak.

Mention navmesh

6. Literatura

[1] Bojana Dalbelo Bašić i Jan Šnajder. 3. heurističko pretraživanje, 2018. URL https://www.fer.unizg.hr/_download/repository/UI-3-HeuristickoPretrazivanje.pdf.

[2] Amit Patel. Introduction to a*. URL https://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html.

[3] Stuart J. Russell i Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, December 2002. ISBN 0137903952.

7. Sažetak

Sažetak.