SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Pronalazak najkraćeg puta algoritmom A*

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

Zagreb, travanj 2019.

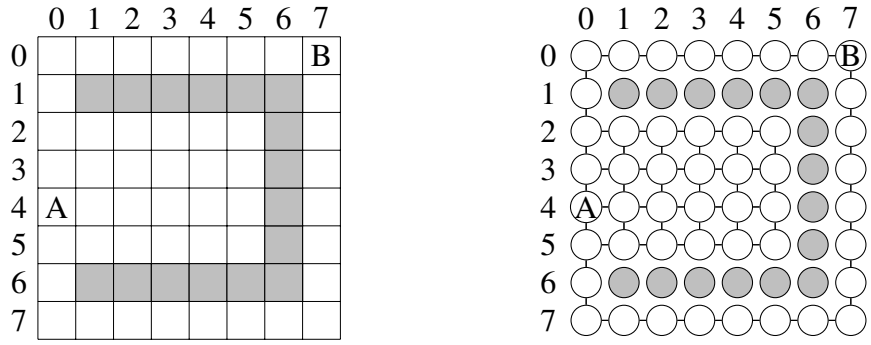
SADRŽAJ

1. Uvod

Problem pronalaska najkraćeg puta između dva vrha u grafu jedan je od najstarijih i najbolje istraženih problema u teoriji grafova. Rješenja tog problema imaju široku primjenu jer se velik broj situacija u pravom svijetu može modelirati kao grafovi, gdje vrhovi predstavljaju stanja, a bridovi prijelaze između tih stanja. Provedbom ovih algoritama minimizira se broj prijelaza, odnosno trošak za prijelaz od početnog do završnog stanja.

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 njoj ekvivalentnog grafa.

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. Njihova prednost je jednostavna implementacija, a mana je neefikasnost.

2.1. Pretraga u širinu

Pretraga u širinu (engl. *breadth-first search*) je jednostavan algoritam za pronalazak najkraćeg puta u grafu s uniformnim težinama bridova. 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 neefikasnosti samog algoritma.

Spomenuti da je BFS dobar generalni algoritam, ali da je zato vremenski i memorijski zahtjevan.

Na slici 2.1 je vidljiva ta neefikasnost. 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 44 polja, dok bi optimalni algoritam obradio samo 8.

Spomenuti vremensku i memorijsku složenost.

	0	1	2	3	4	5	6	7
0	4	5	6					
1	3	4	5	6				
2	2	3	4	5	6			
3	1	2	3	4	5	6		
4	A	1	2	3	4	5	6	B
5	1	2	3	4	5	6	7	
6	2	3	4	5	6	7		
7	3	4	5	6	7			

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4	A	1	2	3	4	5	6	B
5								
6								
7								

Slika 2.1: Usporedba pretrage u širinu i optimalnog algoritma.

```python
def pretraga_u_sirinu(pocetni, zavrsni, bridovi):
  pregledani = set()
  pronadeni = Queue()
  prethodni = dict()

  pronadeni.put(pocetni)

  while not pronadeni.empty():
    trenutni = pronadeni.get()

    if trenutni == zavrsni:
      return procitaj_put(pocetni, zavrsni, prethodni)

    for sljedeci in bridovi[trenutni]:
      if sljedeci not in pregledani:
        pregledani.add(sljedeci)
        pronadeni.put(sljedeci)
        prethodni[sljedeci] = trenutni

  return None # Ne postoji put između početnog i završnog.
```

Programski isječak 2.1: Pretraga u širinu implementirana u Pythonu.

```python
def procitaj_put(pocetni, zavrsni, prethodni):
  put = []
  trenutni = zavrsni

  while pocetni != trenutni:
    put.append(trenutni)

    trenutni = prethodni[trenutni]

  put.append(pocetni)

  return list(reversed(put)) # Obrne poredak elemenata u listi.
```

Programski isječak 2.2: Pomoćna funkcija za čitanje najkraćeg puta.

2.2. Dijkstrin algoritam

3. Informirani algoritmi

3.1. Pohlepna pretraga korištenjem prvog najboljeg čvora

engl. Greedy Best-First-Search

3.2. Algoritam A*

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

5. Zaključak

Zaključak.

6. Literatura

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

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

7. Sažetak

Sažetak.