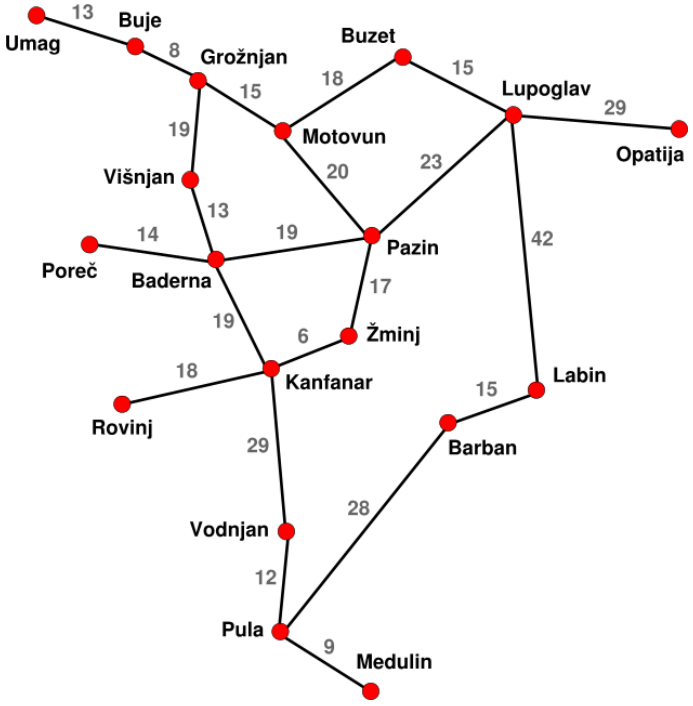
Umjetna inteligencija – Laboratorijska vježba 1

UNIZG FER, ak. god. 2019/20.

Zadano: 16.3.2020. Rok predaje: 29.3.2020. do 23.59 sati.

Pretraživanje prostora stanja: od Buzeta do slagalice 3x3 (24 boda)

U prvoj laboratorijskoj vježbi proučavat ćemo probleme rješive pretraživanjem prostora stanja, te analizirati složenost različitih algoritama slijepog i heurističkog pretraživanja. Za prvi, motivacijski problem, pokušat ćemo naći koji je stvarno najkraći put iz Pule do Buzeta kako bismo došli do divovske fritade s tartufima.



Putovanje kroz Istru

1. Učitavanje podataka

Kako bismo mogli analizirati naše algoritme na različitim problemima, definirat ćemo općeniti format ulaznih podataka za algoritme pretraživanja prostora. Konkretni primjeri formata ulaznih datoteka mogu se vidjeti na kraju uputa. Svaki od problema definiran je pomoću dvije tekstualne datoteke: (1) opisnika prostora stanja te (2) opisnika heuristike. Svaka od tekstualnih datoteka može sadržavati linije s komentarima. Takve linije uvijek počinju sa simbolom `#` i trebaju se ignorirati.

Opisnik prostora stanja sadržava informacije o početnom stanju, ciljnom stanju (ili stanjima) te prijelazima. Prva linija datoteke koja nije komentar sadrži početno stanje, dok se u drugoj liniji nalaze ciljna stanja odvojena s jednim razmakom. Preostale linije opisnika prostora stanja sastoje se od zapisa funkcije prijelaza. Svaki redak je u sljedećem formatu:

```
state: next_state_1,cost next_state_2,cost
```

Konkretan primjer retka opisnika prostora stanja:

```
Barban: Pula,28 Labin,15
```

Svi elementi jednog retka uvijek će biti odvojeni s jednim razmakom. Prvi element retka je ime izvornog stanja te dvotočka, dok svaki idući element sadrži ime ciljnog stanja te cijenu prijelaza, međusobno odvojene zarezom. Ime svakog stanja sastoji se od proizvoljno dugačkog niza simbola. Simboli dozvoljeni u imenima stanja su velika i mala slova, brojevi te podvlaka. Cijene prijelaza mogu biti i decimalni brojevi.

Opisnik heurističke funkcije sadržava informacije o vrijednosti heurističke funkcije za svako stanje. Svaki redak je u sljedećem formatu:

```
state: heuristic_value
```

Konkretan primjer retka opisnika heurističke funkcije:

```
Barban: 35
```

Imena stanja u opisniku heurističke funkcije poklapat će s onima iz opisnika prostora stanja, a vrijednost heuristike može biti decimalni broj.

Vaš prvi zadatak je implementirati učitavanje podataka iz navedenog formata u strukture prikladne za obradu algoritmima pretraživanja stanja.

Ovaj dio laboratorijske vježbe nije bodovan, no nužan je za implementaciju sljedećih zadataka.

2. Algoritmi pretraživanja (12 bodova)

Jednom kad ste učitali navedene podatke, vaš zadatak je implementirati osnovne algoritme pretraživanja prostora stanja. U sklopu ove laboratorijske vježbe, potrebno je implementirati:

1. Algoritam pretraživanja u širinu **(4 boda)**
2. Algoritam pretraživanja s jednolikom cijenom **(4 boda)**
3. Algoritam A* **(4 boda)**

Svaki od navedenih algoritama kao rezultat pretraživanja zadanog prostora stanja mora ispisivati: (1) duljinu (cijenu) pronađenog rješenja, (2) putanju pronađenog rješenja (listu stanja od početnog do ciljnog) te (3) broj otvorenih stanja pri pretraživanju (veličinu skupa *closed*). Primjer jednog takvog ispisa za algoritam A* na problemu putovanja kroz Istru dan je u nastavku:

```
States visited = 14
Found path of length 5 with total cost 100:
Pula =>
Barban =>
Labin =>
Lupoglav =>
Buzet
```

3. Provjera heurističke funkcije (12 bodova)

Kvaliteta heuristike značajno utječe na performanse algoritma A*. U slučaju da heuristika nije optimistična ili konzistentna, algoritam ne mora nužno vratiti optimalno rješenje. Takve situacije bismo htjeli izbjeći ako je to moguće. U ovom dijelu vježbe vaš je zadatak implementirati funkcije koje za zadani prostor stanja te heurističku funkciju "h(s)" provjeravaju:

1. Je li heuristička funkcija optimistična? **(6 bodova)**
2. Je li heuristička funkcija konzistentna? **(6 bodova)**

Vaše implementacije funkcija provjere moraju ispisivati **svaki** slučaj u kojemu se neki od uvjeta optimističnosti ili konzistentnosti krši. Primjerice, za provjeru optimističnosti heuristike, osim odgovora na pitanje je li heuristika optimistična, potrebno je ispisati i svako stanje za koje heuristika **precjenjuje** stvarnu cijenu puta do cilja, te za koliko ju precjenjuje. Primjer takvog ispisa za jednu lošu heuristiku za problem putovanja po Istri dan je u nastavku:

```
Checking if heuristic is optimistic.
  [ERR] h(Lupoglav) > h*: 35.0 > 15
  [ERR] h(Pazin) > h*: 40.0 > 37
Heuristic is not optimistic.

Checking if heuristic is consistent.
  [ERR] h(Lupoglav) > h(Buzet) + c: 35.0 > 0.0 + 15
  [ERR] h(Pazin) > h(Motovun) + c: 40.0 > 12.0 + 20
Heuristic is not consistent.
```

Odredite složenost vaše implementacije provjere optimističnosti i provjere konzistentnosti. Za jednostavne probleme poput putovanja po Istri, čak i naivne provjere heurističkih funkcija su dovoljno brze. Probajte pokrenuti vaše implementacije provjere optimističnosti i konzistentnosti na problemu slagalice 3x3 (datoteke `3x3_puzzle.txt`, `3x3_misplaced_heuristic.txt`). Izvode li se i one u razumnom vremenu? Mogu li se te provjere optimizirati? Razmislite kako biste optimizirali svaku od tih funkcija, pa elaborirajte svoj pristup na predaji vježbe.

Zadaci za dodatne bodove: složenost slagalice 3x3 (+6 bodova)

Napomena: Rješenja svih dodatnih zadataka moraju se predati na Ferko u istoj arhivi kao i rješenje ostatka laboratorijske vježbe. Rješenja koja ne zahtjevaju programsku implementaciju nego pismeni odgovor potrebno je također uploadati kao skenirani ili slikani dokument ili tekstualnu datoteku.

1. Rješivost slagalice 3x3 (2 boda)

Pri učitavanju prostora stanja slagalice 3x3 ispišite broj stanja koji je učitan. Je li to ukupan broj mogućih stanja slagalice 3x3? Ako nije, koji je ukupan broj mogućih stanja?

Je li moguće riješiti slagalicu 3x3 počevši iz bilo koje moguće početne konfiguracije? Ako nije, **dokažite** koji je broj stanja za koji to nije moguće i demonstrirajte jednostavan primjer početne pozicije koja je nerješiva. Ponovite istu analizu za slagalicu 4x4.

Pri dokazivanju, dopušteno je služiti se materijalima s interneta, no preporučamo da uložite neko vrijeme i probate samostalno doći do rješenja.

2. Optimizacija provjera optimističnosti i konzistentnosti (2 boda)

U sklopu laboratorijske vježbe trebali ste odgovoriti mogu li se funkcije provjere optimističnosti i konzistentnosti optimizirati, te predložiti način kako bi ih (ili jednu od njih?) optimizirali. Implementirajte i detaljno kvantificirajte vašu ideju optimizacije. **Bitno:** pripazite da vaše rješenje mora raditi na usmjerenim grafovima. Primjer usmjerenog grafa možete vidjeti u testnom prostoru stanja `ai.txt`.

Pri kvantifikaciji, (1) izračunajte konkretne složenosti vaše implementacije te (2) stvarno vremensko ubrzanje izvođenja (*wall clock time*) u usporedbi s naivnom implementacijom. Ako vaša naivna implementacija traje dulje od 5 minuta, ne morate čekati završetak izvođenja već samo napišite `5+` kao trajanje. Ako pozivate pomoćnu funkciju za izračun najkraćeg puta između dva stanja, mjerite i (3) broj poziva pomoćne funkcije. Pokrenite vaša mjerenja na barem 10 različitih početnih konfiguracija slagalice te zapišite prosjek i devijaciju (2) i (3).

Sva mjerenja potrebno je imati spremna prije termina laboratorijske vježbe!

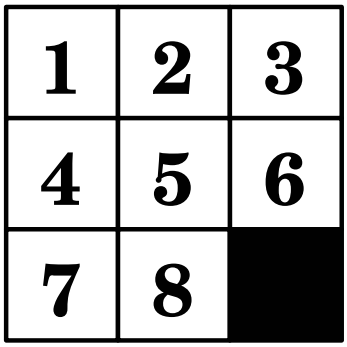
3. Dizajn heuristike (2 boda)

Vrijednost heuristike dane kao primjer za slagalicu 3x3 je broj elemenata slagalice koji su na krivim mjestima u odnosu na ciljno stanje, pri čemu za razliku od heuristike s predavanja **brojimo i prazni element**. Zadana heuristika nije optimistična niti konzistentna. Koju jednostavnu izmjenu možete napraviti kako bi ona postala optimistična i konzistentna? Možete li dati jednostavan primjer iz kojeg se vidi da ta heuristika nije optimistična?

Probajte smisliti barem dvije nove dobre heurističke funkcije. Mjerite broj posjećenih stanja algoritma A* za svaku heuristiku na barem 10 različitih početnih konfiguracija, uključujući zadano početno stanje, te spremite izlazni ispis posjećenih stanja za svaku konfiguraciju.

Zapis stanja slagalice 3x3:

Izračun vrijednosti heurističke funkcije koju dizajnirate trebali bi napraviti programski budući da je broj stanja slagalice prevelik za ručni dizajn heuristike. Za ovo, potrebno je znati kako pretvoriti naziv stanja slagalice u matrični oblik prikladan za izračun vrijednosti heuristike. Svako stanje u prostoru stanja slagalice 3x3 je idućeg oblika: `123_456_78x`. Retci slagalice odvojeni su znakom `_`, što znači da navedeno stanje odgovara idućoj slici:



Izgled stanja “123_456_78x” slagalice

Dodatak: Ulazne datoteke i očekivani ispis

U ovom poglavlju ćemo navesti nekoliko niz ulaza i izlaza pomoću kojih možete provjeriti ispravnost vašeg rješenja. Moguće je da se broj otvorenih stanja u vašoj implementaciji razlikuje, ovisno o tome kako razrješavate slučajeve s jednakim prioritetom.

1. Prolaz umjetne inteligencije

Kao dodatnu provjeru vaše implementacije, u datoteci `ai.txt` nalazi se prostor stanja za (pojednostavljenu) navigaciju prolaskom ovog predmeta. Prostor stanja je usmjeren graf s dva ciljna stanja, te je razumne veličine za ručnu provjeru rezultata.

Učitavanje podataka:

```
Start state: enroll_artificial_intelligence
End state(s): ['pass_course', 'fail_course']
State space size: 9
Total transitions: 12
```

Pretraživanje u širinu:

```
Running bfs:
States visited = 6
Found path of length 3:
enroll_artificial_intelligence =>
fail_lab =>
fail_course
```

Pretraživanje s jednolikom cijenom:

```
Running ucs:
States visited = 7
Found path of length 4 with total cost 17.0:
enroll_artificial_intelligence =>
complete_lab =>
pass_continuous =>
pass_course
```

Algoritam A* + heuristika `ai_fail.txt`:

```
Running astar:
States visited = 6
Found path of length 3 with total cost 21.0:
enroll_artificial_intelligence =>
fail_lab =>
fail_course
```

Provjere heuristika za heuristiku `ai_fail.txt`:

```
Checking heuristic
Checking if heuristic is optimistic.
  [ERR] h(pass_continuous) > h*: 20.0 > 1.0
```

```
Heuristic is not optimistic
Checking if heuristic is consistent.
  [ERR] h(complete_lab) > h(fail_continuous) + c: 10.0 > 6.0 + 1.0
  [ERR] h(enroll_artificial_intelligence) > h(fail_lab) + c: 17.0 > 1.0 + 1.0
  [ERR] h(enroll_artificial_intelligence) > h(complete_lab) + c: 17.0 > 10.0 + 4.0
  [ERR] h(pass_continuous) > h(pass_course) + c: 20.0 > 0.0 + 1.0
Heuristic is not consistent
```

Algoritam A* + heuristika `ai_pass.txt`:

```
Running astar:
States visited = 4
Found path of length 4 with total cost 17.0:
enroll_artificial_intelligence =>
complete_lab =>
pass_continuous =>
pass_course
```

Provjere heuristika za heuristiku `ai_pass.txt`:

```
Checking heuristic
Checking if heuristic is optimistic.
Heuristic is optimistic
Checking if heuristic is consistent.
Heuristic is consistent
```

2. Put do Buzeta

U idućim ispisima koristiti ćemo datoteku s opisnikom prostora stanja `istra.txt`.

Učitavanje podataka:

```
Start state: Pula
End state(s): ['Buzet']
State space size: 19
Total transitions: 44
```

Pretraživanje u širinu:

```
Running bfs:
States visited = 15
Found path of length 5:
Pula =>
Barban =>
Labin =>
Lupoglav =>
Buzet
```

Pretraživanje s jednolikom cijenom:

```
Running ucs:
States visited = 17
Found path of length 5 with total cost 100:
Pula =>
Barban =>
Labin =>
Lupoglav =>
Buzet
```

Algoritam A* + heuristika `istra_heuristic.txt`:

```
Running astar:
States visited = 14
Found path of length 5 with total cost 100:
Pula =>
Barban =>
Labin =>
Lupoglav =>
Buzet
```

Algoritam A* + heuristika `istra_pessimistic_heuristic.txt`:

```
Running astar:
States visited = 13
Found path of length 7 with total cost 102:
Pula =>
Vodnjan =>
Kanfanar =>
Žminj =>
Pazin =>
Motovun =>
Buzet
```

Provjere heuristika za heuristiku `istra_heuristic.txt`:

```
Checking if heuristic is optimistic.
Heuristic is optimistic
Checking if heuristic is consistent.
Heuristic is consistent
```

Provjere heuristika za heuristiku `istra_pessimistic_heuristic.txt`:

```
Checking if heuristic is optimistic.
  [ERR] h(Lupoglav) > h*: 35.0 > 15
  [ERR] h(Pazin) > h*: 40.0 > 38
Heuristic is not optimistic
Checking if heuristic is consistent.
  [ERR] h(Lupoglav) > h(Buzet) + c: 35.0 > 0.0 + 15
  [ERR] h(Pazin) > h(Motovun) + c: 40.0 > 12.0 + 20
Heuristic is not consistent
```

3. Slagalica 3x3

U idućim ispisima koristiti ćemo datoteku s opisnikom prostora stanja `3x3_puzzle.txt`.

Algoritam A* + heuristika `3x3_misplaced_heuristic.txt`:

```
Running astar:
States visited = 95544
Found path of length 31 with total cost 30:
876_543_21x =>
876_54x_213 =>
876_5x4_213 =>
876_x54_213 =>
876_254_x13 =>
876_254_1x3 =>
876_2x4_153 =>
876_24x_153 =>
876_243_15x =>
876_243_1x5 =>
876_2x3_145 =>
8x6_273_145 =>
x86_273_145 =>
286_x73_145 =>
286_173_x45 =>
286_173_4x5 =>
286_1x3_475 =>
2x6_183_475 =>
26x_183_475 =>
263_18x_475 =>
263_1x8_475 =>
2x3_168_475 =>
x23_168_475 =>
123_x68_475 =>
123_468_x75 =>
123_468_7x5 =>
123_468_75x =>
123_46x_758 =>
123_4x6_758 =>
123_456_7x8 =>
123_456_78x
```

Provjere heuristika za heuristiku `3x3_misplaced_heuristic.txt`:

```
Checking if heuristic is consistent.
  [ERR] 5040 errors, omitting output.
Heuristic is not consistent
Checking if heuristic is optimistic.
  [ERR] 78 errors, omitting output.
Heuristic is not optimistic
```