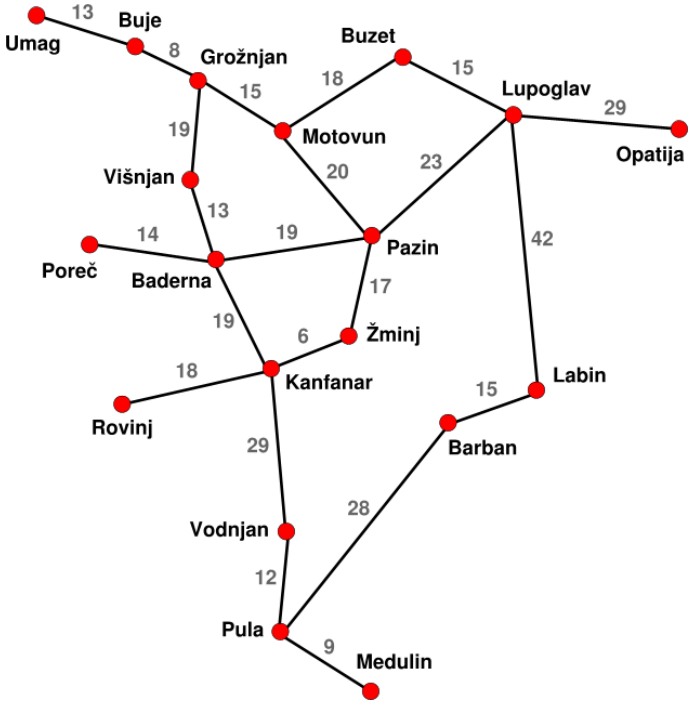
Uvod u umjetnu inteligenciju – Laboratorijska vježba 1

UNIZG FER, ak. god. 2021/22.

Zadano: 8.3.2022. Rok predaje: 27.3.2022. 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

Usprkos tome što kroz upute prolazimo kroz jedan primjer, vaša implementacija **mora** funkcionirati za sve algoritme pretraživanja prostora stanja na **svim** datotekama priloženim uz laboratorijsku vježbu u razumnom vremenu (max 2 minute). Ovaj uvjet ne uključuje provjeru optimističnosti heuristike "`3x3_misplaced_heuristic.txt`", ali uključuje sve ostale provjere heuristika i probleme pretraživanje prostora stanja za sve zadane mape (uključujući primjenu algoritama BFS, UCS, i A* na mapi "`3x3_puzzle.txt`").

Provjera koliko dobro vaša implementacija radi provodit će se pomoću *autogradera*. Očekuje se da znate pokrenuti svoje rješenje ispred ispitivača pri predaji laboratorijske vježbe pomoću *autogradera*. "Upute za autograder" pisane su uzevši u obzir da ćemo **mi**

pokretati vaše rješenje što ove godine **nećemo** raditi, nego se **vi** morate pobrinuti da se vaše rješenje može pokrenuti i evaluirati *autograderom* ispred ispitivača. Dobit ćete sve testne primjere unaprijed.

Rok za predaju arhive s rješenjem na Moodle **ne uključuje** posljednju minutu u danu, pa se pobrinite da svoju arhivu uploadate **prije** 23:59. Predaja u točno 23:59 ili nakon toga smatrat će se kasnom predajom.

Detaljno pročitajte upute za formatiranje vaših ulaza i izlaza u poglavlju "Upute za autograder", kao i primjere ispisa u poglavlju "Dodatak: Ulazne datoteke i očekivani ispis". Predana laboratorijska vježba koja se ne može pokrenuti na autograderu će se bodovati s 0 bodova **bez iznimki**. Vaš kod ne smije koristiti **nikakve dodatne vanjske biblioteke**. Ako niste sigurni smijete li koristiti neku biblioteku, provjerite je li ona dio standardnog paketa biblioteka za taj jezik.

Ukupan broj bodova po zadacima u ovoj laboratorijskoj vježbi je 24 boda. Broj bodova koji ostvarite pri predaji vježbe bit će skaliran prema postotku na 7.5 bodova.

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 (BFS) **(4 boda)**
2. Algoritam pretraživanja s jednolikom cijenom (UCS) **(4 boda)**
3. Algoritam A* (A-STAR) **(4 boda)**

Svaki od navedenih algoritama u prvom retku ispisa, koji započinje sa simbolom `#`, mora ispisati kraticu algoritma koji je korišten (BFS, UCS, ili A-STAR), te u slučaju algoritma A* putanju do opisnika heurističke funkcije korištene u algoritmu, odvojenu jednim znakom razmaka od kratice algoritma (npr., "`# A-STAR istra_heuristic.txt`").

Nakon linije s kraticom algoritma, ako je algoritam pronašao rješenje, za svaki od algoritama potrebno je ispisati informacije o sljedećih **pet** elemenata:

1. informaciju o tome je li rješenje pronađeno (`yes` ili `no`), uz prefiks "`[FOUND_SOLUTION]:`",
2. broj posjećenih stanja (veličinu skupa *closed*), uz prefiks "`[STATES_VISITED]:`",
3. duljinu pronađenog rješenja, uz prefiks "`[PATH_LENGTH]:`",
4. cijenu pronađenog rješenja (u zapisu s jednom decimalom), uz prefiks "`[TOTAL_COST]:`",
5. putanju pronađenog rješenja (listu stanja od početnog do ciljnog odvojene nizom znakova " `=>` "), uz prefiks "`[PATH]:`".

Ako rješenje nije pronađeno, dovoljno je ispisati prvi element ("`[FOUND_SOLUTION]:`") iz prethodne liste.

Primjer jednog takvog ispisa za algoritam A* na problemu putovanja kroz Istru uz heuristiku iz datoteke `istra_heuristic.txt`:

```
# A-STAR istra_heuristic.txt
[FOUND_SOLUTION]: yes
[STATES_VISITED]: 14
[PATH_LENGTH]: 5
[TOTAL_COST]: 100.0
[PATH]: Pula => Barban => Labin => Lupoglav => Buzet
```

Nužno je da i vaš ispis sadrži sve navedene elemente (ako je rješenje pronađeno), uključujući i liniju s informacijom o korištenom algoritmu, te da su vrijednosti elemenata ispisa odvojene **jednim znakom razmaka** od naziva elementa (npr. "`[PATH_LENGTH]:␣5`"). Više primjera ispisa dano je u poglavlju "Dodatak: Ulazne datoteke i očekivani ispis".

U slučaju više čvorova s istim prioritetom u listi *open* u algoritmima UCS i A*, redoslijed njihovog otvaranja definiran je prema **abecednom redu** gledajući nazive čvora. Kod BFS algoritma, potrebno je sortirati abecednim redom susjede pojedinog čvora, pa ih onda tim poretkom dodavati u listu *open*.

Pobrinite se da koristite skup *closed* (ili *visited*) u vašim implementacijama svih algoritama pretraživanja. To je potrebno kako biste osigurali da se vaš ispis podudara s očekivanim.

Pri evaluaciji autograderom, za algoritam BFS provjeravat će se podudaranje vašeg ispisa s očekivanim u poljima "`[FOUND_SOLUTION]:`" i "`[PATH_LENGTH]:`", dok će se kod algoritama UCS i A* provjeravati podudaranje s očekivanim vrijednostim u poljima "`[FOUND_SOLUTION]:`" i "`[TOTAL_COST]:`". Dakle, ako se vaše rješenje ne podudara s očekivanim u ostalim poljima, testovi će svejedno uspješno završiti, budući da do razlike između dobivenih i očekivanih vrijednosti može doći zbog razlika u implementacijama.

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)**

Ispis kod provjere uvjeta optimističnosti ili konzistentnosti mora započeti retkom u kojem je naznačeno o kojoj je provjeri riječ. Ako se radi o ispisu za provjeru optimističnosti, prvi redak ispisa mora biti u sljedećem formatu:

```
# HEURISTIC-OPTIMISTIC putanja_do_datoteke_s_heuristikom
```

Konkretan primjer prvog retka pri provjeri optimističnosti heuristike:

```
# HEURISTIC-OPTIMISTIC istra_heuristic.txt
```

Za provjeru konzistentnosti, riječ "`OPTIMISTIC`" zamijenjena je s "`CONSISTENT`" (npr., "`# HEURISTIC-CONSISTENT istra_heuristic.txt`").

Vaše implementacije funkcija provjere moraju ispisivati **svaku** provjeru uvjeta optimističnosti ili konzistentnosti, te ispisati je li u toj provjeri uvjet zadovoljen ili prekršen. Primjerice, za provjeru optimističnosti heuristike, osim odgovora na pitanje je li heuristika optimistična, potrebno je za svako stanje ispisati stvarnu cijenu puta do cilja, te vrijednost heuristike u tom stanju. Ako je vrijednost heuristike veća od stvarne cijene puta do cilja, odnosno ako heuristika **precjenjuje** stvarnu cijenu puta do cilja, potrebno je ispisati da je u tom slučaju uvjet prekršen. Ako je vrijednost heuristike manja od stvarne cijene puta do cilja, odnosno ako heuristika **ne precjenjuje** stvarnu cijenu puta do cilja, potrebno je ispisati da je u tom slučaju uvjet zadovoljen.

Skraćeni primjer takvog ispisa za jednu lošu heuristiku za problem putovanja po Istri dan je u nastavku. Potpuni ispis, koji je **nužan** u vašoj implementaciji, dan je u poglavlju "Dodatak: Ulazne datoteke i očekivani ispis".

```
# HEURISTIC-OPTIMISTIC istra_pessimistic_heuristic.txt
[CONDITION]: [OK] h(Opatija) <= h*: 26.0 <= 44.0
...
[CONDITION]: [ERR] h(Pazin) <= h*: 40.0 <= 38.0
...
[CONCLUSION]: Heuristic is not optimistic.
```

```
# HEURISTIC-CONSISTENT istra_pessimistic_heuristic.txt
[CONDITION]: [OK] h(Baderna) <= h(Višnjan) + c: 25.0 <= 20.0 + 13.0
...
[CONDITION]: [ERR] h(Lupoglav) <= h(Buzet) + c: 35.0 <= 0.0 + 15.0
...
[CONCLUSION]: Heuristic is not consistent.
```

Svaki od uvjeta koji se provjerava potrebno je prefiksirati s "`[CONDITION]:`", nakon čega je potrebno ispisati rezultat provjere, uz oznaku "`[OK]`" ako je uvjet zadovoljen, odnosno "`[ERR]`" ako uvjet nije zadovoljen. U posljednjoj liniji ispisa, ispisuje se zaključak provjere prefiksiran s "`[CONCLUSION]:`". U slučaju provjere optimističnosti, nakon prefiksa se kao zaključak ispisuje rečenica "`Heuristic is optimistic.`" ako je heuristika optimistična, odnosno "`Heuristic is not optimistic.`" ako to nije. Slično vrijedi i za provjeru konzistentnosti.

Pri ispisu rezultata provjere za dani uvjet, nužno je da vaš ispis za svaki uvjet bude jednakog formata kao u navedenom primjeru. Dakle, u slučaju provjere optimističnosti za stanje S, ispis za provjeru uvjeta u tom stanju (nakon oznake "`[OK]`" ili "`[ERR]`") mora biti u formatu:

```
h(S) <= h*: num_1 <= num_2
```

gdje je `num_1` vrijednost heuristike za to stanje iskazana u zapisu s jednom vrijednosti iza decimalne točke, a `num_2` stvarna vrijednost cijene puta do cilja od stanja S.

Pri provjeri konzistentnosti za stanje S i njemu susjedno stanje T, ispis treba biti u formatu:

```
h(S) <= h(T) + c: num_1 <= num_2 + num_3
```

gdje su `num_1` i `num_2` vrijednosti heuristike za stanja S i T, a `num_3` je cijena prelaska iz stanja S u stanje T. Vrijednosti heuristika za stanja S i T trebaju biti prikazana u decimalnom zapisu s jednom vrijednosti iza decimalne točke.

U ispisu provjera za određenu heuristiku danima u poglavlju "Dodatak: Ulazne datoteke i očekivani ispis stanja su poredana prema abecednom redu prems njihovim imenima. Preporučamo da i u vašoj implementaciji strukturirate linije ispisa na taj način, iako to nije nužno za provjeru autograderom. Dovoljno je da ispišete sve linije s provjerama za sva stanja u traženom formatu, neovisno kojim poretkom.

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.

Dodatni zadaci

Za one koji žele proširiti znanje u području pretraživanja prostora stanja, preporučujemo rješavanje sljedećih zadataka. Iako su zadaci dani u okviru laboratorijske vježbe, njihova rješenja **neće** se bodovati u ukupnom broju bodova vježbe. Ako odlučite riješiti dodatne zadatke, njihova rješenja možete predati u istoj arhivi kao i rješenje ostatka laboratorijske vježbe, te ih prodiskutirati s asistentima pri predaji vježbe. Rješenja koja ne zahtijevaju

programsku implementaciju nego pismeni odgovor potrebno je uploadati kao skenirani ili slikani dokument ili tekstualnu datoteku.

1. Rješivost slagalice 3x3

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

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).

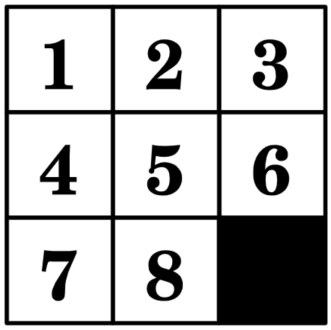
Sva mjerenja potrebno je imati spremna prije termina laboratorijske vježbe!

3. Dizajn heuristike

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

Upute za autograder

U nastavku slijede upute o strukturi arhive za upload, a detaljne upute kako pokrenuti *autograder* koristeći zadanu strukturu nalaze se u `autograder.zip` arhivi u datoteci `README.md`.

Struktura uploadane arhive

Arhiva koju uploadate na Moodle **mora** biti naziva `JMBAG.zip`, dok struktura raspakirane arhive **mora** izgledati kao u nastavku (primjer u nastavku je za Python, a primjeri za ostale jezike slijede u zasebnim poglavljima):

```
|JMBAG.zip
|-- lab1py
|----solution.py [!]
|----...
```

Arhive koje nisu predane u navedenom formatu se **neće priznavati**. Vaš kod se mora moći pokrenuti tako da prima iduće argumente putem komandne linije:

- `--alg`: kratica za algoritam za pretraživanje (vrijednosti: `bfs`, `ucs`, ili `astar`),
- `--ss`: putanja do opisnika prostora stanja,
- `--h`: putanja do opisnika heuristike,
- `--check-optimistic`: zastavica koja signalizira da se za danu heuristiku želi provjeriti optimističnost,
- `--check-consistent`: zastavica koja signalizira da se za danu heuristiku želi provjeriti konsistentnost.

Pri pokretanju vašeg rješenja na testnim primjerima, za algoritme BFS i UCS rješenju će se predavati prva dva argumenta (`--alg` i `--ss`), dok će se u slučaju A* algoritma predavati i argument s putanjom do opisnika heuristike (`--h`). U zadacima provjere optimističnosti i konzistentnosti heuristika, predavat će se argumenti `--ss`, `--h`, i `--check-optimistic` za provjeru optimističnosti, odnosno `--ss`, `--h`, i `--check-consistent` za provjeru konzistentnosti. U primjerima ispisa u poglavlju "Dodatak: Ulazne datoteke i očekivani ispis" dani su primjeri poziva funkcija s odgovarajućim argumentima u Pythonu. Za Java i C++

rješenja, vrijede ista pravila što se tiče korištenja argumenata pri pokretanju testnih primjera. **NAPOMENA**: Redoslijed argumenata predanih vašem rješenju ne mora nužno biti isti kao redoslijed kojim su navedeni u gore navedenoj listi. Pobrinite se da vaše rješenje prima argumente neovisno o njihovom redoslijedu.

Vaš kod će se pokretati na linux-u. Ovo nema poseban utjecaj na vašu konkretnu implementaciju osim ako ne hardkodirate putanje do datoteka (što **ne bi smjeli**). Vaš kod ne smije koristiti **nikakve dodatne vanjske biblioteke**. Koristite encoding `UTF-8` za vaše datoteke s izvornim kodom.

Primjer pokretanja vašeg koda pomoću autogradera za A* algoritam na prostoru stanja `istra.txt` uz heuristiku `istra_heuristic.txt` (u nastavku za Python):

```
>>> python solution.py --alg astar --ss istra.txt --h istra_heuristic.txt
```

Upute: Python

Ulazna točka vašeg koda **mora** biti u datoteci `solution.py`. Kod možete proizvoljno strukturirati po ostalim datotekama u direktoriju, ili sav kod ostaviti u `solution.py`. Vaš kod će se uvijek pokretati iz direktorija vježbe (`lab1py`).

Struktura direktorija i primjer naredbe mogu se vidjeti na kraju prethodnog poglavlja. Verzija Pythona na kojoj će se vaš kod pokretati biti će `Python 3.7.4`.

Upute: Java

Uz objavu laboratorijske vježbe objaviti ćemo i predložak Java projekta koji možete importati u vaš IDE. Struktura unutar arhive `JMBAG.zip` definirana je u predlošku i izgleda ovako:

```
|JMBAG.zip
|--lab1java
|----src
|------main.java.ui
|--------Solution.java [!]
|--------...
|----target
|----pom.xml
```

Ulazna točka vašeg koda **mora** biti u datoteci `Solution.java`. Kod možete proizvoljno strukturirati po ostalim datotekama unutar direktorija, ili sav kod ostaviti u `Solution.java`. Vaš kod će se kompajlirati pomoću Mavena.

Primjer pokretanja vašeg koda pomoću autogradera za A* algoritam na prostoru stanja `istra.txt` uz heuristiku `istra_heuristic.txt` (iz direktorija `lab1java`):

```
>>> mvn compile
>>> java -cp target/classes ui.Solution --alg astar --ss istra.txt --h
    istra_heuristic.txt
```

Informacije vezano za verzije Mavena i Jave:

```
>>> mvn -version
Apache Maven 3.6.3
Maven home: /opt/maven
Java version: 15.0.2, vendor: Oracle Corporation, runtime: /opt/jdk-15.0.2
```

```
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-139-generic", arch: "amd64", family: "unix"

>>> java -version
openjdk version "15.0.2" 2021-01-19
OpenJDK Runtime Environment (build 15.0.2+7-27)
OpenJDK 64-Bit Server VM (build 15.0.2+7-27, mixed mode, sharing)
```

Bitno: provjerite da se vaša implementacija može kompajlirati s zadanom `pom.xml` datotekom.

Upute: C++

Struktura unutar arhive `JMBAG.zip` mora izgledati ovako:

```
|JMBAG.zip
|--lab1cpp
|----solution.cpp [!]
|----...
```

Ako u arhivi koju predate ne postoji `Makefile`, za kompajliranje vašeg koda iskoristiti će se `Makefile` koji je dostupan uz vježbu. **Ako** predate `Makefile` u arhivi (ne preporučamo, osim ako stvarno ne znate što radite), očekujemo da on funkcionira.

Primjer pokretanja vašeg koda pomoću autogradera za A* algoritam na prostoru stanja `istra.txt` uz heuristiku `istra_heuristic.txt` (iz direktorija `lab1cpp`):

```
>>> make
>>> ./solution --alg astar --ss istra.txt --h istra_heuristic.txt
```

Informacije vezano za gcc:

```
>>> gcc --version
gcc (Ubuntu 9.3.0-11ubuntu0~18.04.1) 9.3.0
```

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.

Napomena: U nekim od linija ispisa linije su prelomljene kako bi stale u prostor dokumenta. U vašem ispisu, vrijednosti svakog elementa ispisa trebaju biti ispisane u jednom retku.

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.

Pretraživanje u širinu:

```
# BFS
[FOUND_SOLUTION]: yes
```

```
[STATES_VISITED]: 6
[PATH_LENGTH]: 3
[TOTAL_COST]: 21.0
[PATH]: enroll_artificial_intelligence => fail_lab => fail_course
```

Pretraživanje s jednolikom cijenom:

```
# UCS
[FOUND_SOLUTION]: yes
[STATES_VISITED]: 7
[PATH_LENGTH]: 4
[TOTAL_COST]: 17.0
[PATH]: enroll_artificial_intelligence => complete_lab => pass_continuous =>
    pass_course
```

Algoritam A* + heuristika `ai_fail.txt`:

```
# A-STAR ai_fail.txt
[FOUND_SOLUTION]: yes
[STATES_VISITED]: 6
[PATH_LENGTH]: 3
[TOTAL_COST]: 21.0
[PATH]: enroll_artificial_intelligence => fail_lab => fail_course
```

Provjera optimističnosti za heuristiku `ai_fail.txt`:

```
# HEURISTIC-OPTIMISTIC ai_fail.txt
[CONDITION]: [OK] h(complete_lab) <= h*: 10.0 <= 13.0
[CONDITION]: [OK] h(enroll_artificial_intelligence) <= h*: 17.0 <= 17.0
[CONDITION]: [OK] h(fail_continuous) <= h*: 6.0 <= 17.0
[CONDITION]: [OK] h(fail_course) <= h*: 0.0 <= 0.0
[CONDITION]: [OK] h(fail_exam) <= h*: 5.0 <= 20.0
[CONDITION]: [OK] h(fail_lab) <= h*: 1.0 <= 17.0
[CONDITION]: [ERR] h(pass_continuous) <= h*: 20.0 <= 1.0
[CONDITION]: [OK] h(pass_course) <= h*: 0.0 <= 0.0
[CONDITION]: [OK] h(pass_exam) <= h*: 1.0 <= 1.0
[CONCLUSION]: Heuristic is not optimistic.
```

Provjera konzistentnosti za heuristiku `ai_fail.txt`:

```
# HEURISTIC-CONSISTENT ai_fail.txt
[CONDITION]: [ERR] h(complete_lab) <= h(fail_continuous) + c: 10.0 <= 6.0 + 1.0
[CONDITION]: [OK] h(complete_lab) <= h(pass_continuous) + c: 10.0 <= 20.0 + 12.0
[CONDITION]: [ERR] h(enroll_artificial_intelligence) <= h(complete_lab) + c:
    17.0 <= 10.0 + 4.0
[CONDITION]: [ERR] h(enroll_artificial_intelligence) <= h(fail_lab) + c: 17.0 <=
    1.0 + 1.0
[CONDITION]: [OK] h(fail_continuous) <= h(fail_exam) + c: 6.0 <= 5.0 + 1.0
[CONDITION]: [OK] h(fail_continuous) <= h(pass_exam) + c: 6.0 <= 1.0 + 16.0
[CONDITION]: [OK] h(fail_exam) <= h(fail_course) + c: 5.0 <= 0.0 + 20.0
[CONDITION]: [OK] h(fail_lab) <= h(complete_lab) + c: 1.0 <= 10.0 + 4.0
[CONDITION]: [OK] h(fail_lab) <= h(fail_course) + c: 1.0 <= 0.0 + 20.0
[CONDITION]: [OK] h(fail_lab) <= h(fail_lab) + c: 1.0 <= 1.0 + 1.0
[CONDITION]: [ERR] h(pass_continuous) <= h(pass_course) + c: 20.0 <= 0.0 + 1.0
[CONDITION]: [OK] h(pass_exam) <= h(pass_course) + c: 1.0 <= 0.0 + 1.0
```

```
[CONCLUSION]: Heuristic is not consistent.
```

Algoritam A* + heuristika `ai_pass.txt`:

```
# A-STAR ai_pass.txt
[FOUND_SOLUTION]: yes
[STATES_VISITED]: 4
[PATH_LENGTH]: 4
[TOTAL_COST]: 17.0
[PATH]: enroll_artificial_intelligence => complete_lab => pass_continuous =>
    pass_course
```

Provjera optimističnosti za heuristiku `ai_pass.txt`:

```
# HEURISTIC-OPTIMISTIC ai_pass.txt
[CONDITION]: [OK] h(complete_lab) <= h*: 13.0 <= 13.0
[CONDITION]: [OK] h(enroll_artificial_intelligence) <= h*: 17.0 <= 17.0
[CONDITION]: [OK] h(fail_continuous) <= h*: 17.0 <= 17.0
[CONDITION]: [OK] h(fail_course) <= h*: 0.0 <= 0.0
[CONDITION]: [OK] h(fail_exam) <= h*: 20.0 <= 20.0
[CONDITION]: [OK] h(fail_lab) <= h*: 17.0 <= 17.0
[CONDITION]: [OK] h(pass_continuous) <= h*: 1.0 <= 1.0
[CONDITION]: [OK] h(pass_course) <= h*: 0.0 <= 0.0
[CONDITION]: [OK] h(pass_exam) <= h*: 1.0 <= 1.0
[CONCLUSION]: Heuristic is optimistic.
```

Provjera konzistentnosti za heuristiku `ai_pass.txt`:

```
# HEURISTIC-CONSISTENT ai_pass.txt
[CONDITION]: [OK] h(complete_lab) <= h(fail_continuous) + c: 13.0 <= 17.0 + 1.0
[CONDITION]: [OK] h(complete_lab) <= h(pass_continuous) + c: 13.0 <= 1.0 + 12.0
[CONDITION]: [OK] h(enroll_artificial_intelligence) <= h(complete_lab) + c: 17.0
    <= 13.0 + 4.0
[CONDITION]: [OK] h(enroll_artificial_intelligence) <= h(fail_lab) + c: 17.0 <=
    17.0 + 1.0
[CONDITION]: [OK] h(fail_continuous) <= h(fail_exam) + c: 17.0 <= 20.0 + 1.0
[CONDITION]: [OK] h(fail_continuous) <= h(pass_exam) + c: 17.0 <= 1.0 + 16.0
[CONDITION]: [OK] h(fail_exam) <= h(fail_course) + c: 20.0 <= 0.0 + 20.0
[CONDITION]: [OK] h(fail_lab) <= h(complete_lab) + c: 17.0 <= 13.0 + 4.0
[CONDITION]: [OK] h(fail_lab) <= h(fail_course) + c: 17.0 <= 0.0 + 20.0
[CONDITION]: [OK] h(fail_lab) <= h(fail_lab) + c: 17.0 <= 17.0 + 1.0
[CONDITION]: [OK] h(pass_continuous) <= h(pass_course) + c: 1.0 <= 0.0 + 1.0
[CONDITION]: [OK] h(pass_exam) <= h(pass_course) + c: 1.0 <= 0.0 + 1.0
[CONCLUSION]: Heuristic is consistent.
```

2. Put do Buzeta

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

Pretraživanje u širinu:

```
# BFS
[FOUND_SOLUTION]: yes
[STATES_VISITED]: 11
[PATH_LENGTH]: 5
```

```
[TOTAL_COST]: 100.0
[PATH]: Pula => Barban => Labin => Lupoglav => Buzet
```

Pretraživanje s jednolikom cijenom:

```
# UCS
[FOUND_SOLUTION]: yes
[STATES_VISITED]: 17
[PATH_LENGTH]: 5
[TOTAL_COST]: 100.0
[PATH]: Pula => Barban => Labin => Lupoglav => Buzet
```

Algoritam A* + heuristika `istra_heuristic.txt`:

```
# A-STAR istra_heuristic.txt
[FOUND_SOLUTION]: yes
[STATES_VISITED]: 14
[PATH_LENGTH]: 5
[TOTAL_COST]: 100.0
[PATH]: Pula => Barban => Labin => Lupoglav => Buzet
```

Algoritam A* + heuristika `istra_pessimistic_heuristic.txt`:

```
# A-STAR istra_pessimistic_heuristic.txt
[FOUND_SOLUTION]: yes
[STATES_VISITED]: 13
[PATH_LENGTH]: 7
[TOTAL_COST]: 102.0
[PATH]: Pula => Vodnjan => Kanfanar => Žminj => Pazin => Motovun => Buzet
```

Provjera optimističnosti za heuristiku `istra_pessimistic_heuristic.txt`:

```
# HEURISTIC-OPTIMISTIC istra_pessimistic_heuristic.txt
[CONDITION]: [OK] h(Baderna) <= h*: 25.0 <= 57.0
[CONDITION]: [OK] h(Barban) <= h*: 35.0 <= 72.0
[CONDITION]: [OK] h(Buje) <= h*: 21.0 <= 41.0
[CONDITION]: [OK] h(Buzet) <= h*: 0.0 <= 0.0
[CONDITION]: [OK] h(Grožnjan) <= h*: 17.0 <= 33.0
[CONDITION]: [OK] h(Kanfanar) <= h*: 30.0 <= 61.0
[CONDITION]: [OK] h(Labin) <= h*: 35.0 <= 57.0
[CONDITION]: [ERR] h(Lupoglav) <= h*: 35.0 <= 15.0
[CONDITION]: [OK] h(Medulin) <= h*: 61.0 <= 109.0
[CONDITION]: [OK] h(Motovun) <= h*: 12.0 <= 18.0
[CONDITION]: [OK] h(Opatija) <= h*: 26.0 <= 44.0
[CONDITION]: [ERR] h(Pazin) <= h*: 40.0 <= 38.0
[CONDITION]: [OK] h(Poreč) <= h*: 32.0 <= 71.0
[CONDITION]: [OK] h(Pula) <= h*: 57.0 <= 100.0
[CONDITION]: [OK] h(Rovinj) <= h*: 40.0 <= 79.0
[CONDITION]: [OK] h(Umag) <= h*: 31.0 <= 54.0
[CONDITION]: [OK] h(Višnjan) <= h*: 20.0 <= 52.0
[CONDITION]: [OK] h(Vodnjan) <= h*: 47.0 <= 90.0
[CONDITION]: [OK] h(Žminj) <= h*: 27.0 <= 55.0
[CONCLUSION]: Heuristic is not optimistic.
```

Provjera konzistentnosti za heuristiku `istra_pessimistic_heuristic.txt`:

```
# HEURISTIC-CONSISTENT istra_pessimistic_heuristic.txt
[CONDITION]: [OK] h(Baderna) <= h(Kanfanar) + c: 25.0 <= 30.0 + 19.0
[CONDITION]: [OK] h(Baderna) <= h(Pazin) + c: 25.0 <= 40.0 + 19.0
[CONDITION]: [OK] h(Baderna) <= h(Poreč) + c: 25.0 <= 32.0 + 14.0
[CONDITION]: [OK] h(Baderna) <= h(Višnjan) + c: 25.0 <= 20.0 + 13.0
[CONDITION]: [OK] h(Barban) <= h(Labin) + c: 35.0 <= 35.0 + 15.0
[CONDITION]: [OK] h(Barban) <= h(Pula) + c: 35.0 <= 57.0 + 28.0
[CONDITION]: [OK] h(Buje) <= h(Grožnjan) + c: 21.0 <= 17.0 + 8.0
[CONDITION]: [OK] h(Buje) <= h(Umag) + c: 21.0 <= 31.0 + 13.0
[CONDITION]: [OK] h(Buzet) <= h(Lupoglav) + c: 0.0 <= 35.0 + 15.0
[CONDITION]: [OK] h(Buzet) <= h(Motovun) + c: 0.0 <= 12.0 + 18.0
[CONDITION]: [OK] h(Grožnjan) <= h(Buje) + c: 17.0 <= 21.0 + 8.0
[CONDITION]: [OK] h(Grožnjan) <= h(Motovun) + c: 17.0 <= 12.0 + 15.0
[CONDITION]: [OK] h(Grožnjan) <= h(Višnjan) + c: 17.0 <= 20.0 + 19.0
[CONDITION]: [OK] h(Kanfanar) <= h(Baderna) + c: 30.0 <= 25.0 + 19.0
[CONDITION]: [OK] h(Kanfanar) <= h(Rovinj) + c: 30.0 <= 40.0 + 18.0
[CONDITION]: [OK] h(Kanfanar) <= h(Vodnjan) + c: 30.0 <= 47.0 + 29.0
[CONDITION]: [OK] h(Kanfanar) <= h(Žminj) + c: 30.0 <= 27.0 + 6.0
[CONDITION]: [OK] h(Labin) <= h(Barban) + c: 35.0 <= 35.0 + 15.0
[CONDITION]: [OK] h(Labin) <= h(Lupoglav) + c: 35.0 <= 35.0 + 42.0
[CONDITION]: [ERR] h(Lupoglav) <= h(Buzet) + c: 35.0 <= 0.0 + 15.0
[CONDITION]: [OK] h(Lupoglav) <= h(Labin) + c: 35.0 <= 35.0 + 42.0
[CONDITION]: [OK] h(Lupoglav) <= h(Opatija) + c: 35.0 <= 26.0 + 29.0
[CONDITION]: [OK] h(Lupoglav) <= h(Pazin) + c: 35.0 <= 40.0 + 23.0
[CONDITION]: [OK] h(Medulin) <= h(Pula) + c: 61.0 <= 57.0 + 9.0
[CONDITION]: [OK] h(Motovun) <= h(Buzet) + c: 12.0 <= 0.0 + 18.0
[CONDITION]: [OK] h(Motovun) <= h(Grožnjan) + c: 12.0 <= 17.0 + 15.0
[CONDITION]: [OK] h(Motovun) <= h(Pazin) + c: 12.0 <= 40.0 + 20.0
[CONDITION]: [OK] h(Opatija) <= h(Lupoglav) + c: 26.0 <= 35.0 + 29.0
[CONDITION]: [OK] h(Pazin) <= h(Baderna) + c: 40.0 <= 25.0 + 19.0
[CONDITION]: [OK] h(Pazin) <= h(Lupoglav) + c: 40.0 <= 35.0 + 23.0
[CONDITION]: [ERR] h(Pazin) <= h(Motovun) + c: 40.0 <= 12.0 + 20.0
[CONDITION]: [OK] h(Pazin) <= h(Žminj) + c: 40.0 <= 27.0 + 17.0
[CONDITION]: [OK] h(Poreč) <= h(Baderna) + c: 32.0 <= 25.0 + 14.0
[CONDITION]: [OK] h(Pula) <= h(Barban) + c: 57.0 <= 35.0 + 28.0
[CONDITION]: [OK] h(Pula) <= h(Medulin) + c: 57.0 <= 61.0 + 9.0
[CONDITION]: [OK] h(Pula) <= h(Vodnjan) + c: 57.0 <= 47.0 + 12.0
[CONDITION]: [OK] h(Rovinj) <= h(Kanfanar) + c: 40.0 <= 30.0 + 18.0
[CONDITION]: [OK] h(Umag) <= h(Buje) + c: 31.0 <= 21.0 + 13.0
[CONDITION]: [OK] h(Višnjan) <= h(Baderna) + c: 20.0 <= 25.0 + 13.0
[CONDITION]: [OK] h(Višnjan) <= h(Grožnjan) + c: 20.0 <= 17.0 + 19.0
[CONDITION]: [OK] h(Vodnjan) <= h(Kanfanar) + c: 47.0 <= 30.0 + 29.0
[CONDITION]: [OK] h(Vodnjan) <= h(Pula) + c: 47.0 <= 57.0 + 12.0
[CONDITION]: [OK] h(Žminj) <= h(Kanfanar) + c: 27.0 <= 30.0 + 6.0
[CONDITION]: [OK] h(Žminj) <= h(Pazin) + c: 27.0 <= 40.0 + 17.0
[CONCLUSION]: 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` (ispis za element "`[PATH]:`" je u jednom retku, ali je prelomljen u ispisu u ovom dokumentu):

```
# A-STAR 3x3_misplaced_heuristic.txt
[FOUND_SOLUTION]: yes
[STATES_VISITED]: 95544
[PATH_LENGTH]: 31
[TOTAL_COST]: 30.0
[PATH]: 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
```