Informirano pretraživanje: uputa za samostalno učenje

U nastavku je dana uputa za učenje gradiva s referencama na materijale te zadatcima koje trebate riješiti. Zadatci su napisani crvenim slovima.

Proučite slideove o informiranom pretraživanju te pročitajte u skripti poglavlje 3 (Informirano pretraživanje).

Odgovorite na sljedeća pitanja.

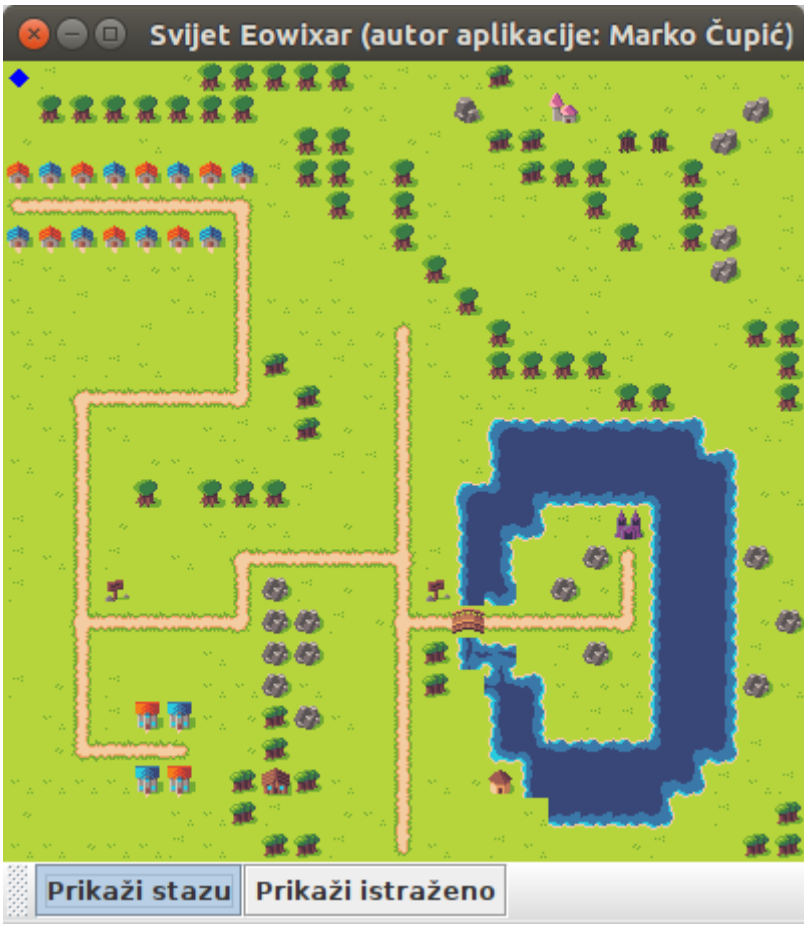
1. U kontekstu informiranog pretraživanja, dajte formalnu definiciju heurističke funkcije.
2. Što znači da je neka heuristička funkcije optimistična?
3. Što znači da je neka heuristička funkcija konzistentna?
4. Povlači li optimističnost konzistentnost? Povlači li konzistentnost optimističnost?
5. Za graf stanja zadan u nastavku, provedite sve korake postupkom pretraživanja s jednolikom cijenom, uz uporabu kolekcije posjećenih stanja, ako je (a) početno stanje, a (g) ciljno.
6. Za graf stanja zadan u nastavku, provedite sve korake pretraživanja postupkom A* uz uporabu kolekcije posjećenih stanja, ako je (a) početno stanje, a (g) ciljno, te uporabom heuristike h1 (ispod je definicija).
7. Utvrdite je li heuristika h1 optimistična? Je li konzistentna?
8. Što znači da je neka heuristika informiranija od druge?
9. Neka je h2 najinformiranija optimistična heuristika. Odredite je.
10. Provedite postupak pretraživanja algoritmom A* uz kolekciju posjećenih stanja i heuristiku h2, ako je (a) početno stanje, a (g) ciljno.
11. Kakav utjecaj ima informiranost heuristike na broj čvorova koje će algoritam pretraživanja otvarati?

Graf uz pitanja 5. - 10. Funkcija sljedbenika navodi skup sljedeće stanje – cijena prijelaza.
succ(a) = {(b,25), (c,15)}
succ(b) = {(a,25), (d,30), (e,30)}
succ(c) = {(a,15), (e,20), (f,40)}
succ(d) = {(b,30), (g,20)}
succ(e) = {(b,30),(c,20),(g,25)}
succ(f) = {(c,40),(g,100)}
succ(g) = {(d,20),(e,25),(f,100)}

Heuristika h1:
h(a)=55, h(b)=40, h(c)=42, h(d)=15, h(e)=20, h(f)=80, h(g)=0

Programski zadatak

Na planetu Eowixar život je relativno miran. Stanovnici se bave poljoprivredom i stočarstvom te u miru provode dane. Dio površine planeta prikazan je na slici u nastavku.



Za ovaj svijet potrebno je implementirati programsku podršku koja će omogućiti kretanje stanovnika Javka uz minimalni utrošak energije. Javko se inicijalno nalazi u gornjem lijevom uglu (redak 0, stupac 0), što je označeno plavim rombom.

Kretanjem po travi, Javko gubi 5 jedinica energije. Kretanjem po cesti, Javko gubi jednu jedinicu energije (u oba slučaja, po svakom napravljenom koraku). Javko se može kretati samo po travi i cesti – ne zna plivati, ne može se penjati po drveću niti preskakati kamenje. Korisnik mišem može kliknuti na bilo koji dio mape, i naš je zadatak pronaći put do te pozicije koji troši minimalnu količinu energije (ako je put uopće moguć).

Kako bismo započeli rješavati ovaj problem, pripremljena je biblioteka `apps-eowixar.jar` (skinite je i dodajte u novi Eclipse projekt). Paket `model` sadrži sučelje `IPosition` koje modelira jednu poziciju u ovome svijetu te sučelje `IPathFinder` koje modelira objekte koje možemo iskoristiti da nam pronađu optimalan put. Ova dva sučelja dana su i u nastavku.

```
package model;

public interface IPosition {
	int getRow();
	int getColumn();
}
```

```
package model;

public interface IPathFinder {
	SearchResult findPath(IPosition start, IPosition end);
}
```

Isti paket sadrži i razred `SearchResult` koji sadrži informacije prikupljene prilikom traženja staze. Njegova definicija dana je u nastavku.

```
package model;

import java.util.List;
import java.util.Optional;
import java.util.OptionalDouble;

public class SearchResult {

	private Optional<List<? extends IPosition>> path;
	private OptionalDouble cost;
	private List<? extends IPosition> exploredPositions;

	public SearchResult(Optional<List<? extends IPosition>> path, OptionalDouble cost,
					List<? extends IPosition> exploredPositions) {
		super();
		this.path = path;
		this.cost = cost;
		this.exploredPositions = exploredPositions;
	}

	public OptionalDouble getCost() {
		return cost;
	}

	public List<? extends IPosition> getExploredPositions() {
		return exploredPositions;
	}

	public Optional<List<? extends IPosition>> getPath() {
		return path;
	}

}
```

Objekt koji radi pretraživanje treba preko objekta koji je primjerak razreda `SearchResult` pozivatelju vratiti opcionalnu cijenu pronađenog puta te opcionalnu listu pozicija koje čine put (ako put ne postoji, oboje treba biti prazno), te treba vratiti listu svih pozicija koje je algoritam razmatrao (sva stanja koja su ikada prošla kroz kolekciju otvorenih čvorova kao stanja tih čvorova).

Evo što znamo o svijetu Eowixar. Prikazani svijet sastoji 625 ćelija raspoređenih u 25 redaka i 25 stupaca. Retke i stupce numeriramo počev od 0. Sadržaj ćelija dostupan je kroz tekstovnu datoteku koja ima 25 redaka, i u svakom retku niz od 25 brojeva odijeljenih razmakom. U jednom koraku, Javko se može pomaknuti na jednu od četiri susjedne ćelije (ćeliju iznad, ispod, lijevo ili desno); dijagonalni pomaci nisu dozvoljeni.

Evo primjera formata datoteke (dana su prva dva retka) s podatcima za prikazani svijet.

```
34 36 34 34 34 35 105 105 105 105 105 37 36 37 37 104 37 37 37 34 34 37 37 34 37
34 105 105 105 105 105 105 105 34 34 35 37 34 34 97 34 37 4 37 34 35 35 34 89 34
```

U prvom retku prvih 6 ćelija sadrži travnatu površinu, sljedećih 5 sadrži drveće, itd.

Ćelije u koje je upisan neki od brojeva iz skupa {34, 35, 36, 37} predstavljaju travu. Ćelije u koje je upisan neki od brojeva iz skupa {38, 90, 91, 92, 93, 95, 99, 101, 102, 103, 108, 109, 110} predstavljaju cestu. Sve ostale ćelije sadrže nešto po čemu se Javko ne može kretati.

U biblioteci je dostupan razred `Loader`, čijom uporabom možete dobiti `InputStream` prema ovoj datoteci:

```
InputStream is = Loader.findInputStream("mapa.txt");
```

Navedena datoteka dio je biblioteke i navedenim ćete je pozivom moći pročitati.

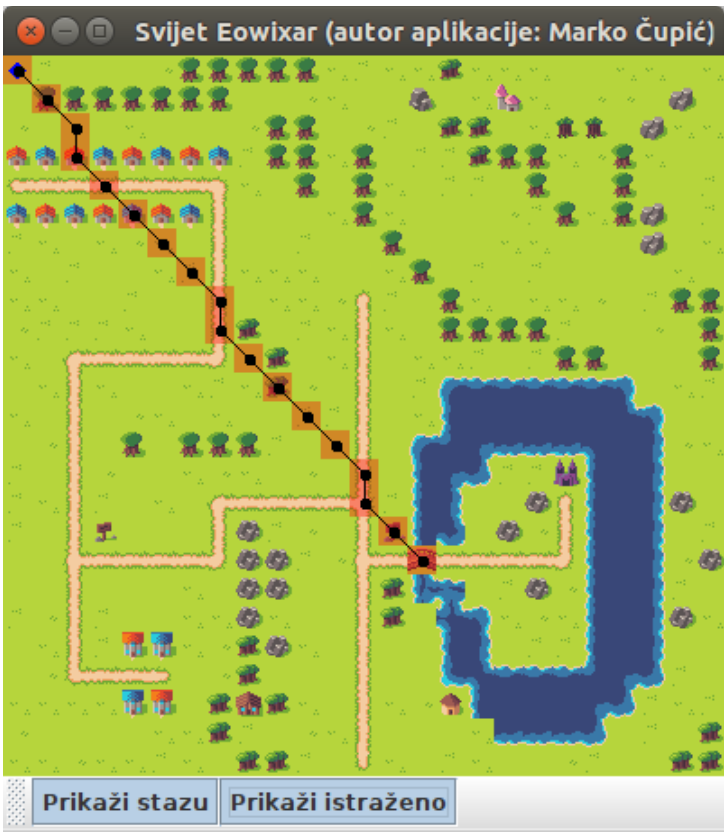
U paketu `gui` dostupan je razred `Playground` koji ima javnu statičku metodu:

```
public static void playWith(IPathFinder pathFinder);
```

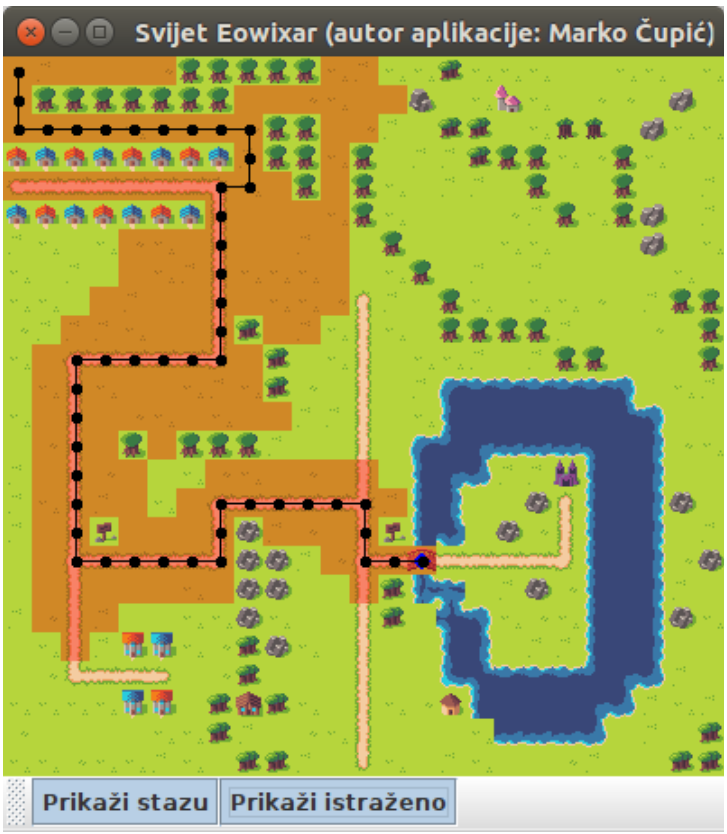
Metodi treba predati referencu na objekt koji će raditi pretraživanje, i metoda će dalje napraviti sve potrebno (otvoriti prozor, prikazati mapu te svaki puta kada korisnik klikne negdje na mapu, pitat će ovaj predani objekt da pronađe optimalan put). Za vraćeni put, napravit će i niz provjera (tipa: kreće li se Javko preko nedozvoljenih ćelija i slično). Pretpostavimo da je razred NaivePathFinder razred koji implementira sučelje `IpathFinder`. Glavni program koji će njega koristiti za pretraživanje prikazan je u nastavku.

```
package demo;

import gui.Playground;

public class Pokreni {

	public static void main(String[] args) {
		Playground.playWith(new NaivePathFinder());
	}
}
```

Algoritam koji je u njemu implementiran gradi stazu po izravnoj spojnici između početnog i konačnog stanja i pretpostavlja da je utrošak energije uvijek 1 za svaki korak. Stoga će u većini slučajeva njegova rješenja biti nevaljala i s pogrešnom cijenom. Pokretanjem gornjeg programa, pa klikom na most dobit ćete sljedeći prikaz (vidi sliku u nastavku). Pri tome će program otvoriti dijaloški okvir u kojem će prijaviti niz problema koje to rješenje ima. U prozoru je pritisnut i gumb "Prikaži istraženo", zbog čega su u prikazu zacrvenjene ćelije koje je program pretraživanja razmatrao, i one se podudaraju sa samom vraćenom stazom (program nije razmatrao ništa više od ćelija koje je odabrao za stazu).



Implementirajte objekt koji će pretraživanje raditi uporabom algoritma A* uz kolekciju posjećenih stanja i Manhattan heuristiku. Uz taj algoritam, cijena pronađenog puta trebala bi biti 95, a slika:



Put od mosta pa do ćelije koja je desno od kuće na vrhu trebao bi imati cijenu 216; algoritam bi pri tome trebao razmotriti sve ćelije na kojima Javko može boraviti: