Priprema za vežbe iz predmeta Osnovi računarske inteligencije

Redni broj vežbe: 2

Oblast:

**PRETRAGE**

1. Otvoriti projekat **Lavirint.sln**.
2. Uvodna napomena – u ovom zadatku, lavirint je definisan sa određenim brojem kolona i vrsta. Njihova deklaracija se nalazi u klasi **Main.cs**. Svako stanje, između ostalog je i opisano vrstom i kolonom, odnosno pozicijom u lavirintu. Početno stanje je predstavljeno zelenom, a krajnje stanje crvenom bojom. Odabrani algoritam treba da pronađe put između početnog i krajnjeg stanja.
3. *TODO 1*: Implementirati metodu **getLinkedNodes()** u klasi **Node.cs**. Ova metoda određuje dozvoljeno kretanje (gore, dole, levo, desno) u lavirintu u odnosnu na trenutnu vrstu i kolonu. *Napomena*: broj vrsta i broj kolona lavirinta su statičke promenljive deklarisane u klasi **Main.cs** kojima se pristupa sa **Main.lavirint.brojVrsta** i **Main.lavirint.brojKolona**.
4. *TODO 2*: Implementirati metodu **validCoords()** tako da se pri odabiru sledećeg stanja ne uzima u obzir pozicija u lavirintu obeležena sa „sivom kutijom“, ili van granica lavirinta. Ovu metodu iskoristiti u logici metode iz prethodnog zadatka.
5. *TODO 3*: Implementirati metodu ***cirkularnaPutanja()*** u klasi ***State.cs.*** Ideja jeste da prođete kroz celu parent hijerarhiju tekućeg stanja i da proverite da li smo tekuće stanje već posetili u ovoj grani pretraživanja. Na taj način ćemo sprečiti prolazak kroz cikluse u grafu i potencijalne mrtve petlje.
6. *TODO 4.1 i 4.2*: Proširiti DisplayPanel tako da sadrži još jednu kutiju proizvoljne boje (npr: Color.MediumBlue), promene izvršiti u metodama **onPaint(...)** i **onMouseDown(...)**.

*Napomena:* nakon izvršenog dodavanja algoritam treba da prvo „pokupi“ kutiju pa tek onda da nastavi do rešenja.

1. *TODO 5.1 i 5.2*: Implementirati A\* pretragu i heurističku funkciju koju će ta pretraga koristiti.
2. *TODO 6*: Nakon dodavanja nove kutije, proširiti klasu **State.cs** sa promenljivom **kutija(boolean)** čija vrednost zavisi od toga da li trenutno stanje sadrži kutiju.  *Napomena:* Sve ostale detalje koje treba da dodate nakon proširivanja klase State ćete dobiti od predmetnog asistenta na vežbama. Implementirati klasu **ADepthSearch.cs** koja kombinuje algoritme Prvi u dubinu i A\*. Pretraga počinje algoritmom Prvi u dubinu koji vrši pretragu od početnog stanja pa dok ne pronađe kutiju, nakon pronalaska pretragu nastavlja algoritam A\* od kutije pa do cilja. Kombinovati rešenja ta dva algoritma i ispisati ih na glavnoj formi.  
   *Napomena:* Moguće je iskoristiti i prilagoditi algoritme Prvi u dubinu i A\* koji su dati u ovom rešenju.
3. *TODO 7:* Dopuniti metodu **ADepth\_Click(...)** pozivima odgovarajućih metoda iz klase **ADepthSearch.cs** koje su napisane u prethodnom koraku.
4. *DODATNO:* Optimizovati metodu da proveru cirkularnosti u grafu.
5. *DODATNO:* Implementirati MinMax pretragu sa protivnikom. Pretraga je već pozvana kroz GUI i ideja jeste kao kod A\* Game mod-a, implementirati logiku bota koji pokušava da uhvati robotića.