**IZVEŠTAJ**

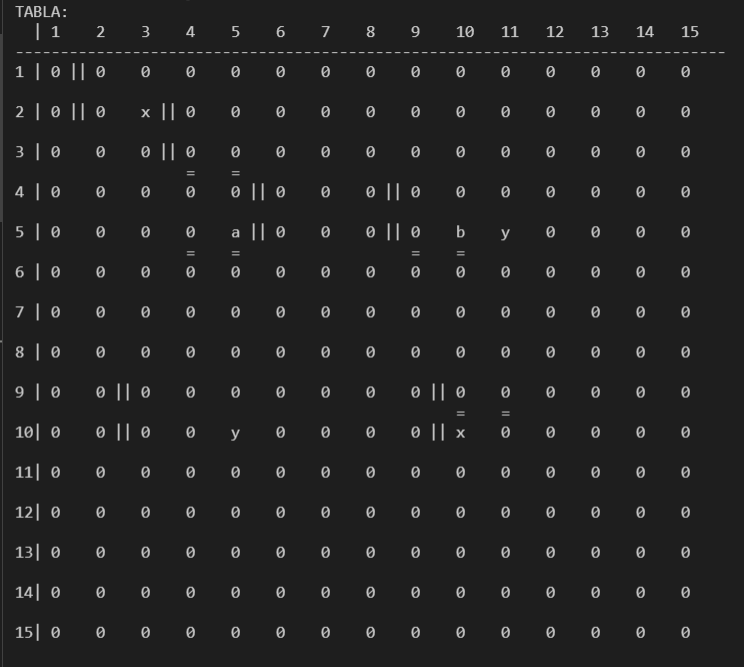
**Projekat:** Blockade

**Faza:** III

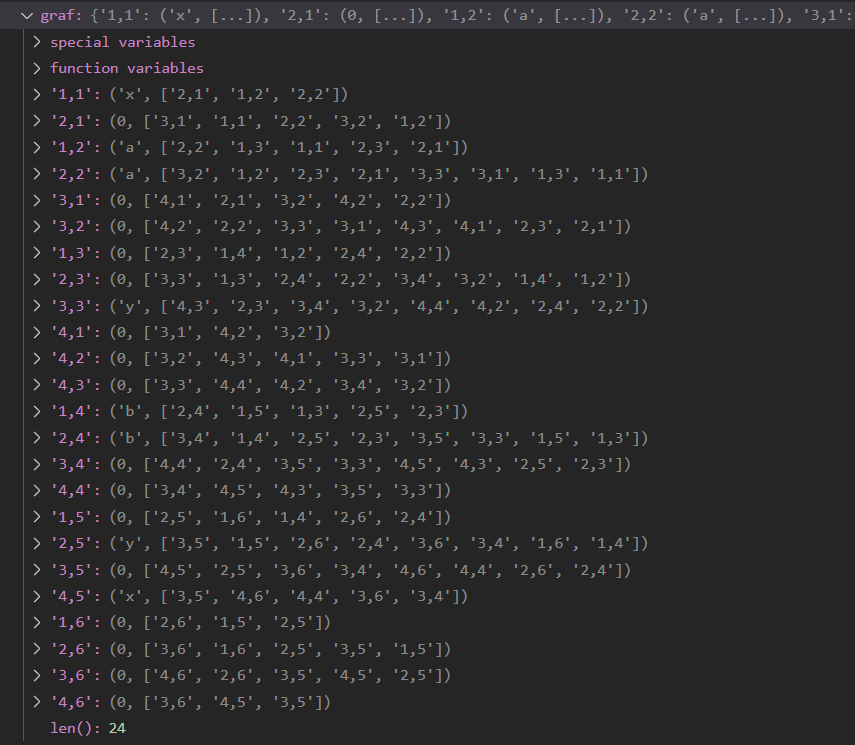
**Tim:** FVZTeam

**Studenti**:

* Veljko Mijović Ušković - 17224
* Željko Miloradović - 17254
* Filip Trajković - 17503

**Konzolni prikaz aplikacije:**

**Prikaz grafa u memoriji:**



Napomena: Velicina grafa je setovana na (4,6) zbog lakseg prikaza na slici. Aplikacija podrzava maksimalni broj preporucen u dokumentaciji projekta.

**Metode koje se koriste u prvoj fazi:**

* def generisiGraf(m, n, nizZidova, listaPoz, pobedaX, pobedaY)
* def pomocnoBrisanje(a, b, c, d, listaZidova, graf)
* def obrisi(listaZidova, graf)
* def SetujPocetnoStanje(velicinaX, velicinaY, listaIgraca, pobedaX, pobedaY)

**Metode koje se koriste u drugoj fazi:**

* def NotClosedPath(pobedaPozicije,graph)
* def gameLoop()
* def validacijaPokreta(graf, trenutno, ciljno, endpoz, startPoz)
* def pomeriIGraca(graf, m, n, startPoz, endPoz, naPotezu, pobeda, px, py)
* def proveriPobednika(pobedaPravilnoTuple)
* def unesiZidove(graf, listaZidova, m, n)
* def stampajGraf(graf, M, N)
* def generisiSvaMogucaStanja(graf, m, n, startPoz, naPotezu, pobeda, px1, px2, py1, py2)
* def generisiNovoStanjeZaUlazniPotez(graf, m, n, startPoz, endPoz, naPotezu, pobeda, px1, px2, py1, py2)

**Metode koje se koriste u trećoj fazi:**

* def pozicijePijuna(graph)
* def minimax(graph, depth, alpha, beta, maximizing\_player, naPotezu, m, n, pobeda, px1, px2, py1, py2)
* def ProveriDaLiPostojiVeza(graf, listaZidova, m, n)
* def generisiLogicnaStanjaZaUlazniPotezPijuna( graf, m, n, startPoz, endPoz, naPotezu, pobeda, px1, px2, py1, py2, BrojZidova)
* def generisiSvaLogicnaStanja(graf, m, n, startPoz, naPotezu, pobeda, px1, px2, py1, py2, BrojZidova)
* def pozicijePobednickihPolja()
* def proveriPrelazakPrekoPobednickePozicije(graf, pobedaA1, pobedaA2, pobedaB1, pobedaB2)
* def VratiBrojKorakaDoCilja(graf, m, n, naPotezu, pobedaA1, pobedaA2, pobedaB1, pobedaB2)
* def heuristika(graf, m, n, naPotezu, pobedaA1, pobedaA2, pobedaB1, pobedaB2)
* def bestSearch(graf, m, n, start, goal)
* def heuristikaZaTrazenjeputa(tacka1, tacka2)

***OPISI FUNKCIJA KOJE SE KORISTE U PROJEKTU:***

1. **generišiGraf**

Funkcija generišiGraf treba da na osnovu ulaznih parametara generiše proizvoljno stanje na tabli, odnosno graf koji predstavljastanje na tabli.  
Parameti ove funkcije su:

* m, n – veličina table u poljima
* nizZidova – niz pozicija izmedju kojih se nalaze zidovi
* listaPoz – pozicije igrača
* pobedaA1,A2,B1,B2 – pozicije pobedničkih polja za igrače

Funkcija prvo inicijalizuje prazan graf i zatim izvršava pretragu po širini tog grafa gde u svakoj iteraciji dodaje čvorove susedne trenutnom čvoru, počev od čvora 1,1 do čvora m,n. Vodeći računa o tome da ne prekorači veličinu table. Zatim postavlja igrače na unete pozicije i odredjuje pobedničke pozicije za sve igrače.  
Poslednji deo funkcije je foreach petlja koja na osnovu ulazne lista zidova briše odgovarajuće veze u grafu an taj način ubacuje zidove u igru, čime kreira proizvoljnu poziciju u igri.

1. **unesiZidove**

Funkcija unesiZidove unosi vertikalne ili horizontalne zidove na osnovi zadatih parametra u graf. Parametri su:

* graf – koji se modifikuje
* listaZidova – zidovi koji se unose
* m – broj vrsta
* n – broj kolona

Funkcionalnost je radjena tako da se unose samo dva cvora izmedju kojih se postavlja zid, na osnovu toga zakljucujemo da li se radi o vertikalnom ili horizontalnom zidu. Primer: listaZidova=([’1,1’, ’2,1’]), na osnovu indeksa znamo da se radi o horizontalnom zidu i zbog toga stavljamo zid izmedju cvorova 1,1 i 2,1 takodje i za cvorove 2,1 i 2,2, sto znaci da unosimo samo gornji par cvorova kod vertikalnog zida ili levi par cvorova kod horizontalnih zidova. Proveravaju se i granicni slucajevi na ivicama table nije moguce postaviti zidove. Zatim se pozivaju funkcije za brisanje potega izmedju svih cvorova. Pre svega na pocetku se proverava da ne dolazi do ukrstanja horizontalnih i vertikalnih zidova.

1. **pomocnoBrisanje**

Funkcija pomocnoBrisanje sluzi da poboljsa preglednost koda, naime prosledjuju joj se parametri koji oznacavaju izmedju kojih sledecih cvorova treba da se obrise poteg. Parametri su:

* a, b, c, d – odredjuju u kom pravcu se pomera cvor
* listaZidova – par cvorova izmedju kojih se brisu potege
* graf – koji se modifikuje

1. **obrisi**

Funkcija obrisi obilazi graf i brise potege koristeci „update“ princip, obilazi oba prosledjena cvora i brise poteg ka svom paru. Parametri su:

* listaZidova – par cvorova izmedju kojih se brisu potege
* graf – koji se modifikuje

1. **SetujPočetnoStanje**

Ova funkcije služi za generisanje inicijalnog stanja table na početku igre. Njeni prosledjeni parametri su:

* veličina table za x i y
* startne pozicije igrača
* pozicije za pobedu

Funkcija poziva funkciju generisiGraf sa odgovarajućim parametrima kako bi se dobio graf definisan na odgovarajući način bez zidova, i kao povratnu vrednost vraća taj startni graf.

1. **pomeriIgrača**

Ova funkcija pomera igrača sa odgredjene pozicije na drugu poziciju, pri tom pomeranju vodi računa da je pomeranje igrača pravilno, odnosnoda se igrač može pomeriti na datu poziciju i da izmedju njegove početne i krajnje pozicije nema zidova. Parametri ove funkcije su:

* graf koji predstvlja trenutno stanje igre
* veličinu table (m,n)
* startnu poziciju igrača i krajnju poziciju igrača
* podatak o tome koji igrač je na potezu
* polja pobede igrača

Kao povratnu vrednost funkcija vraća tuple (pobeda, ValidiranPokret) gde je pobeda bool podatak o tome da li je tim pomeranjem doslo do kraja igre, a ValdiranPokret je bool vrednost koja je True ukoliko je pokret dozovljen i uspešno izvršen.

Fukcija prvo proverava da li je na početnom polju pomeranja igrač koji je trenutno na potezu, ukoliko jeste, proverava se da li kretanje u opsegu ivica table, zatim validira ostala pravila kretanja pozivanjem funkcije validacijaPokreta. Ukoliko je ustanovljeno da je kretanje pravilno, proverava da li tim kretanjem neki od igrača dolazi do svoje pobedničke pozicije, i zavisno od toga setuje povratni parametar pobeda. Zatim, vrši samo pomeranje igrače po čvorovima grafa i vraća tuple (pobeda, ValidiranPokret).

1. **validacijaPokreta**

Ova funkcija vrši proveru mogućeg poteza pomeranja figure na tabli. Poziva se iz funkcije “pomeriIgraca” i proverava da li je moguće odigrati prosledjeni potez. Važno je napomenuti da se ova funkcija poziva isključivo ako je prosleđena krajnja pozicija realna u odnosu na veličinu table.

Provera 1: Funkcija validacije pokreta proverava da li na krajnjem polju već postoji neki igrač. Ukoliko je ciljno polje zauzeto nekom od figura, povratna vrednost je “False”.

Provera 2: Vrši se provera po pravilima partije po svim susednim poljima po širini I visini u pravcu prema ciljnoj poziciji. Ukoliko susedno polje trenutnog polja, u svojoj listi suseda sadrži ciljno polje, povratna vrednost je “True”.

Provera 3: Vrši se provera po pravilima partije po svim susednim poljima po dijagonalama. Ukoliko je , ciljno polje jedno od susednih polja, povratna vrednost je “True”.

Ukoliko Provera 2 I Provera 3 nisu uspešne, prikazuje se poruka o grešci I povratna vrednost funkcije je “False”.

Parametri ove funkcije su:

* graf koji predstvlja trenutno stanje igre
* trenutna pozicija igrača predstavljena u brojnom formatu
* ciljna pozicija igrača predstavljena u brojnom formatu
* ciljna pozicija igrača predstavljena u tekstualnom formatu
* trenutna pozicija igrača predstavljena tekstualnom formatu

1. **stampajGraf**

Funkcija stampajGraf se koristi da prikaže trenutno stanje igre u određenom trenutku. U ovom trenutnom obliku na kraju Faze 1, koristi se za konzolni prikaz stanja igre. Na početku funkcije se štampaju brojevi kolona I horizontalni okvir. Nakon toga, štampaju se sva polja po vrstama gde se na početku svake vrste ispisuje redni broj te vrste I vertikalni okvir. Prilikom štampanja se obilazi graf po vrstama, nadovezuje se vrednost koju ima svako polje u toj vrsti I mogući horizontalni zid, ukoliko postoji između trenutnog polja I njegovog suseda sa desne strane. Nakon svake vrste, štampaju se horizontalni zidovi koji predstavljaju zidove između određenog polja vrste I njegovog suseda u narednoj vrsti.

**Napomena**: Prilikom štampanja se void računa o graničnim slučajevima kada se štampa poslednja vrsta, odnosno poslednja kolona. Zbog formatiranja vrsta I kolona sa dvocifrenim brojevima, na određenim mestima se koristi veći ili manji broj razmaka.

Parametri ove funkcije su:

* graf koji predstvlja trenutno stanje igre
* M – Broj vrsta prosledjenog grafa
* N – Broj kolona prosledjenog grafa

1. **gameLoop**

Funkcija gameLoop je glavna funkcija koja se poziva prilikom pokretanja aplikacije. U njoj se od korisnika traži da unese neophodne parametre, kao što su veličina table I broj mogućih zidova koje igrači mogu da koriste. Nakon ovog inicijalnog unosa, prikazuje se početno stanje table. Zatim kreće beskonačna petlja koja se prekida kada jedan od igrača pobedi. U ovoj petlji se od igrača traži unos polja za igranje igre. Prvo se traži unos polja na kojem se nalazi figura koju igrač želi da pomeri. Zatim se traži krajnje polje na koje igrač želi da tu figure pomeri. Traže se lokacije zidova na kojim igrač želi da postavi zidove. Posle tog unosa, proverava se validnost unetih parametara, pozivaju funkcije za pomeranje igrača I postavljanje zidova. Ukoliko je nastala greška prilikom unosa pozicije ili uneta pozicija nije validna, igraču se prikazuje poruka o grešci I biće mu traženo da ponovo unese parametre sve dok ne unese pravilno. Kada unese pravilno, na potezu će biti drugi igrač. Ukoliko u igri dobijemo pobednika, izlazi se iz beskonačne petlje I prikazuje poruka o pobedniku.

1. **proveriPobednika**

Funkcija koja prima ulazni parameter koji je tipa “tuple”. Iz ovog parametra se čita podatak na lokaciji [0] I vraća vrednost tog parametra. Ovaj vraćeni parametar predstavlja bool promenljivu o podatku da li je na tabli odigran potez koji vodi u pobedničko stanje igrača koji je na potezu.

Parametar ove funkcije je:

* tuple podatak koji u sebi nosi podatak o pobedi

1. **NotClosedPath**

NotCLosedPath funkcija proverava da li postoji bar jedan put do svakog od ciljeva u igri. Kao argument funkcije uzima se touple pobeda pozicije i trenutno stanje grafa.

Pocevsi od jednog od ciljeva vrši se obilazak grafa po širini i cuva se skup obidjenih čvorova dok god se ne obidju svi čvorovi koje je moguće obići.

Ukoliko se u oblisku graf došlo do svih mogućih igrača znači da je uslov zadovoljen za taj cilj, a ukoliko nije vraca se vrednost false, odnosno signalizira se da pravila igre nisu zadovoljena.

Postupak se ponavlja za svaki cilj u grafu, s tim što je uvedena optimizacija da ukoliko se počeši od jednog cilja dodje do nekog drugog. obrada za taj cilj se ne izvršava s obzirom da je rezultat jednak obradi prvog cilja.

Parametri ove funkcije su:

* pobedaPozicije - tuple svih pobedničkih pozicija
* graph – stanje table u trenutku poziva funkcije

1. **generisiSvaMogucaStanja**

generisiSvaMogucaStanja je funkcija koja na osnovu unetih parametara vrši generisanje svih mogućih stanja igre. Ovo generisanje se vrši tako što se na osnovu unete startne pozicije koja predstavlja poziciju pijuna igrača koji je naPotezu. Ukoliko se na startnoj poziciji ne nalazi pijun igrača koji je na potezu, prikazuje se poruka o grešci i vraća se prazna lista. U daljem regularnom toku funkcije, se po pravilima igre za svako moguće naredno polje pijuna sa startne pozicije poziva metoda „generisiNovoStanjeZaUlazniPotez“. Ova funkcija vraća novo stanje ukoliko je potez validan. Ukoliko je potez validan, u listuStanja ćemo upisati novo-kreirano stanje. Povratna vrednost funkcije je popunjena listaStanja.

Parametri ove funkcije su:

* graf
* M
* N
* startPoz
* naPotezu
* pobeda
* px1
* px2
* py1
* py2

1. **generisiNovoStanjeZaUlazniPotez**

generisiNovoStanjeZaUlazniPotez je funkcija koja funkcioniše po principu funkcije PomeriIgraca. Na početku funkcije se kreira noviGraf od postojećeg prosledjenog u parametrima. Izvršava se validacija unetih parametara, izvršava se validacija pokreta istoimenom funkcijom I ukoliko je pokret validan, stanje novog Grafa se ažurira I prosledjuje kao povratna vrednost. Ostali povratni parametri su prisutni zbog daljeg korišćenja ovih funkcija u narednim fazama projekta. Povratna vrednost “pobeda” signalizira da li je igrač na pobedničkom mestu ili

ne, dok ValidiranPokret parameter služi zbog dodele novokreiranog stanja u listuStanja metode “generisiSvaMogucaStanja”.

Parametri ove funkcije su:

* graf
* M
* N
* startPoz
* endPoz
* naPotezu
* pobeda
* px1
* px2
* py1
* py2

1. **pozicijePijuna**

Funkcija pozicijePijuna je funkcija koja vrši pretragu prosleđenog grafa kao ulaznog parametra. Funkcija na početku poseduje praznu listu u koju se kasnije dodaju pozicije igrača na tabli. Pretraga se vrši po svim “key” vrednostima prosleđenog grafa i traže se pozicije pijuna u grafu. Ukoliko je na određenoj poziciji pronađen jedan od igrača “x” ili “y” u postojeću listu se dodaje novi element. Ukoliko je na određenoj poziciji pronadjen igrač “x”, njegova pozicija se u listu dodaje na početku. Ukoliko je na određenoj poziciji pronađen igrač “y”, ta pozicija se dodaje na kraju liste. Na ovaj način kao povratnu vrednost imamo uređenu listu četiri elemenata od kojih su prva dva elementa pozicije “x” igrača, a treći I četvrti element su pozicije “y” igrača. Ovakvo uređenje je neophodno zbog minmax funkcije u kojoj se ova funkcija poziva. Povratna vrednost funkcije je uređena lista.

Parametar ove funkcije je:

* graf

1. **minMax**

Ova funkcija predstavlja glavu funkciju za generisanje novog poteza. Ova funkcija generiše najbolji mogući potez za ulazni graf, odnosno za ulazno stanje na tabli. Funkcija najpre ima graničnu vrednost za vrednost kada je dubina jednaka nuli, odnosno kada se u stablu svih stanja dođe do čvorova koji su listovi. Obzirom da još uvek nemamo heuristiku, za potrebe zadatka u ovom delu vraćamo random integer

vrednost između -100 i 100. U narednoj fazi ćemo na ovom mestu pozivati funkciju heuristike za evaluaciju trenutnog stanja na tabli. Ukoliko dubina nije jednaka nuli, izvršavanje se nastavlja. U nastavku se najpre pribavljaju sve pozicije pijuna na tabli. U zavisnosti od toga koji je igrač na potezu, poziva se metoda „generisiSvaMogucaStanja“ za oba pijuna određenog igrača. Sva moguća stanja se ubacuju u zajedničku listu „children“. Ova lista predstavlja sve potomke trenutnog stanja grafa. Dalje sledi regularan tok minMax funkcije sa alfa-beta odsecanjem koji se koristi i na računskim vežbama. Za svako moguće potencijalni stanje u listi „children“ se rekurzivno poziva minMax funkcija. Povratna vrednost ovog poziva se evaluira i upoređuje sa max\_eval ili min\_eval vrednošću u zavisnosti od toga koji je igrač na potezu. Takodje u zavisnosti od toga koji je igrač na potezu, vrši se upoređivanje sa alpha/beta vrednošću i vrši odsecanje ukoliko se ispuni određeni uslov. Povratna vrednost ove funkcije nakon obrade svih mogućih poteza je „best\_move“ – novo stanje koje predstavlja stanje na tabli sa odigranim najboljim potezom. Drugi parametar povratne vrednosti je vrednost heuristike najboljeg poteza.

Parametri ove funkcije su:

* graf
* dubina
* alpha
* beta
* maximizing\_player
* naPotezu
* M
* N
* pobeda
* px1
* px2
* py1
* py2

1. **ProveriDaLiPostojiVeza**

Funkcija proverava da li postoji veza izmedju data dva cvora koja su prosledjena u parametru 'listaZidova' u samom grafu svih cvorova.

Parametri:

* graf
* listaZidova
* m
* n

1. **generisiLogicnaStanjaZaUlazniPotezPijuna**

Funkcija generisiLogičnaStanjaZaUlazniPotezPijuna predstavlja redukciju funkcije generisiNovoStanjeZaUlazniPotez. Najpre proverava da li je uneti potez validan. Vrši se provera poteza I svih ostalih vrednosti. Ukoliko je potez validan, a stanje nije pobedničko, generišu se logični zidovi na osnovu ugrađene logike sistema.

Parametri**:**

* graf,
* m
* n
* startPoz
* endPoz
* naPotezu
* pobeda
* px1
* px2
* py1
* py2
* BrojZidova

1. **generisiSvaLogicnaStanja**

Funkcija vrši redukciju generisanja svih mogućih stanja na osnovu logike koju smo razvili. Ova funkcija poziva funkciju generisiLogicnaStanjaZaUlazniPotezPijuna i za ulazni potez generiše moguća stanja.

Parametri:

* graf
* m
* n
* startPoz
* naPotezu
* pobeda
* px1
* px2
* py1
* py2
* BrojZidova

1. **pozicijePobednickihPolja**

Funkcija vraca pozicije pobednickih polja, na osnovu globalnih promenljivih M i N automatski generise pobednicke pozicije. Podrazumeva se da na osnovu velicine tabele M i N unapred znamo pobednicke pozicije za oba igraca.

1. **proveriPrelazakPrekoPobednickePozicije**

U slucaju da se protivnik pomeri sa pozicije na kome se nalazi vase pobednicko polje, nakon toga je potrebno opet uneti vrednost za pobednicko polje 'a' ili 'b' zbog tzv. overwrite vrednosti dok je 'y' ili 'x' bilo na toj poziciji u prethodnom krugu.

Parametri:

* graf
* pobedaA1
* pobedaA2
* pobedaB1
* pobedaB2

1. **VratiBrojKorakaDoCilja**

Ova funkcija vraca broj poteya potrebnih da odredjeni igrac stigne do svog najbližeg cilja.  
Ulazni parametri:

* graf - trenutno stanje table
* m, n – dimenzije table
* naPotezu – inforamciaj o tome koji igrač je na potezu
* pobedaA1
* pobedaA2
* PobedaB1
* pobedaB2 – pozicije pobede odgovarajučih igrača

Način rada:  
Pronalazi se pozicija igrača za kog se traži put. Zatim se poziva funkcije bestSearch koja vrača broj koraka I taj rezltat se vraća kao povratna vrednost funckije.

1. **heuristika**

Opis rada:  
Funckija inicjalizuje maksimalnu heuristicku vrednost na 500, pozivanjem funkcije bestSearch I sortiranjem rezultata pronalazi se najkraći put do cilja za igrača, ova vrednost opisuje koliko je potez doprineo približavanju igrača do pobede. Zatim se isti postupak vrši I za protivnika, da bi se saznalo koliko je postavljanje zida da odgovarajuće mesto doprinelo u udaljavanju protivnika od pobede. Konačna vrednost heuristike je vrednsot izmedju -100 I 100, gde je -100 pobeda protivnika a 100 pobeda igrača.

Ulazni parametri:

* graf – trenutno stanje table
* m, n – dimnezije stable
* naPotezu – inforamciaj o tome koji igrač je na potezu
* pobedaA1, pobedaA2, PobedaB1, pobedaB2 – pozicije pobede odgovarajučih igrača

1. **bestSearch**

Opis rada:

Funkcija implementria klasičan best first search gde je heuristika za pretragu udaljenost trenutnog stanja od ciljnog proračunata na osnovu indeksa pozicija na tabli. Povratna vrednost funkcije jeste dužina tog puta.

Ulazni parametra:  
graf- treutno stanej table,

m, n – dimenzije table,

start, goal – polazna I ciljna pozicija (čvor)

1. **heuristikaZaTrazenjeputa**

Opis rada:

Funkcija računa najkraću udaljenost izmedju dve tačke u grafu bez obzira na prepreke na putu na osnovu indeksa pozicija tih polja na tabli, odnosno dva polja na table I vraća tu vrednost kao povratnu vrednost funkcije

Ulazni parametri:

* tacka1 – početna tačka
* tacka2 – krajnja tacka