**Byte**

Dokumentacija za ceo kod igre byte

Vojin Minic 18278

Matija Savic 18365

**\*Napomena :** Za korisecenje programa neophodni su paketi **art** I **tabulate** ukoliko nisu instalirani na masini potrebno je to pribaviti naredbom u konzoli : pip install art ili pip install tabulate

**Dimension\_input\_validate**

Korisnik unosi dimenziju table,vrednost se proverava da li je odgovarajuca shodno pravilima igre ukoliko je uneta losa vrednost korisnik mora ponovo uneti novu pre nastavljanja dalje.

**Choose\_figure**

Igrac bira figuru koju zeli da igra,unosi vrednost W ili B zavisno da li zeli da igra White ili Black figuru.Voditi racuna da W igrac uvek igra prvi tako da ukoliko izaberete B racunar ce igrati prvi potez.

**Is\_game\_finished** I **silent\_is\_game\_finished**

Ova funkcija uzima niz [whiteStacks,blackStacks] I proverava da li je neki od igraca dosao do pobede.Funkcija se poziva nakon svakog poteza kako bi se utvrdilo kada je kraj igre.Silent verzija funkcije je potpuno ista osim sto nema print funcija kako bi konzola ostala cista kada ovu funkciju koristi racunar za proveru buducih poteza.

**CheckBoard** I **Silent\_checkBoard**

Funcija koja proverava stanje na table I ukoliko je neki stack pun sklanja ga sa table,I igracu cija je figura na vrhu se dodaje poen u nizu koji prati score.Na isti nacin silent funkcija ostaje bez print funkcija sa istim funkcionalnostima.

**ReadPosition**

Ispituje potez koji korisnik zeli da igra tako sto se prvo unosi broj vrste a zatim kolone tako da su vrednosti 1-based tj krecu od 1 a ne od 0.A je 1 ,B je 2 itd..

Format unosa je BrVrste BrKolone. Sa razmakom izmedju.Dole su navedena ogranicenja koja se proveravaju prilikom unosa.

Nakon unosenja pozicije unosi se mesto na stacku odakle se pomera stek,takodje je 1-based vrednost.Na kraju se unosi I smer po vrednostima 0,1,2,3 (GoreLevo,GoreDesno,DoleLevo,DoleDesno) sa respektivnim vrednostima.

1. Pozicija na tabli
   1. Ukoliko je pozicija nevalidna trazi se ponovo
   2. Ukoliko je na poziciji None trazi se ponovo
   3. Ukoliko je na toj poziciji jedna figura i ona ne odgovara trenutnom igracu na potezu trazi se ponovo
   4. Ukoliko na toj poziciji postoje vise figura ali ni jedna ne odgovara trenutnom igracu trazi se ponovo
2. Mesto na stacku
   1. Ustanovljeno je prethodnim da tu postoji figura koju mozemo pomerati,korisnik unosi mesto na stacku sve dok ne pogodi svoju figuru u suprotnom program ne daje da se igra protivnickom figurom(stackom)
3. Smer
   1. Smer se odmah proverava da li sa te pozicije se smer moze odigrati kao i da li je unos dobar

Ukoliko je svako ogranicenje ispunjeno readPosition vraca podatak oblika:

(Position\_i,Position\_j,Direction,Stack\_place) kao tuple.

**makeMove** I **tryMakeMove**

Funkcija koja potez koji je napravljen u funkciji **readPosition** I trenutno stanje table kako bi u start uzeli stek od stack\_position na dalje. Na slican nacin radi I tryMove samo bez print funkcije buduci da ova funkcija vraca False ukoliko taj potez nije dozvoljen moze se koristiti I da kasnije proveravamo da li je on moguc za to stanje table.

1. Sa startnog mesta uzimamo deo stacka koji pomeramo (ili ceo ako je stack\_place : 0) a ostatak ce ostati na tom mestu
2. Nalazimo sledece polje na osnovu direction na koje se pomeramo
3. Uvodimo odredjene provere:
   1. Velicina trenutnog steka i steka na koji se pomeramo ne sme preci 8
   2. Pozicija nase prve figure mora biti veca u odnosu na stek sa kojeg dolazi(na vecoj visini)
   3. Ukoliko se krece ka praznom polju mora se pomeriti ceo stek inace je potez nevalidan
4. Ukoliko su sva pravila ispostovana sa startnog mesta se skida deo ili ceo stek i prebacuje se na odedisni

**DrawTable**

Funkcija iscrtava tablu na osnovu podataka koji su prosledjeni u argumentu u vidu matrice kao i vrednosti n koja predstavlja duzinu i sirinu matrice.

Startletter sadrzi slovo A koje se inkrementira u ascii obliku i time dobijamo n sukcesivnih slova koje sluze kao indeksi kolona,slicno vazi i za brojeve koji se ispisuju kao indeksi vrste osim za 0-tu gde je ’/’

Ukoliko je vrednost matrice None znaci da se tu ne nalazi polje na kojem se moze igrati tj. Nije crno dijagonalno polje zbog toga cemo ga ostaviti prazno u tabeli.Ovo vazi za sva polja osim prvog i poslednjeg gde smo i pored None vrednosti(prva i zadnja vrsta se ne pune figurama) dodali tacke koje cemo koristiti kao moguca polja za kretanje(deo if indX == 0 i if indX == n-1), u suprotnom None ostaje.

Else deo obuhvata vrednosti matrice koji sadrze vrednost stringa koji predstavlja niz **B**lack ili **W**hite figura kao niz u stacku.Ovaj string ispisujemo u tom polju tako sto ga prvo isecemo na trecine kojima dodajemo tacke i \n kako bi prikaz bio u 3 reda

**Set\_Start\_state**

Postavlja prosledjenu tablu u pocetno stanje igre,poziva se na pocetku igre.

**Fill\_board\_default**

Inicijalizuje nizove koji ce biti u table.

**checkDirection**

Funkcija koja je koriscena od strane **getMoves** kako bi proverili da li je dati smer validan tj da li se mozemo kretati u tom smeru I inkrementira cenu puta koja se koristi kod pretrage po dubini kako bi dobili smerove do najblizih stekova.

**getMoves**

Funkcija koja obilazi po dubini sve oko polja koje smo prosledili na nacin da belezi pravce sa najmanjom cenom do prvog stacka.Ogranicenje igre je takvo da se figura mora kretati u smeru najblizeg steka ili na sam stek ukoliko je neposredno pored.U funkciji pamtimo stanja oblika [pos\_i,pos\_j,cena] kako bi kasnije ako naidjemo na mesto sa istom cenom dodali I taj put u listu.Smerovi koji imaju manju cenu od prethodne kreiraju novi niz sa tom cenom,ukoliko imaju istu cenu dodaju se u vec kreiranu listu sa tom cenom a ukoliko imaju vecu cenu taj put se prekida buduci da nam sve sto bi naislo nakon toga bilo bespotrebno.Obilazak se vrsi sve dok se ne dodje do nekog steka a kada je polje prazno onda se od tog polja siri algoritam u sva 4 smera.Rezultat funkcije je lista sa smerovima koji su dozvoljeni u notaciji koja je zabelezena u aplikaciji (0-3) smerovi gorelevo goredesno…

**getPlayerStates**

Funkcija koja za datu tablu I igraca daje za svako polje koji se smerovi mogu igrati.Vraca se u formatu (indx,indy,[nizDirekcija]) I ovu funkciju koristi getGameStates koja daje sve moguce poteze.

**getGameStates**

Funkcija koja uz pomoc **getPlayerTurns** pribavlja sve poteze za svako polje I kasnije proverava svaki od tih poteza nad svakim stekom unutar polja(sve mogucnosti moguce) I ubacuje sve potezi koji su zapravo moguci za trenutno stanje igre.

**getAdventage**

Proverava trenutno stanje igre I vraca ko je u prednosti

**Evaluation**

Funkcija koja za datu tabelu daje ocenu Koliko je to stanje dobro po igraca kojeg igra racunar.Unutar min-max se proverava prednost I prosledjuje se u ovu funkciju kao madePoint.Ukoliko je igrac postigao poen u ovom stanju njegova vrednost je skoro maksimalna,u suprotnom ukoliko dovodi do gubitka poena ovaj potez je minmalno ocenjen.Unutar for petlje se kroz svako polje prolazi I dodeljuju poeni sto je veci stek na polju kao I da li je na vrhu steka figura trenutnog igraca.Stanje ce biti vrednije ukoliko je vise figura na polju tako sto se broj figura stavlja na kub.

**minMax**

Minmax algoritam koji omogucava da racunar pronadje optimalan potez na odredjenoj dubini koja je navedena u startGame.Ovde takodje koristimo alpha beta odsecanje kako bi optimizovali pretrazivanje poteza.

LocalStacks pribavlja trenutan score igre kako bi ga prosledjivao dalje kroz stanja.Ovde proveravamo I da li u trenutnom stanju je neki od igraca postigao poen I to stavljamo u madePoint sto ce znaciti pri evaluaciji stanja.Ukoliko je dubina 0 ili je stanje u kraju igre onda se moze vrsiti evaluacija.

Ukoliko je maxPlayer = true znaci da gledamo max vrednost potomaka a ako je false onda suprotno.Pribavljamo svaki moguci potez za dato stanje table koriscenjem funkcije **getGameStates** I iteriramo kroz svaki od njih I rekurzivno zovemo minmax.Ukoliko vrednost evaluacije je veca od trenutno najvece onda se taj potez belezi kao optimalan I propagira nazad.Analogno max takodje se radi I min deo.

**StartGame**

Inicijalizujemo stacks (score) na 0:0,biramo figuru I pocetno stanje table se postavlja I prikazuje u konzoli.

DrawCounter sluzi kako bi predvideo da je doslo do neresenog rezultata tj ukoliko dva puta za redom igraci nemaju ni jedan potez u listi poteza.

Sve dok se igra ne zavrsi stampa se ko je na redu I ucitavaju se potezi(racunar preko minmax a korisnik preko readPosition.Ukoliko je taj potez u listi poteza on se odigrava I prelazi se na sledeci potez sve dok se ne utvrdi kraj igre.