Teorija igara i primjene

Lazarić, Antonio

Sveučilište Jurja Dobrile u Puli
Tehnički fakultet u Puli

Antonio Lazarić

Teorija igara i primjene

Završni rad

Pula, rujan, 2024. godine

Sveučilište Jurja Dobrile u Puli
Tehnički fakultet u Puli

Antonio Lazarić

Teorija igara i primjene

Završni rad

JMB:0303096828, redoviti student

Studijski smjer: Računarstvo

Predmet: Matematika 1, Matematika 2, Numerička matematika, Strukture podataka i algoritmi

Znanstveno područje: 1. Područje prirodnih znanosti, 2. Tehničke znanosti

Znanstveno polje: 1.01. Matematika, 2.09. Računarstvo

Znanstvena grana: 1.01.03 diskretna i kombinatorna matematika, 1.01.05 matematička logika i računarstvo, 2.09.03 obrada informacija

Mentor: Neven Grbac

Pula, rujan, 2024. godine

Tehnički fakultet u Puli

Ime i prezime studenta/ice Antonio Lazarić

JMBAG 0303096828

Status: ☒ redoviti ☐ izvanredni

PRIJAVA TEME ZAVRŠNOG RADA

Neven Grbac
Ime i prezime mentora

Računarstvo
Studij

Matematika 1, Matematika 2, Numerička matematika, Strukture podataka i algoritmi
Kolegij

Potvrđujem da sam prihvatio/la temu završnog/diplomskog rada pod naslovom:

Teorija igara i primjene

(na hrvatskom jeziku)

Game theory and applications

(na engleskom jeziku)

Datum: 14.4.2023.

Sadržaj

1. Uvod……..1
2. Teorija igara……..2
2.1. Definicija i povijest……... 2
2.2. Osnovni pojmovi teorije igara……..…… 2
2.3. Klasifikacija igara…… 3
3. Vrste igara…… 3
3.1. Broj igrača…… 3
3.2. Kooperativne i nekooperativne igre…….. 4
3.3. Igre s potpunom i nepotpunom informacijom……..… 4
3.4. Statičke i dinamičke igre…….…… 5
3.5. Igre s nultom sumom i igre s nenultom sumom…….…… 6
4. Primjeri igara……..……...6
4.1. Križić-kružić…….. 7
4.2. Zatvorenikova dilema..……... 7
4.3. Šah…….…… 8
4.4. Poker…….…… 8
4.5. Evolucijska igra……..…… 8
4.6. Aukcije……..…….. 9
5. Igra križić-kružić…….…… 9
5.1. Pravila igre……... 9
5.2. Strategije i analize……..…….. 10
5.3. Primjer igre potez po potez…….…… 11
5.4. Važnost igre križić-kružić u teoriji igara…… 13
6. Stablo igre križić-kružić……..…….. 13
6.1. Definicija i struktura stabla igre……..……14
6.2. Proces izgradnje stabla igre……..……..14
6.3. Primjeri stabala igre križić-kružić……..……..15
6.4. Primjena stabla igre u analizi igre križić-kružić…..…….. 19
6.5. Vizualizacija i reprezentacija stabla igre……..…… 20
6.6. Stablo igre u drugim igrama…….. 21
7. Zapis stabla igre u računalu……..…… 21

7.1. Struktura podataka za zapis stabla igre…… 21
7.2. Zapis stabla igre križić-kružić…… 22
7.3. Optimizacija zapisa stabla igre…….. 24
7.4. Primjena u stvarnim sustavima…….. 24
8. Algoritam Minimax……... 25
8.1. Osnovna ideja Minimax algoritma…….. 25
8.2. Koraci u Minimax algoritmu…… 26
8.3. Implementacija Minimax algoritma u igri križić-kružić…… 26
8.4. Optimizacija Minimax algoritma…….. 28
9. Detaljna objašnjenja koda za aplikaciju križić-kružić…….. 29
9.1. Uvod…… 29
9.2. Globalne varijable…… 29
9.3. Funkcije za logiku igre…… 30
10. Upute za korištenje aplikacije križić-kružić…… 36
10.1. Pokretanje aplikacije…… 36
10.2. Interakcija s GUI-jem…… 36
10.3. Pravila igre…… 37
10.4. Kako igrati…… 37
11. Zaključak…….. 39
11.1. Ključni aspekti projekta…… 39
11.2. Postignuti rezultati…… 40
12. Literatura…….. 41
13. Popis slika…… 43
14. Popis tablica…… 43
15. Sažetak……44
16. Summary…… 45

1. Uvod

Teorija igara je matematička disciplina koja proučava strategije donošenja odluka u situacijama gdje ishodi ovise o interakcijama među igračima. Analizirajući racionalne odluke pojedinaca ili grupa, teorija igara se koristi za modeliranje i optimizaciju ponašanja u raznim područjima, uključujući ekonomiju, politiku, biologiju i računalne znanosti.

Ovaj završni rad bavi se primjenom teorije igara na jednostavnu igru križić-kružić (eng. *Tic-Tac-Toe*). Iako igra naizgled djeluje jednostavno, ona pruža odličan okvir za analizu osnovnih principa teorije igara, poput strategije, ravnoteže i optimizacije.

Rad se sastoji od teorijskog i praktičnog dijela. U teorijskom dijelu istražit ćemo ključne pojmove i klasifikacije igara prema broju sudionika, dostupnim informacijama i dinamici igre. Praktični dio obuhvaća analizu igre križić-kružić kroz konstrukciju i analizu stabla igre, koje prikazuje sve moguće poteze od početnog do završnog stanja. Također ćemo razmotriti metode zapisivanja stabla igre u računalu i implementaciju algoritama za optimizaciju poteza, poput minimax algoritma.

Konačno, razvijena aplikacija za igru križić-kružić demonstrira kako teorija igara i analiza stabla igre mogu biti primijenjeni u razvoju računalnih rješenja. Cilj ovog rada je povezati teoriju i praksu, te pokazati važnost teorije igara u razvoju inteligentnih sustava.

2. Teorija igara

Teorija igara proučava donošenje odluka u situacijama gdje su ishodi tih odluka međusobno povezani i ovise o interakciji više sudionika ili "igrača" [3]. Ova disciplina analizira strategije koje racionalni igrači mogu koristiti kako bi ostvarili svoje ciljeve, uzimajući u obzir moguće poteze drugih igrača i njihove interese [1]. Ključna ideja teorije igara je modeliranje situacija konflikta i suradnje između pojedinaca, organizacija ili država, kako bi se pronašle optimalne strategije koje maksimiziraju korist za sudionike.

2.1. Definicija i povijest

Teorija igara potječe iz sredine 20. stoljeća, s radovima pionira kao što su John von Neumann i Oskar Morgenstern, koji su formalizirali koncept igre i razvili matematičke modele za analizu strateškog odlučivanja. U ekonomiji, teorija igara se koristi za analizu tržišta, pregovora i ponašanja potrošača; u političkim znanostima za proučavanje izbora, koalicija i međunarodnih odnosa; u biologiji za razumijevanje evolucije i ponašanja životinja; te u računalnim znanostima za razvoj algoritama i umjetne inteligencije [3].

2.2. Osnovni pojmovi teorije igara

Jedan od osnovnih pojmova u teoriji igara je igra, koja se definira kao situacija u kojoj dva ili više sudionika donose odluke koje utječu na ishod. Svaka igra se sastoji od igrača, njihovih mogućih poteza i ishoda koji ovise o kombinaciji tih poteza. Pored toga, važno je razumjeti koncept strategije. Strategija je plan ili niz poteza koje igrač može koristiti kako bi postigao svoj cilj. Postoje različite vrste strategija, uključujući dominantne strategije (one koje su najbolje bez obzira na poteze drugih igrača) i mješovite strategije (gdje igrači nasumično biraju između različitih poteza).

Još jedan ključan koncept je ravnoteža, posebno Nashova ravnoteža, nazvana po Johnu Nashu, koji je dokazao da u svakoj igri s konačnim brojem igrača i strategija postoji barem jedna takva ravnoteža. Nashova ravnoteža je stanje u kojem nijedan igrač ne može poboljšati svoj rezultat promjenom svoje strategije, pod pretpostavkom da svi ostali igrači ostanu pri svojim strategijama. Ova ravnoteža je centralna za teoriju igara jer omogućava predikciju ishoda u situacijama gdje svi igrači djeluju racionalno [3].

2.3. Klasifikacija igara

Teorija igara se dijeli na nekoliko grana, od kojih su najznačajnije kooperativne i nekooperativne igre. U kooperativnim igrama igrači mogu formirati koalicije i zajednički raditi kako bi ostvarili zajedničke ciljeve, dok u nekooperativnim igrama svaki igrač djeluje samostalno i u vlastitom interesu. Postoje i igre s potpunom informacijom, gdje su sve informacije dostupne svim igračima, te igre s nepotpunom informacijom, gdje neki aspekti igre ostaju nepoznati nekim igračima [3].

Pored toga, igre se mogu klasificirati prema tome donose li se odluke simultano ili sekvencijalno. U statičkim igrama svi igrači donose svoje odluke istovremeno, bez znanja o odlukama drugih igrača. U dinamičkim igrama, odluke se donose u sekvencama, gdje igrači mogu reagirati na poteze drugih igrača.

Razumijevanje teorije igara pruža duboke uvide u ponašanje racionalnih sudionika u različitim situacijama konflikta i suradnje. Ona omogućava ne samo analizu postojećih scenarija, već i dizajniranje novih sustava i pravila koji vode ka poželjnijim ishodima. U kontekstu ovog rada, teorija igara će biti korištena za analizu i optimizaciju strategija u igri križić-kružić, čime ćemo pokazati njenu primjenjivost čak i na naizgled jednostavne probleme.

3. Vrste igara

Klasifikacija igara u teoriji igara igra ključnu ulogu u razumijevanju različitih aspekata donošenja odluka i strategija koje igrači mogu koristiti. Igre se mogu razvrstati prema različitim kriterijima, a svaki od tih kriterija pruža drugačiji uvid u dinamiku igre i ponašanje sudionika. Ova raznolikost omogućava teoriji igara široku primjenu u različitim područjima, od ekonomije i politike do biologije i računalnih znanosti [1],[3].

3.1. Broj igrača

Jedan od najosnovnijih načina klasifikacije igara je prema broju sudionika ili igrača. U tom kontekstu, igre se mogu podijeliti na dvočlane igre, koje uključuju dva igrača, i n-člane igre, koje uključuju tri ili više igrača [3].

- Dvočlane igre: To su igre u kojima dva igrača donose odluke koje utječu na krajnji ishod igre. Ove igre mogu biti kompetitivne, gdje je interes jednog igrača

u sukobu s interesom drugog. Primjeri dvočlanih igara uključuju šah, dame i križić-kružić. U dvočlanim igrama, analize su često usmjerene na identifikaciju optimalnih strategija za svakog igrača, pri čemu se koriste koncepti kao što su minimax strategija i Nashova ravnoteža.

- n-člane igre: Ove igre uključuju više od dva igrača i mogu biti daleko složenije, jer svaki igrač mora uzeti u obzir strategije svih ostalih sudionika. Primjeri n-članih igara uključuju poker, monopol i mnoge društvene igre. U n-članim igrama, interakcije među igračima mogu rezultirati formiranjem koalicija ili saveza, što dodaje složenost u analizi i donošenju odluka .

3.2. Kooperativne i nekooperativne igre

Igre se također klasificiraju prema mogućnosti suradnje među igračima:

- Kooperativne igre: U kooperativnim igrama, igrači mogu formirati koalicije i zajedno raditi kako bi postigli zajedničke ciljeve ili maksimizirali ukupnu korist. Ove igre su usmjerene na analizu mogućnosti suradnje i načina raspodjele dobiti među igračima unutar koalicija. U teoriji kooperativnih igara, ključni pojmovi uključuju koalicijske strukture, core (skup isplata koji nijedan igrač ili grupa igrača ne može poboljšati), i Shapleyjeva vrijednost (metoda pravedne raspodjele dobiti među igračima). Primjeri kooperativnih igara uključuju pregovore između zemalja o trgovinskim sporazumima ili formiranje saveza u političkim kampanjama [3].
- Nekooperativne igre: U nekooperativnim igrama, svaki igrač djeluje samostalno i u vlastitom interesu, bez mogućnosti stvaranja obvezujućih dogovora s drugim igračima. Ove igre se fokusiraju na analizu individualnih strategija i načina na koji racionalni igrači donose odluke u konkurentnom okruženju. Nashova ravnoteža je centralni koncept u analizi nekooperativnih igara. Primjeri nekooperativnih igara uključuju šah, poker i križić-kružić [3].

3.3. Igre s potpunom i nepotpunom informacijom

Još jedan važan način klasifikacije igara odnosi se na vrstu informacija koje su dostupne igračima tijekom igre:

- Igre s potpunom informacijom: U igrama s potpunom informacijom, svi igrači imaju pristup svim relevantnim informacijama o stanju igre i potezima koje su

drugi igrači napravili. To znači da svaki igrač u svakom trenutku zna sve što je potrebno da donese racionalnu odluku. Primjeri igara s potpunom informacijom uključuju šah, dame i križić-kružić. U ovim igrama, igrači mogu koristiti retrospektivu i predviđanje kako bi planirali svoje strategije i anticipirali poteze protivnika[3].

- Igre s nepotpunom informacijom: U igrama s nepotpunom informacijom, neki aspekti igre ostaju nepoznati nekim ili svim igračima. To može uključivati nepoznavanje poteza drugih igrača, neizvjesnost oko stanja igre ili nepoznanice o strategijama koje drugi igrači koriste. Nepotpune informacije uvode dodatnu složenost u donošenje odluka, jer igrači moraju donositi odluke na temelju procjena i vjerojatnosti, umjesto na temelju potpunog znanja. Primjeri igara s nepotpunom informacijom uključuju poker, gdje igrači ne znaju karte koje drugi igrači drže [3].

3.4. Statičke i dinamičke igre

Dinamika igre odnosi se na redoslijed u kojem igrači donose odluke, a na temelju toga igre se mogu podijeliti na:

- Statičke igre: U statičkim igrama, svi igrači donose svoje odluke simultano, bez znanja o odlukama drugih igrača. Rezultati ovih odluka postaju poznati tek nakon što su sve odluke donesene. Statičke igre se često modeliraju pomoću matrice isplata, gdje svaki red predstavlja strategiju jednog igrača, a svaka kolona strategiju drugog igrača, pri čemu ćelije matrice sadrže isplate za oba igrača. Statičke igre su posebno pogodne za analizu u situacijama gdje nema vremenskog aspekta odlučivanja. Primjeri statičkih igara uključuju aukcije zatvorenog tipa i igre poput kamena-papir-škare [3].
- Dinamičke igre: U dinamičkim igrama, igrači donose odluke u sekvencama, gdje svaki igrač može reagirati na poteze koji su prethodno napravljeni. Ove igre uključuju vremenski aspekt, gdje redoslijed poteza igra važnu ulogu u strategiji. Dinamičke igre se često modeliraju pomoću stabala igara, gdje svaki čvor predstavlja stanje igre nakon određenog poteza, a grane predstavljaju moguće poteze. Dinamičke igre omogućavaju igračima da promatraju poteze svojih

protivnika i prilagođavaju svoje strategije u skladu s tim. Primjeri dinamičkih igara uključuju šah, gdje igrači naizmjenično donose odluke, i pregovore, gdje se strane međusobno nadopunjuju kroz seriju ponuda i kontraponuda [3].

3.5. Igre s nultom sumom i igre s nenultom sumom

Klasifikacija igara prema isplatama također je važna za razumijevanje prirode sukoba ili suradnje među igračima:

- Igre s nultom sumom: U igrama s nultom sumom, dobitak jednog igrača je jednak gubitku drugog igrača, tako da je ukupna suma isplata uvijek nula. Ove igre su suštinski kompetitivne jer svaki igrač pokušava maksimizirati svoj dobitak na račun protivnika. Šah je klasičan primjer igre s nultom sumom, gdje jedan igrač pobjeđuje, a drugi gubi. U ovakvim igrama, igrači su motivirani da pronađu strategije koje minimiziraju potencijalne gubitke, često putem minimax strategije [3].
- Igre s nenultom sumom: U igrama s nenultom sumom, moguće je da svi igrači ostvare dobitak ili gubitak, što znači da ukupna suma isplata može biti različita od nule. Ove igre omogućavaju suradnju među igračima i često su povezane s kooperativnim igrama, gdje igrači mogu formirati saveze i dogovarati se kako bi postigli zajedničke ciljeve. Primjeri uključuju trgovinske pregovore, gdje obje strane mogu profitirati od dogovora [3].

4. Primjeri igara

Primjeri igara u teoriji igara pružaju konkretne scenarije koji omogućuju razumijevanje i primjenu ključnih koncepata poput strategije, ravnoteže i optimizacije. Kroz ove primjere možemo analizirati različite vrste igara, njihove karakteristike i strategije koje igrači koriste u raznim situacijama. U ovom odlomku razmotrit ćemo nekoliko primjera igara koje ilustriraju različite aspekte teorije igara, uključujući jednostavne klasične igre, kompleksne društvene igre, igre s potpunom i nepotpunom informacijom, kao i igre s nultom i nenultom sumom.

4.1. Križić-kružić

Jedan od najpoznatijih i najjednostavnijih primjera igre s potpunom informacijom i igre s dva igrača je križić-kružić (eng. Tic-Tac-Toe). Igra se odvija na mreži 3x3, gdje dva igrača, "X" i "O", naizmjenično postavljaju svoje znakove u prazne ćelije mreže. Cilj svakog igrača je postaviti tri svoja znaka u nizu, bilo horizontalno, vertikalno ili dijagonalno, prije nego što to učini protivnik. Križić-kružić je primjer dinamičke igre, u kojoj svaki igrač može unaprijed izračunati sve moguće poteze i ishode. Unatoč svojoj jednostavnosti, igra pruža osnovu za razumijevanje ključnih koncepata teorije igara, poput optimalne strategije, ravnoteže i stabla igre. Primjerice, analiza svih mogućih stanja igre putem stabla igre pokazuje da, ako oba igrača igraju optimalno, igra uvijek završava neriješeno [8].

4.2. Zatvorenikova dilema

Zatvorenikova dilema klasičan je primjer nekooperativne igre s nenultom sumom, koja se često koristi za ilustraciju problema racionalnog odlučivanja i međusobnog povjerenja između sudionika. U ovoj igri, dva zatvorenika uhapšena su zbog zločina i odvojeno ispitivana. Svaki zatvorenik ima dvije opcije: priznati zločin i svjedočiti protiv drugog (suradnja s vlastima) ili šutjeti (suradnja s drugim zatvorenikom). Ishodi su sljedeći:

- Ako oba zatvorenika šute, dobit će blagu kaznu (npr. 1 godinu zatvora).
- Ako jedan zatvorenik svjedoči protiv drugog, dok drugi šuti, prvi će biti oslobođen, dok će drugi dobiti maksimalnu kaznu (npr. 10 godina zatvora).
- Ako oba zatvorenika svjedoče jedan protiv drugog, oba će dobiti umjerenu kaznu (npr. 5 godina zatvora).

Zatvorenikova dilema pokazuje kako individualno racionalno ponašanje može dovesti do lošijeg ishoda za oba sudionika u usporedbi s kooperativnim ponašanjem. U ovom slučaju, optimalna strategija prema teoriji igara (Nashova ravnoteža) je da oba zatvorenika svjedoče jedan protiv drugog, iako bi za oboje bilo bolje da šute. Ova igra koristi se za proučavanje problema u područjima poput ekonomije, politike i biologije, gdje pojedinci moraju donositi odluke u uvjetima sukoba interesa.

4.3. Šah

Šah je primjer kompleksne igre za dva igrača, dinamičke igre s potpunom informacijom. Igra se odvija na ploči 8x8, gdje svaki igrač kontrolira 16 figura (kralj, kraljica, topovi, lovci, konji i pješaci). Cilj igre je matirati protivničkog kralja, tj. dovesti ga u poziciju u kojoj je pod prijetnjom napada i ne može pobjeći. Šah je klasičan primjer igre koja zahtijeva strateško razmišljanje i predviđanje, jer svaki potez može imati dalekosežne posljedice na tijek igre. Stablo igre u šahu izuzetno je veliko, s brojem mogućih poteza koji eksponencijalno raste sa svakim potezom. Analiza šaha kroz teoriju igara uključuje istraživanje optimalnih strategija, korištenje algoritama za pretraživanje stabla igre i razumijevanje koncepta minimax strategije, koja pomaže u minimiziranju maksimalnog gubitka [3].

4.4. Poker

Poker je primjer dinamičke, nekooperativne igre s nepotpunom informacijom i nenultom sumom. U ovoj igri, igrači drže karte koje su skrivene od ostalih igrača i donose odluke na temelju djelomičnih informacija. Poker zahtijeva kombinaciju vještine, strategije i procjene vjerojatnosti, ali i psiholoških vještina, poput blefiranja i čitanja ponašanja protivnika. Ishodi u pokeru ovise ne samo o sreći (kartama koje igrači dobiju), već i o tome kako igrači koriste dostupne informacije da donesu optimalne odluke. U teoriji igara, poker se koristi za proučavanje strategija u uvjetima nesigurnosti i nepotpunih informacija, a ključni koncepti uključuju mješovite strategije, gdje igrači nasumično biraju između različitih poteza kako bi postali nepredvidivi protivnicima [3].

4.5. Evolucijska igra

Evolucijska igra primjer je igre koja se koristi u biologiji za proučavanje strategija preživljavanja i reprodukcije u prirodi. Za razliku od klasičnih igara, gdje igrači svjesno donose odluke, u evolucijskim igrama strategije su urođene i nasljedne, a uspjeh strategije mjeri se njezinim doprinosom u preživljavanju i reprodukciji. Jedan od najpoznatijih primjera evolucijske igre je igra sokol i golub, koja modelira interakciju između dvije strategije – agresivne (sokol) i miroljubive (golub) – u borbi za resurse. Ishodi igre ovise o proporciji sokola i golubova u populaciji i o troškovima i koristima sukoba. Evolucijske igre pomažu u razumijevanju dinamike prirodne selekcije i stabilnih strategija, koje se s vremenom ne mijenjaju jer su optimalne u danom okruženju [3].

4.6. Aukcije

Aukcije su primjer igre s nepotpunom informacijom i nenultom sumom, koja se koristi u ekonomiji. U aukcijama, sudionici licitiraju za određeni predmet, a pobjednik je onaj koji ponudi najvišu cijenu. Postoji više vrsta aukcija, uključujući engleske aukcije (gdje se cijena postepeno povećava dok ne ostane samo jedan licitator) i aukcije zatvorenog tipa (gdje svi licitatori istovremeno podnose svoje ponude bez znanja o ponudama drugih). Teorija igara koristi se za analizu strategija licitiranja i za optimizaciju formata aukcija kako bi se postigla maksimalna dobit. Sudionici aukcija moraju procijeniti vrijednost predmeta i ponašanje drugih licitatora kako bi donijeli optimalnu odluku, što čini ove igre izazovnima i zanimljivima za analizu.

Ovi primjeri pokrivaju širok spektar igara koje se analiziraju u teoriji igara, od jednostavnih igara poput križić-kružića do kompleksnih igara kao što su šah, poker i evolucijske igre. Svaki primjer ilustrira različite aspekte donošenja odluka, strateškog razmišljanja i interakcije među sudionicima, te pomaže u razumijevanju kako se teorija igara primjenjuje u različitim kontekstima. Kroz analizu ovih primjera moguće je dobiti dublji uvid u ponašanje racionalnih agenata u različitim situacijama, što je ključno za razvoj i primjenu teorije igara u stvarnim problemima [3].

5. Igra križić - kružić

Križić - kružić (engl. Tic-Tac-Toe) jedna je od najjednostavnijih i najpoznatijih igara za dva igrača, koja se često koristi kao uvod u osnovne koncepte teorije igara. Iako na prvi pogled izgleda trivijalno, ova igra pruža bogat teren za analizu strategije, donošenja odluka i optimizacije. U ovom odlomku detaljno ćemo razmotriti pravila igre, moguće strategije te njezinu važnost kao modela za razumijevanje osnovnih principa teorije igara [7],[8],[9].

5.1. Pravila igre

Križić - kružić se igra na mreži dimenzija 3x3 koja sadrži devet ćelija. Igra se odvija između dva igrača. Jedan igrač koristi znak "X", dok drugi koristi znak "O". Cilj igre je postaviti tri svoja znaka u niz, bilo horizontalno, vertikalno ili dijagonalno, prije nego što to učini protivnik [8].

Redoslijed igranja je sljedeći:

1. Igrači naizmjenično postavljaju svoj znak ("X" ili "O") u bilo koju praznu ćeliju mreže.
2. Prvi igrač koji uspije formirati liniju od tri znaka (horizontalno, vertikalno ili dijagonalno) pobjeđuje.
3. Ako su sve ćelije ispunjene, a nijedan igrač nije formirao liniju od tri znaka, igra završava neriješeno.

Primjer igre koja završava pobjedom za "X":

Tablica 1 - Primjer igre koja završava pobjedom za "X"

X	X	X
O	O	

Izvor: obrada podataka

U ovom slučaju, "X" je formirao liniju u gornjem redu i time osvojio igru [8].

5.2. Strategije i analize

Iako su pravila križić-kružića jednostavna, igra sadrži iznenađujuću složenost kada se analizira kroz prizmu teorije igara. Glavni aspekti analize uključuju:

- Optimalna strategija: U križić-kružiću, optimalna strategija osigurava da nijedan igrač ne može izgubiti ako igra savršeno. Igrači moraju predviđati poteze protivnika i birati najbolje poteze kako bi spriječili protivnika da formira niz od tri znaka, dok istovremeno pokušavaju formirati vlastiti niz. Na primjer, prvi potez je često ključan, jer postavljanje znaka "X" ili "O" u sredinu mreže daje igraču veće šanse za pobjedu. Ako protivnik pogriješi u sljedećim potezima, prvi igrač može iskoristiti tu grešku i pobijediti [7], [8].
- Neriješen ishod: Ako oba igrača igraju optimalno, križić-kružić će uvijek završiti neriješeno. Ovo čini igru interesantnom za analizu, jer se igrači moraju truditi da izbjegnu greške koje bi dovele do poraza, a istovremeno pokušavaju natjerati protivnika da pogriješi. Matematički gledano, postoji konačan broj mogućih

poteza i pozicija, što omogućava kompletnu analizu svih mogućih scenarija igre [8].

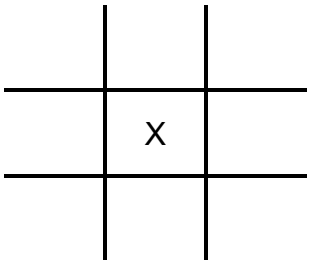
- Stablo igre: Križić-kružić se može predstaviti pomoću stabla igre, gdje svaki čvor predstavlja stanje igre nakon određenog poteza, a svaka grana predstavlja mogući potez. Analizom stabla igre moguće je mapirati sve moguće ishode igre i identificirati optimalne strategije. Ovo je osnovna metoda u teoriji igara za analizu igara s potpunom informacijom.
- Prvi potez: Biti prvi na potezu u križić-kružiću je značajna prednost, ali ne garantira pobjedu. Prvi igrač ima inicijativu i može odabrati središnju poziciju, što otvara više mogućnosti za formiranje niza od tri znaka. Drugi igrač mora pažljivo birati poteze kako bi neutralizirao ovu prednost i spriječio prvi niz [7], [8].

5.3. Primjer igre potez po potez

Pogledajmo primjer igre koja se odvija potez po potez između dva igrača, gdje prvi igrač koristi "X", a drugi "O":

1. **Potez 1**: Igrač "X" postavlja znak u sredinu ploče.

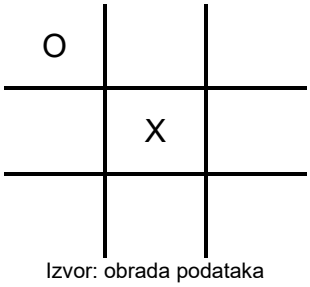
Tablica 2 - Potez 1 primjera igre križić-kružić



Izvor: obrada podataka

2. **Potez 2**: Igrač "O" postavlja svoj znak u gornji lijevi ugao.

Tablica 3 - Potez 2 primjera igre križić-kružić



Izvor: obrada podataka

3. **Potez 3**: Igrač "X" postavlja znak u donji lijevi ugao.

Tablica 4 - Potez 3 primjera igre križić-kružić

O		
	X	
X		

Izvor: obrada podataka

4. **Potez 4**: Igrač "O" postavlja znak u gornji desni ugao.

Tablica 5 - Potez 4 primjera igre križić-kružić

O		O
	X	
X		

Izvor: obrada podataka

5. **Potez 5**: Igrač "X" postavlja znak u donji desni ugao.

Tablica 6 - Potez 5 primjera igre križić-kružić

O		O
	X	
X		X

Izvor: obrada podataka

6. **Potez 6**: Igrač "O" postavlja znak u gornju sredinu.

Tablica 7 - Potez 6 primjera igre križić-kružić

O	O	O
	X	
X		X

Izvor: obrada podataka

Pobjeda: Igrač "O" pobjeđuje jer formira horizontalnu liniju na gornjoj liniji (O, O, O).

Ovaj primjer pokazuje kako se igra može razvijati, kao i kako igrači moraju pažljivo planirati svoje poteze kako bi ostvarili pobjedu ili spriječili protivnika da pobijedi.

5.4. Važnost igre križić - kružić u teoriji igara

Križić-kružić, iako jednostavna igra, ima ključnu ulogu u teoriji igara zbog svoje pristupačnosti i mogućnosti da se modelira kompletno stablo igre. Kroz analizu ove igre, učenici teorije igara mogu naučiti osnovne koncepte kao što su:

- Nashova ravnoteža: Situacija u kojoj nijedan igrač ne može poboljšati svoj rezultat promjenom svoje strategije, pod pretpostavkom da drugi igrač igra optimalno.
- Minimax strategija: Strategija koja minimizira maksimalni gubitak igrača, što je posebno korisno u igrama s nultom sumom [10], [11], [12].
- Optimizacija i donošenje odluka: Križić-kružić omogućava igračima da vježbaju donošenje odluka na temelju trenutne situacije na ploči, anticipirajući buduće poteze i ishode.

Ova igra se također koristi kao osnova za razvoj složenijih igara i algoritama u računalnim znanostima, gdje algoritmi za pretragu stabla igre, poput minimax algoritma, nalaze široku primjenu u razvoju umjetne inteligencije i automatiziranih sustava za donošenje odluka. U kontekstu edukacije, križić-kružić je često prva igra koju učenici programiraju kada uče osnove algoritama i umjetne inteligencije, jer omogućava jednostavnu, ali temeljitu analizu i implementaciju strategija.

6. Stablo igre križić-kružić

Stablo igre je grafički prikaz svih mogućih poteza i ishoda u nekoj igri. U teoriji igara, stablo igre je ključno za razumijevanje dinamike igre, jer omogućava analizu svakog mogućeg scenarija, identifikaciju optimalnih poteza i razvijanje strategija koje vode ka pobjedi ili najboljem mogućem ishodu. U igri križić-kružić, stablo igre pruža potpuni pregled svih mogućih stanja igre, od početka do kraja, i omogućava igračima da predvide sve moguće odgovore protivnika na svaki njihov potez. U ovom odlomku

detaljno ćemo analizirati što je stablo igre križić-kružić, kako se kreira, kako se koristi za analizu igre te koje sve igre mogu imati stablo igre [1], [5].

6.1. Definicija i struktura stabla igre

Stablo igre križić-kružić predstavlja sve moguće sekvence poteza koje igrači mogu napraviti, od početnog stanja (prazne ploče) do konačnih stanja igre (pobjeda, poraz ili neriješeno). Svako stanje igre prikazuje se kao čvor u stablu, dok se svaki mogući potez prikazuje kao grana koja povezuje dva čvora. Korijen stabla predstavlja početno stanje igre, a listovi stabla predstavljaju konačna stanja igre. Svaka putanja od korijena do lista odgovara jednom mogućem razvoju igre.

Stablo igre križić-kružić je konačno, jer postoji ograničen broj mogućih poteza i stanja. S obzirom na to da se igra odvija na mreži 3x3 sa samo devet pozicija, ukupni broj mogućih poteza je ograničen. Ipak, broj mogućih sekvenci poteza je značajan, posebno ako uzmemo u obzir sve moguće varijacije redoslijeda poteza i ishoda [2],[6].

6.2. Proces izgradnje stabla igre

Izgradnja stabla igre križić-kružić započinje s početnim stanjem, tj. praznom mrežom 3x3. Svaki potez igrača dodaje novi čvor u stablo. Nakon svakog poteza, igra prelazi u novo stanje koje postaje početna točka za sljedeći potez. Ovaj proces se ponavlja sve dok igra ne dođe do konačnog stanja – pobjede, poraza ili neriješenog rezultata.

Evo osnovnih koraka u izgradnji stabla igre:

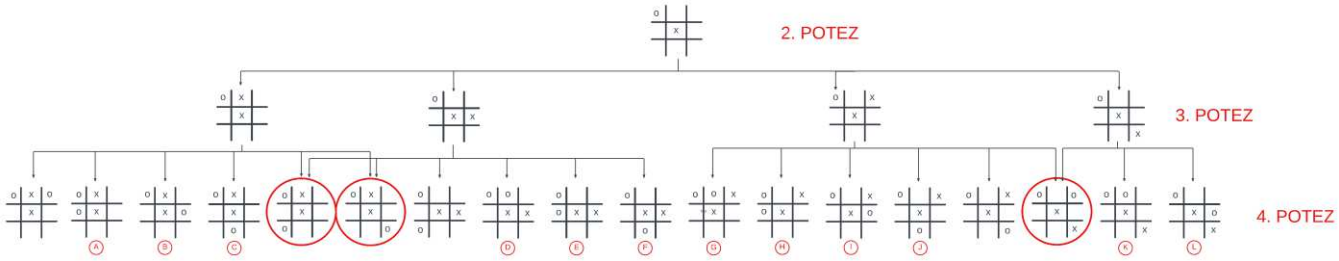
1. Početno stanje: Stablo igre počinje s praznom pločom kao korijenom stabla.
2. Prvi potez: Prvi igrač (obično "X") može postaviti svoj znak na bilo koju od devet pozicija. To stvara devet mogućih stanja igre, što znači devet grana koje izlaze iz korijena stabla.
3. Drugi potez: Nakon što je "X" postavljen, drugi igrač ("O") može birati između osam preostalih praznih pozicija. To rezultira osam novih grana koje izlaze iz svakog prethodnog čvora.
4. Nastavak igre: Ovaj proces se ponavlja za svaki sljedeći potez, pri čemu se broj mogućih grana smanjuje kako se sve više pozicija na ploči popunjava.

5. Konačno stanje: Kada jedan igrač formira niz od tri znaka ili se sve pozicije na ploči popune, igra dolazi do kraja, a odgovarajući čvor u stablu postaje list, označavajući konačno stanje igre.

Ukupni broj čvorova u stablu igre je značajan, ali konačan, zbog ograničenog broja pozicija na ploči. Na primjer, ukupan broj mogućih stanja igre (računajući sve poteze i permutacije) je 255.168 ali nakon uklanjanja simetričnih i redundantnih stanja, taj broj se smanjuje na 765 različitih stanja. Stablo igre omogućava vizualizaciju svih ovih stanja i analizu svih mogućih putanja koje igra može slijediti [1], [13].

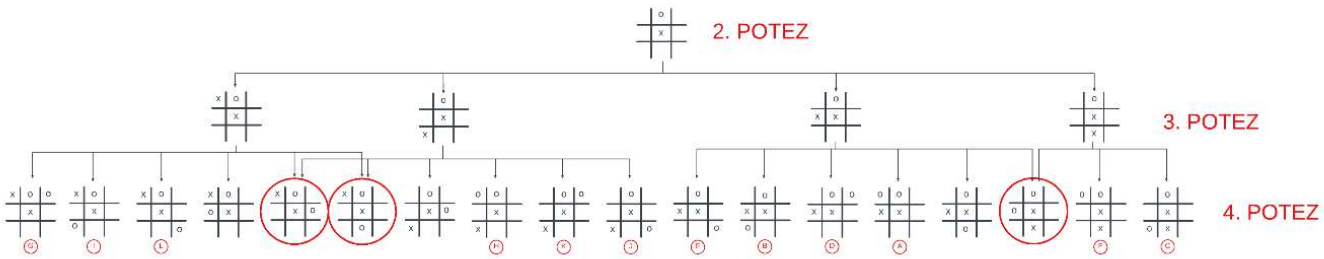
6.3. Primjeri stabala igre križić–kružić

Slika 1 - 1. Primjer stabla igre križić-kružić: simbol "O" je stavljen u kut



Izvor: obrada podataka

Slika 2 - 2. Primjer stabla igre križić-kružić: simbol "O" nije stavljen u kut



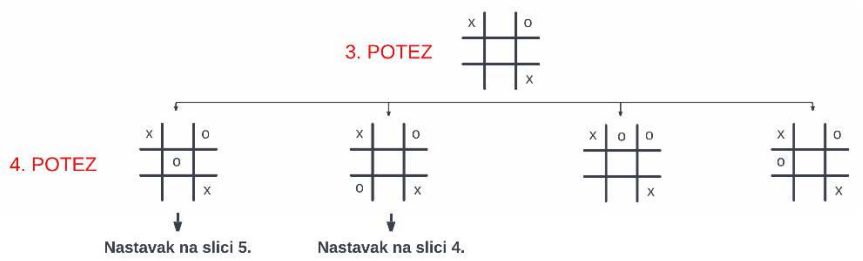
Izvor: obrada podataka

Na prvoj slici, igrač koji igra s "O" postavio je svoj simbol u kut. Analizom poteza dolazimo do situacije u kojoj već na 4. potezu postoje simetrične pozicije koje su nastale u različitim granama stabla i označene su crvenim slovima. Drugim riječima, iako su različite sekvence poteza dovele do tih pozicija, krajnje pozicije su identične. Ova pojava simetrije jasno je označena crvenim krugovima, gdje su te pozicije iste, iako su rezultat različitih nizova poteza.

Na drugoj slici, "O" nije postavljen u kut, ali i dalje se može uočiti ista pojava simetričnih pozicija. Naime, na 4. potezu, različiti nizovi poteza ponovno dovode do identičnih krajnjih pozicija. Potezi koji su se odigrali na različitim dijelovima ploče rezultiraju istim završnim ishodima, što ponovno dokazuje simetriju unutar igre.

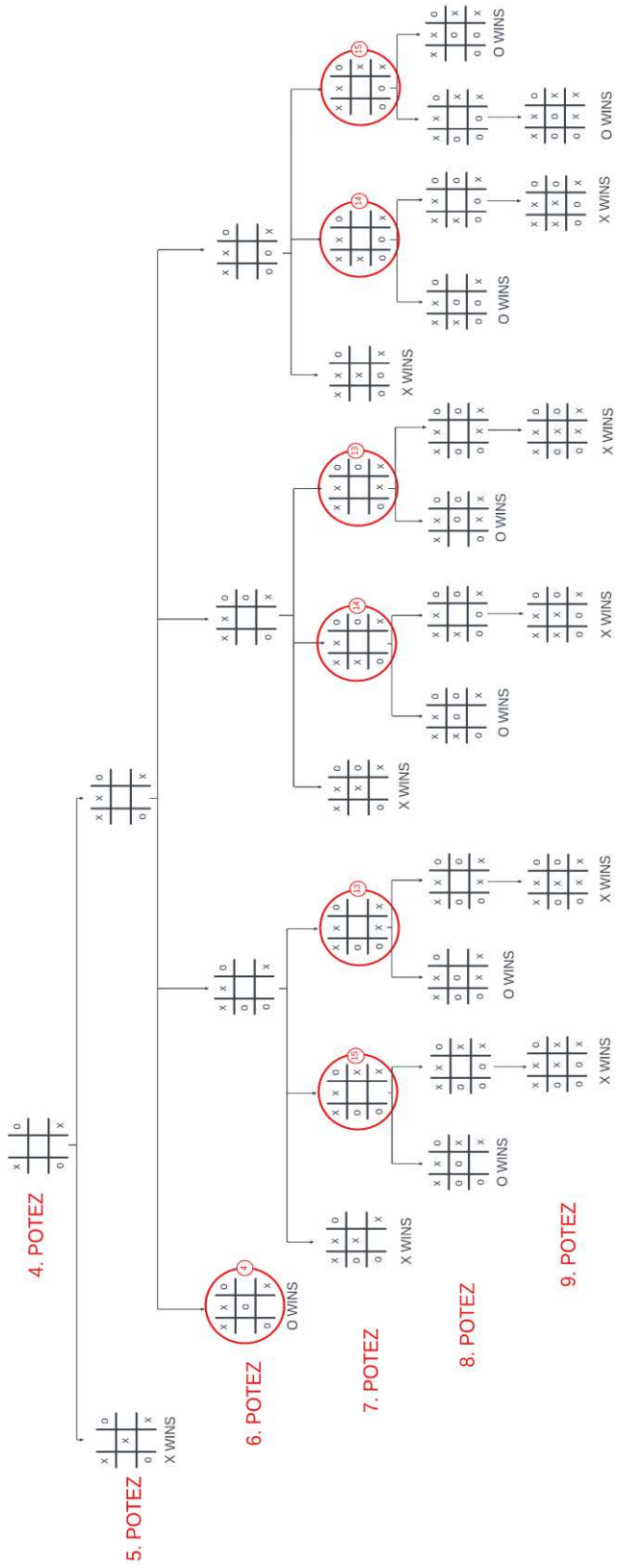
Ove slike jasno pokazuju kako različiti nizovi poteza mogu dovesti do istih ishoda, što ukazuje na visoku simetričnost igre križić-kružić. Zbog te simetrije, analiza igre se može pojednostaviti, jer se mnoge situacije mogu smatrati ekvivalentnima, iako su ostvarene kroz različite sekvence poteza. Bez obzira na to je li "O" postavljen u kut ili nije, na 4. potezu dolazimo do istih pozicija kroz različite grane stabla, što potvrđuje tvrdnju da igra ima simetričnu strukturu.

Slika 3 - 3. Primjer stabla igre križić-kružić



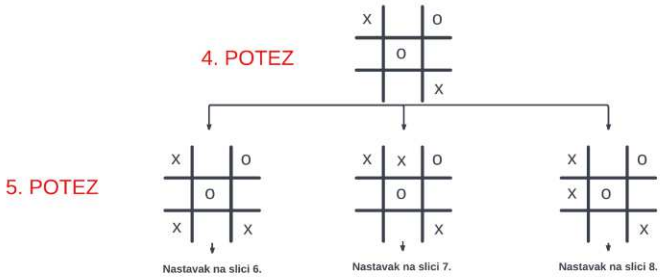
Izvor: obrada podataka

Slika 4 - 3. Primjer stabla igre križić-kružić (nastavak sa slike 3)



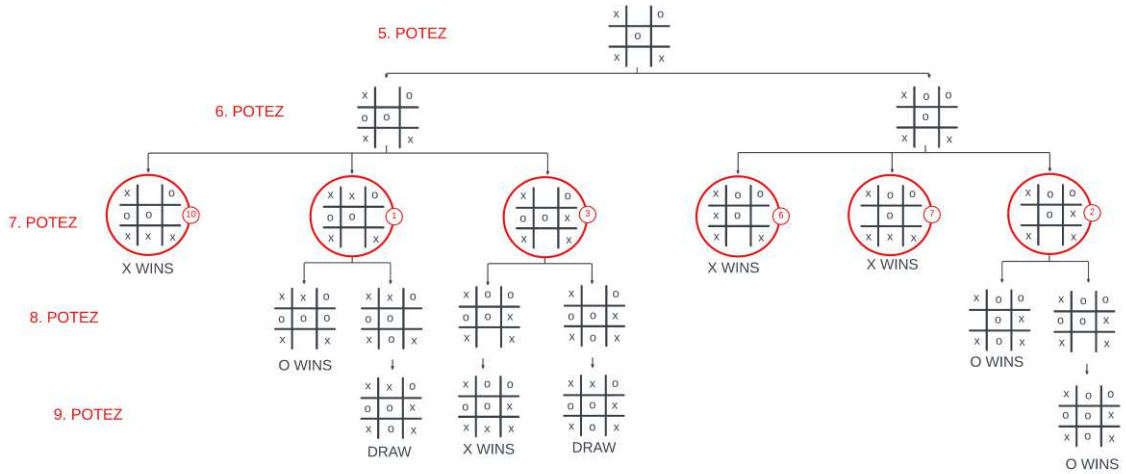
Izvor: obrada podataka

Slika 5 - 3. Primjer stabla igre križić-kružić (nastavak sa slike 3)



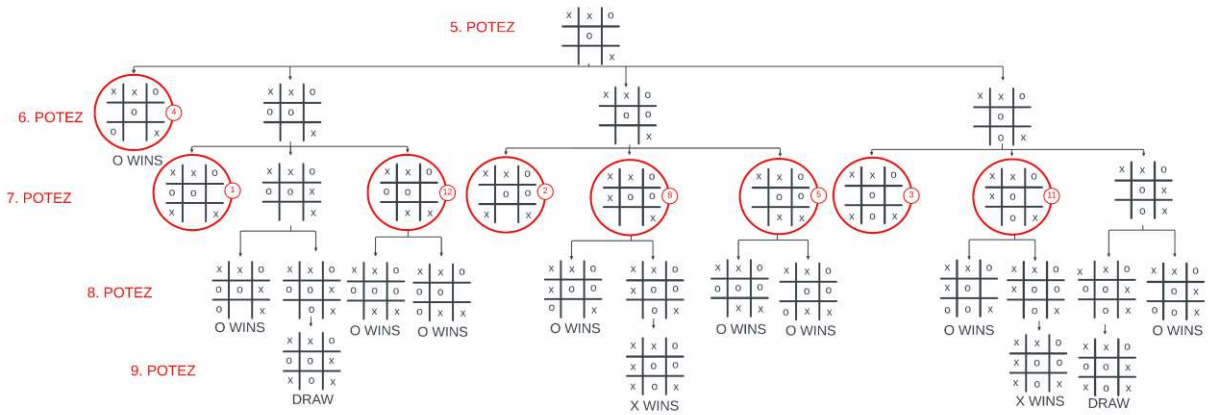
Izvor: obrada podataka

Slika 6 - 3. Primjer stabla igre križić-kružić (nastavak sa slike 5)



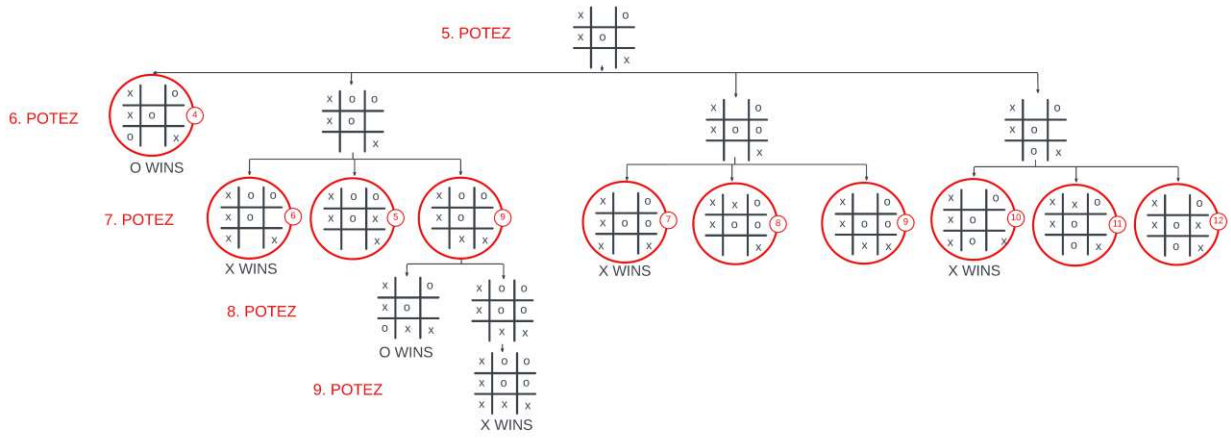
Izvor: obrada podataka

Slika 7 - 3. Primjer stabla igre križić-kružić (nastavak sa slike 5)



Izvor: obrada podataka

Slika 8 - 3. Primjer stabla igre križić-kružić (nastavak sa slike 5)



Izvor: obrada podataka

U primjeru 3 se opet vidi da se kroz cijelo stablo igre pojavljuju simetrične pozicije u različitim dijelovima stabla. Simetrične pozicije su zaokružene crvenim krugom, te označene brojem.

6.4. Primjena stabla igre u analizi igre križić-kružić

Stablo igre se koristi za analizu svih mogućih ishoda igre i za razvoj optimalnih strategija. Igrači mogu koristiti stablo kako bi predvidjeli sve moguće odgovore protivnika na svaki njihov potez i kako bi odabrali poteze koji vode do pobjede ili izbjegavanja poraza. Ovaj proces se naziva *backtracking* ili unazadna analiza, gdje igrači počinju od konačnih stanja igre (listova stabla) i vraćaju se unazad prema korijenu stabla, analizirajući svaki potez i birajući optimalne opcije.

Na primjer, ako igrač analizira stablo igre i vidi da određeni potez protivniku omogućava neizbježnu pobjedu u nekoliko sljedećih poteza, taj potez će biti izbjegnut. Umjesto toga, igrač će tražiti potez koji vodi do ishoda koji mu je najpovoljniji. U optimalnom slučaju, igrači koji koriste stablo igre i igraju savršeno završavaju igru neriješeno, bez pobjednika.

Minimax strategija je jedna od najvažnijih metoda za analizu stabla igre. Ova strategija minimizira maksimalni gubitak igrača, što je posebno korisno u igrama poput križić-kružića, gdje svaki potez može drastično promijeniti ishod igre. Minimax strategija temelji se na ideji da igrači pretpostavljaju da će protivnik uvijek birati potez koji je za njih najnepovoljniji, pa tako biraju poteze koji minimiziraju njihov potencijalni gubitak [1], [10], [11].

6.5. Vizualizacija i reprezentacija stabla igre

Vizualno prikazivanje stabla igre križić-kružić može postati vrlo složeno zbog velikog broja mogućih stanja i poteza. U praksi, stablo igre se obično prikazuje na način da se najvažniji potezi i njihove grane istaknu, dok se simetrične i redundantne grane često izostavljaju kako bi se pojednostavio prikaz.

Na primjer, ako igrač "X" odabere središnju poziciju u prvom potezu, sve moguće ishode tog poteza možemo prikazati u jednom dijelu stabla, dok se drugi mogući potezi (kao što su postavljanje "X" u ugao ili na rub) mogu prikazati u drugim dijelovima stabla. Kako se igra razvija, grane stabla koje vode do istih konačnih stanja mogu se spojiti, čime se smanjuje složenost prikaza.

Zbog ograničenog prostora, stablo igre križić-kružić često se prikazuje na više stranica ili dijelova, pri čemu se svaki dio stabla analizira zasebno. Na primjer, možemo označiti određene čvorove ili grane s posebnim simbolima kako bismo ih povezali s nastavkom na drugoj stranici ili u drugom dijelu prikaza. Ovaj pristup omogućava da se kompleksnost stabla igre zadrži pod kontrolom, istovremeno omogućujući dubinsku analizu svih mogućih scenarija [1],[6].

6.6. Stablo igre u drugim igrama

Stablo igre nije specifično samo za križić-kružić; koristi se u analizi mnogih drugih igara, kako jednostavnih, tako i složenih. Na primjer:

- Šah: Stablo igre u šahu je izuzetno veliko zbog ogromnog broja mogućih poteza u svakoj fazi igre. Ipak, osnovni principi izgradnje stabla igre i analize poteza su slični onima u križić-kružiću, samo na mnogo većem nivou složenosti [2].
- Dama: Kao i u križić-kružiću, stablo igre se koristi za analiziranje svih mogućih poteza i ishoda u igri dame. Broj mogućih stanja je značajno manji nego u šahu, ali je i dalje dovoljno velik da zahtijeva sofisticirane tehnike analize.
- Poker: U igrama s nepotpunom informacijom, poput pokera, stablo igre uključuje ne samo sve moguće poteze igrača, već i sve moguće kombinacije skrivenih informacija (npr. koje karte protivnik ima). Analiza stabla igre u ovom kontekstu je mnogo složenija te se zbog toga uzima veliki broj varijabli.

Stablo igre je univerzalni alat u teoriji igara koji omogućava dubinsku analizu igara s potpunom i nepotpunom informacijom, jednostavnih i složenih igara, kao i igara s različitim vrstama interakcija među igračima.

7. Zapis stabla igre u računalu

Zapisivanje stabla igre u računalu predstavlja ključni korak u razvoju algoritama za analizu i donošenje odluka u igrama poput križić-kružića. Stablo igre je osnovni alat koji omogućava programima da "razmišljaju" i predviđaju sve moguće ishode igre, što je neophodno za izgradnju inteligentnih protivnika ili za automatizaciju igre. U ovom odlomku detaljno ćemo razmotriti načine na koje se stablo igre može zapisati u računalnom programu, različite strukture podataka koje se koriste za ovaj zadatak, te kako se ove tehnike primjenjuju u slučaju igre križić-kružić.

7.1. Struktura podataka za zapis stabla igre

Stablo igre se u računalu najčešće predstavlja korištenjem struktura podataka kao što su stabla i grafovi. Ove strukture omogućavaju efikasno čuvanje i pretraživanje svih mogućih stanja igre i poteza koji vode do tih stanja.

1. Binarno i N-arno stablo: U računalnim znanostima, stablo je hijerarhijska struktura koja se sastoji od čvorova, gdje svaki čvor može imati jedan ili više djece (pod čvorova). Stablo igre križić-kružić je obično N-arno stablo, što znači da svaki čvor može imati više od dva djeteta, ovisno o broju preostalih mogućih poteza u svakom trenutku igre.
    - Korijen stabla predstavlja početno stanje igre, tj. praznu ploču.
    - Unutarnji čvorovi predstavljaju među stanja igre nakon što su određeni potezi već odigrani.
    - Listovi stabla predstavljaju završna stanja igre, koja mogu biti pobjeda, poraz ili neriješeno.
2. Grafovi: Iako se stablo igre često opisuje pomoću stabala, može se zapisati i kao graf, koji je fleksibilniji u prikazivanju međusobno povezanih čvorova. Graf može biti usmjeren ili neusmjeren, ovisno o tome kako su povezani potezi. U

slučaju križić-kružića, obično koristimo usmjereni graf, gdje svaka usmjerena grana predstavlja prijelaz iz jednog stanja igre u drugo.

3. Polje (Array) i popis (List): Jednostavniji način za predstavljanje stabla igre, naročito u manje kompleksnim igrama, je korištenje polja (eng. array) ili popisa (eng. list). U ovom slučaju, svako stanje igre može biti zapisano kao element polja ili popisa, a prijelazi između stanja mogu biti zapisani pomoću indeksa koji povezuju ove elemente. Ovaj pristup je jednostavan i može biti efikasan u igrama poput križić-kružića gdje je broj mogućih stanja relativno mali.

7.2. Zapis stabla igre križić-kružić u računalu

Za igru križić-kružić, zapis stabla igre u računalu obično se temelji na korištenju N-arnih stabala ili usmjerenih grafova, jer ova struktura omogućava jasan i organiziran prikaz svih mogućih sekvenci poteza i ishoda igre. Evo nekoliko pristupa kako se to može učiniti:

1. Struktura čvora: Svaki čvor u stablu igre križić-kružić može biti predstavljen kao struktura ili klasa koja sadrži informacije o trenutnom stanju igre, mogućim potezima i poveznicama na druge čvorove. Na primjer:

```
class Node:
    def __init__(self, board_state):
        self.board_state = board_state  # Trenutno stanje igre (npr. lista ili niz s pozicijama "X" i "O")
        self.children = []  # Popis poveznica na djecu (moguće sljedeće poteze)
        self.value = None  # Vrijednost čvora, npr. +1 za pobjedu, 0 za neriješeno, -1 za poraz
```

2. Generiranje stabla igre: Jednom kada je struktura čvora definirana, možemo generirati stablo igre tako što ćemo rekurzivno prolaziti kroz sve moguće poteze:

```
def generate_tree(node, current_player):
    if is_terminal(node.board_state):
        node.value = evaluate_board(node.board_state)
        return node
```

```
    for move in get_possible_moves(node.board_state):
        new_board_state = apply_move(node.board_state, move, current_player)
        child_node = Node(new_board_state)
        node.children.append(child_node)
        generate_tree(child_node, switch_player(current_player))

    return node
```

Ova funkcija rekurzivno prolazi kroz sva moguća stanja igre, kreirajući novi čvor za svako stanje i povezujući ga s roditeljskim čvorom. Kada se dosegne konačno stanje (pobjeda, poraz ili neriješeno), čvor se evaluira i njegova vrijednost se postavlja.

3. Minimax algoritam: Minimax algoritam je jedna od najčešće korištenih tehnika za evaluaciju stabla igre u igrama poput križić-kružića. Ovaj algoritam prolazi kroz stablo igre, evaluirajući svaki čvor i birajući potez koji minimizira maksimalni gubitak (ili maksimizira minimalni dobitak) za igrača [1], [11].

Evo primjera kako bi izgledala funkcija za primjenu minimax algoritma:

```
def minimax(node, maximizing_player):
    if is_terminal(node.board_state):
        return node.value

    if maximizing_player:
        max_eval = float('-inf')
        for child in node.children:
            eval = minimax(child, False)
            max_eval = max(max_eval, eval)
        node.value = max_eval
        return max_eval
    else:
        min_eval = float('inf')
        for child in node.children:
            eval = minimax(child, True)
```

```
            min_eval = min(min_eval, eval)
        node.value = min_eval
        return min_eval
```

Ovaj algoritam omogućava računalnim programima da igraju križić-kružić optimalno, birajući poteze koji vode do pobjede ili izbjegavanja poraza, čak i protiv savršeno igrajučeg protivnika.

7.3. Optimizacija zapisa stabla igre

Dok stablo igre omogućava detaljnu analizu svih mogućih ishoda igre, kompletno generiranje i pohrana stabla za kompleksnije igre može zahtijevati mnogo memorije i procesorskog vremena. Zbog toga se često primjenjuju različite tehnike optimizacije.

1. Simetrična stanja: Mnoge igre, uključujući križić-kružić, sadrže simetrična stanja. Na primjer, rotacija ili refleksija ploče može rezultirati istim stanjem igre. Prepoznavanje i eliminiranje takvih simetričnih stanja može značajno smanjiti veličinu stabla igre.
2. Alfa-beta obrezivanje: Alfa-beta obrezivanje je tehnika koja optimizira minimax algoritam smanjujući broj čvorova koje treba evaluirati. Ova metoda koristi dva parametra, alfa i beta, kako bi obrezala grane stabla koje ne mogu utjecati na konačnu odluku, čime se ubrzava proces evaluacije [1].
3. Heuristička evaluacija: Umjesto potpunog generiranja stabla igre do konačnih stanja, u složenijim igrama često se koriste heuristički evaluacijski algoritmi koji procjenjuju vrijednost među stanja igre bez potrebe da se dosegne kraj igre. Ova metoda omogućava brže donošenje odluka, ali zahtijeva precizno dizajnirane heurističke funkcije [11].

7.4. Primjena u stvarnim sustavima

Zapis stabla igre i algoritmi za pretragu stabla igre nalaze široku primjenu izvan same igre križić-kružić. Ove tehnike koriste se u raznim sustavima za donošenje odluka, automatizirane sustave, i umjetnu inteligenciju. Na primjer:

- Računalni šah: Algoritmi koji koriste stabla igre i minimax sa alfa-beta obrezivanjem čine osnovu za moderne šahovske programe.

- Automatizirana rješavanja problema: Stabla odluka koriste se za modeliranje i rješavanje problema u industriji, financijama, medicini i mnogim drugim područjima.
- Robotska navigacija: Algoritmi pretrage stabla koriste se za planiranje putanja i donošenje odluka kod autonomnih robota.

8. Algoritam Minimax

Algoritam Minimax je ključna tehnika u teoriji igara koja omogućava optimizaciju donošenja odluka u igrama sa dva igrača, kao što je križić-kružić. Ovaj algoritam se koristi za analizu stabla igre i odabir najboljeg mogućeg poteza za jednog igrača, uz pretpostavku da će protivnik igrati optimalno kako bi minimizirao maksimalni dobitak prvog igrača. U ovom odlomku detaljno ćemo objasniti kako funkcionira Minimax algoritam, uključujući njegovu osnovnu strukturu, kako se primjenjuje u igri križić-kružić, i različite tehnike za poboljšanje njegovih performansi [1], [4], [10], [11] .

8.1. Osnovna ideja Minimax algoritma

Minimax algoritam temelji se na ideji da se igra može modelirati kao stablo odluka, gdje svaki čvor predstavlja stanje igre, a grane predstavljaju moguće poteze. Cilj algoritma je pronaći najbolji potez za trenutnog igrača uz pretpostavku da će protivnik igrati optimalno kako bi minimizirao maksimalni dobitak. Algoritam radi po sljedećim principima:

1. Igrači: Postoje dva igrača - "maksimizator" i "minimizator". Maksimizator pokušava maksimizirati svoj dobitak, dok minimizator pokušava minimizirati maksimalni dobitak maksimizatora.
2. Evaluacija čvorova: Svaki čvor u stablu igre predstavlja stanje igre. Evaluacijska funkcija dodjeljuje vrijednost svakom završnom stanju igre, npr. +1 za pobjedu maksimizatora, -1 za pobjedu minimizatora, i 0 za neriješeno stanje.
3. Propagacija vrijednosti: Algoritam koristi rekurzivnu metodu za propagaciju vrijednosti iz listova stabla prema korijenu, gdje svaki čvor prima vrijednost koja odražava najbolji mogući ishod za igrača čiji je red [1], [10], [11],[12].

8.2. Koraci u Minimax algoritmu

1. Generiranje stabla igre: Stablo igre se generira počevši od početnog stanja igre i širenjem za svaki mogući potez igrača. Svaki čvor u stablu predstavlja stanje igre nakon određenog poteza.
2. Evaluacija završnih stanja: Kada se stigne do završnog stanja igre (pobjeda, poraz ili neriješeno), dodjeljuju se vrijednosti ovim stanjima koristeći evaluacijsku funkciju.
3. Propagacija vrijednosti: Vrijednosti se propagiraju natrag kroz stablo koristeći Minimax pravilo:
    - Maksimizator: Izabire maksimalnu vrijednost među vrijednostima djece čvora. Ovo znači da će izabrati potez koji vodi do najboljeg mogućeg rezultata [11].
    - Minimizator: Izabire minimalnu vrijednost među vrijednostima djece čvora. Ovo znači da će izabrati potez koji minimizira maksimalni dobitak maksimizatora [11].
4. Odabir najboljeg poteza: Nakon što se propagiraju vrijednosti do korijena stabla, najbolji potez za trenutnog igrača je onaj koji vodi do čvora sa najboljom vrijednošću [10].

8.3. Implementacija Minimax algoritma u igri križić-kružić

Implementacija Minimax algoritma u igri križić-kružić slijedi prethodno opisane korake, s posebnim fokusom na evaluaciju stanja igre i rekurzivno generiranje stabla. Evo kako bi izgledala implementacija Minimax algoritma za igru križić-kružić:

1. Definicija čvora i evaluacijske funkcije:

```
class Node:
    def __init__(self, board_state):
        self.board_state = board_state
        self.children = []
        self.value = None
```

```
def evaluate_board(board_state):
    # Evaluira stanje ploče: +1 za pobjedu "X", -1 za pobjedu "O", 0 za neriješeno
    if check_win(board_state, 'X'):
        return 1
    elif check_win(board_state, 'O'):
        return -1
    elif is_full(board_state):
        return 0
    return None
```

2. Rekurzivna funkcija Minimax:

```
def minimax(node, depth, is_maximizing):
    if is_terminal(node.board_state):
        return evaluate_board(node.board_state)

    if is_maximizing:
        max_eval = float('-inf')
        for child in node.children:
            eval = minimax(child, depth + 1, False)
            max_eval = max(max_eval, eval)
        node.value = max_eval
        return max_eval
    else:
        min_eval = float('inf')
        for child in node.children:
            eval = minimax(child, depth + 1, True)
            min_eval = min(min_eval, eval)
        node.value = min_eval
        return min_eval
```

3. Generiranje stabla i odabir poteza:

```
def generate_tree(node, current_player):
    if is_terminal(node.board_state):
        return node
```

```
    for move in get_possible_moves(node.board_state):
        new_board_state = apply_move(node.board_state, move, current_player)
        child_node = Node(new_board_state)
        node.children.append(child_node)
        generate_tree(child_node, switch_player(current_player))

    return node

def find_best_move(root_node):
    best_value = float('-inf')
    best_move = None
    for child in root_node.children:
        move_value = minimax(child, 0, False)
        if move_value > best_value:
            best_value = move_value
            best_move = child
    return best_move
```

Ova funkcija generira stablo igre, koristi Minimax za evaluaciju svih mogućih poteza, i vraća najbolji potez za trenutnog igrača [10] .

8.4. Optimizacija Minimax algoritma

1. Alfa-beta obrezivanje: Alfa-beta obrezivanje je tehnika koja poboljšava performanse Minimax algoritma smanjujući broj čvorova koje treba evaluirati. Ova tehnika koristi dva parametra – alfa i beta – kako bi obrezala grane stabla koje ne mogu utjecati na konačnu odluku:
    - Alfa predstavlja najbolju vrijednost koju trenutni igrač može garantirati do sada[1],[5].
    - Beta predstavlja najbolju vrijednost koju protivnik može garantirati do sada. Ako se bilo koji čvor u stablu pokaže da je lošiji od trenutne alfa ili beta vrijednosti, taj čvor i svi njegovi potomci se mogu zanemariti [1],[5].

2. Heuristička evaluacija: Za složenije igre, gdje je generiranje cijelog stabla nepraktično, koristi se heuristička evaluacija. Ovo uključuje procjenu stanja igre na osnovu različitih kriterija koji nisu potpuno evaluirani, ali pružaju dovoljno informacija za donošenje odluka [11].
3. Pamćenje prethodnih izračuna (Memoizacija): Pamćenje prethodno izračunanih vrijednosti za određena stanja može značajno smanjiti vrijeme potrebno za evaluaciju stabla, posebno u igrama s velikim brojem mogućih stanja.

9. Detaljna objašnjenja koda za aplikaciju križić-kružić

U ovom odlomku ćemo detaljno objasniti svaki dio koda aplikacije križić-kružić, koja je izrađena pomoću Pythonove Tkinter biblioteke za grafičko korisničko sučelje. Kod implementira igru križić-kružić s osnovnim funkcionalnostima za igranje između korisnika i računala, uključujući algoritam Minimax za odlučivanje poteza računala.

Kod igre je pohranjen na github repozitoriju:
https://github.com/antoniolazaric/KrizicKruzicZavrsniRad

9.1. Uvod

Kod se sastoji od tri glavna dijela: definicija globalnih varijabli i funkcija za logiku igre, funkcionalnosti za igru između korisnika i računala, i postavljanje grafičkog korisničkog sučelja (GUI) pomoću Tkinter-a.

9.2. Globalne varijable

```
board = {1: ' ', 2: ' ', 3: ' ',
        4: ' ', 5: ' ', 6: ' ',
        7: ' ', 8: ' ', 9: ' '}
player = 'O'
bot = 'X'
```

- board: Rječnik koji predstavlja stanje ploče igre. Ključevi (1-9) odgovaraju pozicijama na ploči, a vrijednosti su znakovi ' ' (prazno), 'O' ili 'X', koji označavaju koji je igrač postavio znak na tu poziciju.

- player i bot: Varijable koje definiraju znakove igrača i računala. U ovom slučaju, igrač koristi 'O', a računalo koristi 'X'.

9.3. Funkcije za logiku igre

9.3.1 Provjera pobjednika

checkForWin()

```
def checkForWin():
    if (board[1] == board[2] and board[1] == board[3] and board[1] != ' '):
        return True
    elif (board[4] == board[5] and board[4] == board[6] and board[4] != ' '):
        return True
    elif (board[7] == board[8] and board[7] == board[9] and board[7] != ' '):
        return True
    elif (board[1] == board[4] and board[1] == board[7] and board[1] != ' '):
        return True
    elif (board[2] == board[5] and board[2] == board[8] and board[2] != ' '):
        return True
    elif (board[3] == board[6] and board[3] == board[9] and board[3] != ' '):
        return True
    elif (board[1] == board[5] and board[1] == board[9] and board[1] != ' '):
        return True
    elif (board[7] == board[5] and board[7] == board[3] and board[7] != ' '):
        return True
    else:
        return False
```

- Ova funkcija provjerava sve moguće kombinacije za pobjedu na ploči. Ako se tri znaka u redu, stupcu ili dijagonali poklapaju i nisu prazna, funkcija vraća True, što znači da je netko pobijedio.

9.3.2 Provjera izjednačenja

checkDraw()

```
def checkDraw():
    for key in board.keys():
        if board[key] == ' ':
            return False
    return True
```

- Funkcija provjerava je li ploča puna i da li postoji još neko slobodno polje. Ako su sva polja popunjena i nitko nije pobijedio, vraća True, što označava da je igra završila neriješeno.

9.3.3. Provjera koji je igrač pobijedio

```
 checkWhichMarkWon(mark)
def checkWhichMarkWon(mark):
    if board[1] == board[2] and board[1] == board[3] and board[1] == mark:
        return True
    elif (board[4] == board[5] and board[4] == board[6] and board[4] == mark):
        return True
    elif (board[7] == board[8] and board[7] == board[9] and board[7] == mark):
        return True
    elif (board[1] == board[4] and board[1] == board[7] and board[1] == mark):
        return True
    elif (board[2] == board[5] and board[2] == board[8] and board[2] == mark):
        return True
    elif (board[3] == board[6] and board[3] == board[9] and board[3] == mark):
        return True
    elif (board[1] == board[5] and board[1] == board[9] and board[1] == mark):
        return True
    elif (board[7] == board[5] and board[7] == board[3] and board[7] == mark):
        return True
    else:
        return False
```

- Ova funkcija provjerava je li igrač koji koristi označeni znak (mark) pobijedio. Funkcija uzima jedan argument mark, koji može biti 'O' ili 'X', i provjerava sve pobjedničke kombinacije za taj znak.

9.3.4. Provjera dali je pozicija slobodna

```
 spaceIsFree(position)
def spaceIsFree(position):
    return board[position] == ' '
```

- Funkcija provjerava je li određena pozicija na ploči slobodna za postavljanje znaka.

9.3.5. Postavljanje znaka (X ili O)

```
 insertLetter(letter, position)
def insertLetter(letter, position):
    if spaceIsFree(position):
        board[position] = letter
        buttons[position].config(text=letter)
        if checkForWin():
            if letter == bot:
                messagebox.showinfo("Game Over", "Bot wins!")
            else:
                messagebox.showinfo("Game Over", "Player wins!")
            resetBoard()
        elif checkDraw():
            messagebox.showinfo("Game Over", "It's a draw!")
            resetBoard()
    else:
        messagebox.showerror("Error", "Can't insert there!")
```

- Ova funkcija postavlja znak na određenu poziciju ako je ta pozicija slobodna. Nakon postavljanja znaka, funkcija provjerava je li netko pobijedio ili je li igra završila neriješeno. Ako je igra gotova, prikazuje odgovarajuću poruku i resetira ploču.

9.3.6. Pozivanje funkcije poteza računala

```
 playerMove(position)
def playerMove(position):
    if spaceIsFree(position):
```

```
    insertLetter(player, position)
    compMove()
```

- Funkcija za obradu poteza igrača. Kada igrač odabere poziciju i postavi svoj znak, funkcija poziva compMove() za obavljanje poteza računala.

9.3.7. Pozivanje funkcije minimax

```
compMove()
def compMove():
  bestScore = -800
  bestMove = 0
  for key in board.keys():
    if board[key] == ' ':
      board[key] = bot
      score = minimax(board, 0, False)
      board[key] = ' '
      if score > bestScore:
        bestScore = score
        bestMove = key
  insertLetter(bot, bestMove)
```

- Funkcija za računalo da napravi potez koristeći Minimax algoritam. Računalo prolazi kroz sve moguće slobodne pozicije, simulira postavljanje svog znaka, izračunava ocjenu poteza pomoću Minimax funkcije, i bira najbolji potez na temelju ocjene [10], [11] .

9.3.8. Funkcija minimax

```
 minimax(board, depth, isMaximizing)
def minimax(board, depth, isMaximizing):
  if checkWhichMarkWon(bot):
    return 1
  elif checkWhichMarkWon(player):
    return -1
  elif checkDraw():
    return 0
```

```
    if isMaximizing:
        bestScore = -800
        for key in board.keys():
            if board[key] == ' ':
                board[key] = bot
                score = minimax(board, depth + 1, False)
                board[key] = ' '
                if score > bestScore:
                    bestScore = score
        return bestScore
    else:
        bestScore = 800
        for key in board.keys():
            if board[key] == ' ':
                board[key] = player
                score = minimax(board, depth + 1, True)
                board[key] = ' '
                if score < bestScore:
                    bestScore = score
        return bestScore
```

- Implementacija Minimax algoritma. Ova funkcija rekurzivno prolazi kroz sve moguće poteze, procjenjuje svaki potez i vraća najbolju moguću ocjenu za trenutnog igrača. isMaximizing označava je li trenutni igrač maksimizator ili minimizator [10], [11] .

9.3.9. Funkcija za resetiranje ploče na početno stanje

```
 resetBoard()
def resetBoard():
    global board
    board = {1: ' ', 2: ' ', 3: ' ',
             4: ' ', 5: ' ', 6: ' ',
             7: ' ', 8: ' ', 9: ' '}
```

```
    for button in buttons.values():
        button.config(text='')
```

- Funkcija za resetiranje stanja ploče. Ploča se vraća na početno stanje, a svi gumbi u GUI-u se postavljaju na prazne.

9.3.10. Inicijalizacija Tkinter prozora

```
root = tk.Tk()
root.title("Tic-Tac-Toe")
```

- Ova linija stvara glavni prozor aplikacije i postavlja naslov prozora na "Tic-Tac-Toe".

9.3.11. Postavljanje gumba

```
buttons = {}


for i in range(1, 10):
    button = tk.Button(root, text=' ', font='normal 20 bold', height=3, width=6,
                       command=lambda i=i: playerMove(i))
    button.grid(row=(i-1)//3, column=(i-1)%3)
    buttons[i] = button
```

- Ovdje se stvaraju gumbi za svaku poziciju na ploči igre. Svaki gumb se povezuje s funkcijom playerMove, koja se poziva kada igrač klikne na gumb.

9.3.12 Postavljanje gumba za resetiranje

```
reset_button = tk.Button(root, text='Reset', font='normal 20 bold', height=1, width=6, command=resetBoard)

reset_button.grid(row=3, column=1)
```

- Gumb za resetiranje igre. Kada se klikne, poziva funkciju resetBoard za vraćanje ploče na početno stanje.

9.3.13 Pokretanje glavne petlje Tkinter-a

```
root.mainloop()
```

- Ova linija pokreće glavnu petlju Tkinter-a koja omogućava interakciju s GUI-jem i održava prozor otvorenim.

10. Upute za korištenje aplikacije križić-kružić

Ovo poglavlje pruža detaljne upute o tome kako koristiti aplikaciju križić-kružić. Aplikacija omogućava igru između korisnika i računala, s interaktivnim grafičkim korisničkim sučeljem (GUI) izrađenim pomoću Tkinter biblioteke u Pythonu. Slijedite ove upute za razumijevanje osnovnih funkcionalnosti i kako igrati igru.

10.1. Pokretanje aplikacije

Prvi način pokretanja igre je pomoću Vscode aplikacije ili bilo kojeg programa koji ima mogućnost pokretanja python koda tako što ćete izvršiti Python skriptu koja sadrži kod za križić-kružić. Ako koristite IDE poput PyCharm-a, možete jednostavno otvoriti skriptu i kliknuti na “Run” ili “Execute”. Ako koristite terminal, upišite python KrizicKruzicZavrsniRad.py

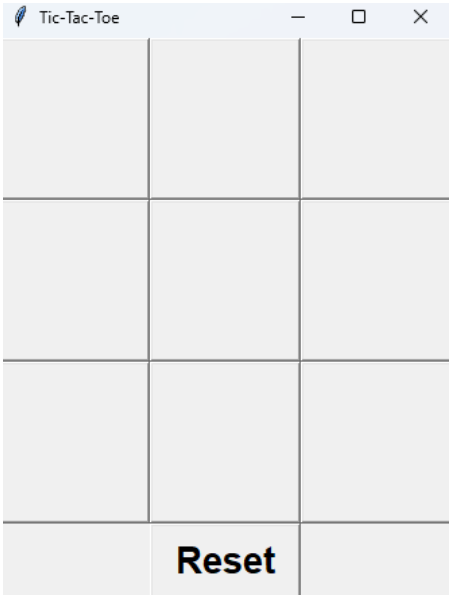
Drugi način je jednostavnim pokretanjem KrizicKruzicZavrsniRad.exe datoteke nakon kojeg će se aplikacija automatski pokrenuti. Aplikacija je prebačena u .exe format zbog lakoće pokretanja i jednostavnosti.

10.2. Interakcija s GUI-jem

Nakon pokretanja aplikacije, otvorit će se prozor s grafičkim korisničkim sučeljem. Sučelje uključuje:

1. Ploča igre: Ploča igre je mreža od 3x3 kvadrata. Svaki kvadrat predstavlja jednu poziciju na ploči gdje igrač ili računalo mogu postaviti svoj znak. Kvadrati su predstavljeni kao gumbi u GUI-u.
2. Gumbi za igru: Svaki kvadrat na ploči igre je interaktivan gumb. Kada igrač klikne na jedan od ovih gumba, postavlja svoj znak ('O') na odabranu poziciju.
3. Gumb za resetiranje: U donjem dijelu prozora nalazi se gumb "Reset". Klikom na ovaj gumb, ploča se vraća u početno stanje, brišući sve oznake i omogućujući novu igru.

Slika 9 - Izgled aplikacije križić-kružić



Izvor: obrada podataka

10.3. Pravila igre

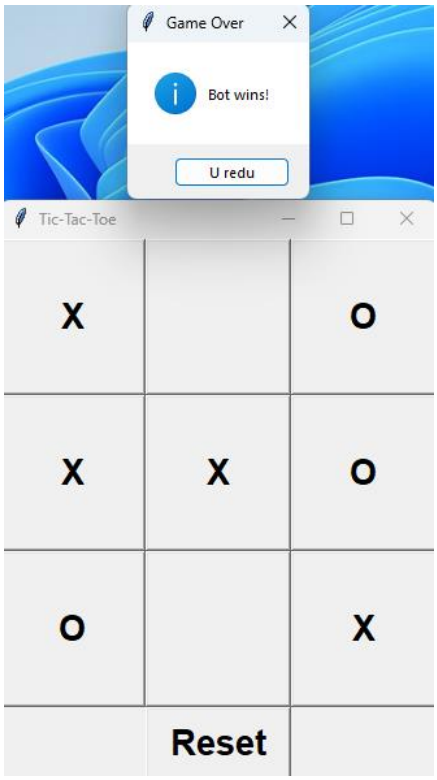
1. Cilj igre: Cilj igre križić-kružić je postaviti tri svoja znaka u red, stupac ili dijagonalu prije nego što to učini protivnik. Igrač koji prvi postigne ovu kombinaciju pobjeđuje.
2. Igrač i računalo: Igrač koristi znak 'O', dok računalo koristi znak 'X'. Igra započinje potezom igrača.

10.4. Kako igrati

1. Izbor poteza:
    - Kliknite na jedan od praznih gumba na ploči igre da biste postavili svoj znak ('O') na tu poziciju.
    - Nakon što igrač postavi svoj znak, računalo će automatski napraviti svoj potez koristeći algoritam Minimax.

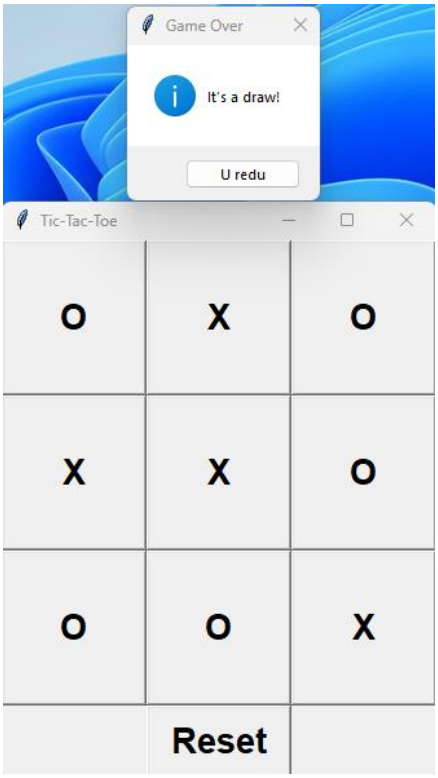
2. Praćenje rezultata:
    - Ako igrač ili računalo postave tri znaka u red, stupac ili dijagonalu, aplikacija će prikazati poruku o pobjedi za odgovarajućeg igrača i automatski resetirati ploču nakon nekoliko trenutaka.
    - Ako su svi kvadrati ispunjeni, a nitko ne pobijedi, aplikacija će prikazati poruku o neriješenom rezultatu i resetirati ploču.

Slika 10 - Izgled aplikacije kada računalo pobijedi



Izvor: obrada podataka

Slika 11 - Izgled aplikacije kada je igra izjednačena



Izvor: obrada podataka

3. Resetiranje igre:
    - Kliknite na gumb "Reset" da biste očistili ploču i započeli novu igru.

11. Zaključak

U ovom završnom radu, razvili smo aplikaciju za igru križić-kružić koristeći Python i Tkinter, koja omogućava interaktivnu igru između korisnika i računala. Kroz projekt smo istražili i primijenili različite koncepte iz teorije igara, implementirali algoritam Minimax za optimizaciju poteza računala i razvili funkcionalno grafičko korisničko sučelje (GUI). Ovaj zaključak pruža sažetak ključnih aspekata projekta, analiziramo postignute rezultate, identificiramo potencijalna poboljšanja i razmatramo moguće smjerove za budući razvoj.

11.1. Ključni aspekti projekta

1. Razumijevanje teorije igara:
    - Teorija igara pruža matematički okvir za analizu strategija u različitim situacijama. U ovom projektu, primijenili smo teoriju igara za analizu i implementaciju strategija koje računalo koristi za donošenje odluka. Implementacija Minimax algoritma omogućava optimalnu igru računala protiv ljudskog igrača, temeljeći se na principima teorije igara.
2. Implementacija Minimax algoritma:
    - Algoritam Minimax igra ključnu ulogu u omogućavanju računalu da donese optimalne odluke. Ovaj algoritam koristi rekurziju za analizu svih mogućih poteza i odabire onaj koji maksimizira šanse za pobjedu ili minimizira gubitke, ovisno o tome tko je na potezu. Analizom svih mogućih scenarija igre, algoritam osigurava da računalo igra na najbolji mogući način.
3. Razvoj grafičkog korisničkog sučelja (GUI):
    - Tkinter je korišten za razvoj GUI-a koji omogućava jednostavno i intuitivno igranje igre. Sučelje uključuje ploču igre s gumbima koji predstavljaju pozicije na ploči, te gumb za resetiranje igre. Kroz GUI, korisnici mogu lako odabrati svoje poteze i pratiti stanje igre, što poboljšava ukupno iskustvo igre.

4. Validacija i testiranje:
    - Implementacija je testirana kroz različite scenarije igre kako bi se osiguralo da aplikacija ispravno prepoznaje pobjede, neriješene rezultate i pravilno reagira na poteze igrača i računala. Testiranje je uključivalo provjeru svih funkcionalnosti, uključujući ispravnost prikaza rezultata i pravilnost poteza računala.

11.2. Postignuti rezultati

Projekt je postigao nekoliko ključnih rezultata:

- Funkcionalna aplikacija: Razvijena aplikacija omogućava igru križić-kružić između korisnika i računala s potpuno funkcionalnim GUI-em.
- Optimalna strategija igre: Računalo koristi Minimax algoritam za donošenje optimalnih poteza, što omogućava izazovnu igru za korisnike.
- Interaktivan dizajn: GUI pruža jednostavno i korisnički prijatno sučelje koje omogućava intuitivno igranje igre i lako praćenje stanja igre.

12. Literatura

[1] Y. Lin, Computer science game trees. Dostupno na:
https://www.ocf.berkeley.edu/~yosenl/extras/alphabeta/alphabeta.html
Pristupljeno 14.08.2024.

[2] A. Chik, Game Trees. Dostupno na:
https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://viterbi-web.usc.edu/~adamchik/15-121/lectures/Game%2520Trees/Game%2520Trees.html&ved=2ahUKEwjwiMbTrMWIAxUI1gIHHca2FT4QFnoECBsQAQ&usg=AOvVaw203jTh2qj71SN4pjNazwAp
Pristupljeno 14.08.2024

[3] Faster Capital, The importance of game trees in game theory. Dostupno na:
https://fastercapital.com/topics/the-importance-of-game-trees-in-game-theory.html
Pristupljeno 14.08.2024

[4] Scaler Academy, Game Tree in AI. Dostupno na:
https://www.scaler.com/topics/game-tree-in-ai/
Pristupljeno 16.08.2024.

[5] Scratch Wiki, Game Tree. Dostupno na: https://en.scratch-wiki.info/wiki/Game_Tree
Pristupljeno 14.08.2024.

[6] Wikipedia, Game tree. Dostupno na: https://en.wikipedia.org/wiki/Game_tree
Pristupljeno 16.08.2024.

[7] Exploratorium, Tic-Tac-Toe Puzzle. Dostupno na:
https://www.exploratorium.edu/explore/puzzles/tictactoe
Pristupljeno 12.08.2024.

[8] WikiHow, How to Play Tic-Tac-Toe. Dostupno na: https://www.wikihow.com/Play-Tic-Tac-Toe
Pristupljeno 16.08.2024.

[9] BoardGameGeek, Tic-Tac-Toe. Dostupno na:
https://boardgamegeek.com/blogpost/152602/tic-tac-toe
Pristupljeno 10.08.2024.

[10] JavaTpoint, Mini-Max Algorithm in AI. Dostupno na:
https://www.javatpoint.com/mini-max-algorithm-in-ai
Pristupljeno 15.08.2024.

[11] Stanford University, Minimax in AI. Dostupno na:
https://cs.stanford.edu/people/eroberts/courses/soco/projects/2003-04/intelligent-search/minimax.html
Pristupljeno 15.08.2024.

[12] Stanford University, Game Theory and Minimax Algorithm. Dostupno na:
https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/game-theory/Minimax.html
Pristupljeno 12.08.2024.

[13] Zayd Muhammad Kawakibi Zuhri, Dostupno na:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Makalah2021/Makalah-Matdis-2021%20(148).pdf

Pristupljeno 12.09.2024.

13. Popis slika

Slika 1 - 1. Primjer stabla igre križić-kružić: simbol "O" je stavljen u kut

Slika 2 - 2. Primjer stabla igre križić-kružić: simbol "O" nije stavljen u kut

Slika 3 - 3. Primjer stabla igre križić-kružić

Slika 4 - 3. Primjer stabla igre križić-kružić (nastavak sa slike 3)

Slika 5 - 3. Primjer stabla igre križić-kružić (nastavak sa slike 3)

Slika 6 - 3. Primjer stabla igre križić-kružić (nastavak sa slike 5)

Slika 7 - 3. Primjer stabla igre križić-kružić (nastavak sa slike 5)

Slika 8 - 3. Primjer stabla igre križić-kružić (nastavak sa slike 5)

Slika 9 - Izgled aplikacije križić-kružić

Slika 10 - Izgled aplikacije kada računalo pobijedi

Slika 11 - Izgled aplikacije kada je igra izjednačena

14. Popis tablica

Tablica 1 - Primjer igre koja završava pobjedom za “X“

Tablica 2 - Potez 1 primjera igre križić-kružić

Tablica 3 - Potez 2 primjera igre križić-kružić

Tablica 4 - Potez 3 primjera igre križić-kružić

Tablica 5 - Potez 4 primjera igre križić-kružić

Tablica 6 - Potez 5 primjera igre križić-kružić

Tablica 7 - Potez 6 primjera igre križić-kružić

15. Sažetak

Teorija igara, kao matematička disciplina, bavi se analizom strategija donošenja odluka u situacijama međusobne interakcije igrača, gdje ishod ovisi o izborima svih sudionika. U ovom radu istražena je primjena teorije igara na jednostavnu igru križić-kružić (eng. *Tic-Tac-Toe*). Križić-kružić, iako naizgled trivijalna igra, predstavlja izvrstan model za proučavanje osnovnih koncepata teorije igara, poput optimalnih strategija, stabla igre i algoritama za donošenje odluka.

Rad se sastoji od dva dijela: teorijskog i praktičnog. Teorijski dio daje pregled osnovnih pojmova teorije igara, klasifikacije igara te njihovih karakteristika, poput broja igrača, kooperativnosti, dostupnosti informacija i dinamike igre. Praktični dio fokusira se na analizu igre križić-kružić kroz konstrukciju stabla igre, koje prikazuje sve moguće poteze i ishode. Implementiran je Minimax algoritam, koji omogućava računalu donošenje optimalnih poteza, te je kroz aplikaciju demonstrirana njegova učinkovitost.

Metodologija rada uključivala je razvoj računalne aplikacije za igru križić-kružić korištenjem programskog jezika Python i Tkinter biblioteke za grafičko korisničko sučelje (GUI). Aplikacija omogućava interaktivnu igru između korisnika i računala, gdje računalo koristi Minimax algoritam za izračunavanje najboljih poteza. Stablo igre križić-kružić korišteno je za prikaz svih mogućih sekvenci igre, što je omogućilo potpunu analizu svih mogućih ishoda i optimalnih strategija.

Rezultati projekta pokazuju da Minimax algoritam pruža optimalno rješenje za igru križić-kružić, gdje, uz optimalno igranje s obje strane, igra uvijek završava neriješeno. Također, aplikacija razvijena u ovom radu pokazuje kako se teorija igara može primijeniti u razvoju inteligentnih sustava za donošenje odluka.

Zaključno, ovaj rad demonstrira važnost teorije igara u analizama čak i jednostavnih igara poput križić-kružića, a implementacija Minimax algoritma daje uvid u optimizacijske procese u igrama s nultom sumom. Daljnji razvoj rada može uključivati složenije igre s većim brojem igrača ili igara s nepotpunom informacijom, čime bi se dodatno proširila primjena teorije igara i algoritama za optimizaciju.

16. Summary

Game theory, as a mathematical discipline, focuses on analyzing decision-making strategies in situations involving player interactions, where outcomes depend on the choices of all participants. This paper explores the application of game theory to the simple game of Tic-Tac-Toe. Although seemingly trivial, Tic-Tac-Toe serves as an excellent model for studying fundamental concepts of game theory, such as optimal strategies, game trees, and decision-making algorithms.

The paper is divided into two parts: theoretical and practical. The theoretical section provides an overview of key game theory concepts, game classifications, and their characteristics, such as the number of players, cooperation, availability of information, and game dynamics. The practical section focuses on the analysis of Tic-Tac-Toe through constructing a game tree, which shows all possible moves and outcomes. The Minimax algorithm, which enables the computer to make optimal moves, was implemented and demonstrated through an application.

The methodology of the project involved developing a computer application for Tic-Tac-Toe using the Python programming language and the Tkinter library for graphical user interface (GUI). The application allows for interactive gameplay between the user and the computer, where the computer uses the Minimax algorithm to calculate the best moves. The Tic-Tac-Toe game tree was used to represent all possible game sequences, allowing a full analysis of all possible outcomes and optimal strategies.

The results of the project show that the Minimax algorithm provides an optimal solution for Tic-Tac-Toe, where, with optimal play from both sides, the game always ends in a draw. Additionally, the developed application demonstrates how game theory can be applied in developing intelligent decision-making systems.

In conclusion, this paper illustrates the importance of game theory in analyzing even simple games like Tic-Tac-Toe, and the implementation of the Minimax algorithm provides insight into optimization processes in zero-sum games. Future developments could involve more complex games with more players or games with incomplete information, further expanding the application of game theory and optimization algorithms.