Kivi-paperi-sakset tekoäly

Joonatan Vuorela

1. Määrittely

1.1 **Toteutettavat algoritmit ja tietorakenteet**

Ohjelmaan toteutetaan algoritmi pelissä tapahtuvien siirtojen tulkitsemiseen ja niissä olevien ”kaavojen” havaitsemiseen. Tekoälyn pitää algoritmin avulla huomata jos jokin tietty siirto toistuu jatkuvasti joidenkin tiettyjen tapahtumien jälkeen. Ei siis riitä että katsotaan vaan edellisiä siirtoja vaan myös kierroksen lopputulos, voitto tai häviö, pitää huomioida seuraavaa siirtoa valittaessa.

Siirrot on talletettu round olioon joka taas talletetaan LinkedList tyyliseen tietorakenteeseen jossa jokainen round olio tietää edellisen ja seuraavan olion (kaksisuuntainen linkitettylista). Näin voidaan toteuttaa tietorakenteeseen next ja prev metodit ja sen läpikäymisestä tulee helppoa. Tietorakenteeseen on myös toteutettu indexillä haku. Esimerkiksi voidaan siirtyä suoraan 10:een pelikierrokseen ja aloittaa läpikäyminen siitä.

1.2 **Ratkaistavat ongelmat ja valitut tietorakenteen**

Oma tietorakenne on kaksisuuntainen linkitetty lista joka käyttää jokaisen olion next ja prev muuttujia seuraavaa oliota hakiessaan. Tietorakenteesta voi hakea indexillä ja poistaa myös indexistä jolloin tietorakenne osaa muuttaa indexit ja viittaukset oikeiksi eli viittauksia null olioihin ei pääse syntymään.

1.3 **Mitä syötteitä ohjelma saa ja miten niitä käytetään**

Ohjelma saa syötteenä string-muuttujia. Pelissä käytetään vain käskyjä k,p,s joilla viitataan siirtoihin. Pelin lopettamiselle on myös käsky ”stop”

1.4 **Tavoitteena olevat aika- ja tilavaativuudet**

Aikavaativuus on O(n). Round olioiden lukumäärä (n) on ratkaiseva tekijä niiden läpikäyntiä ajatellen. Algoritmin aikavaativuus ei koskaan ole eksponentiaalisesti kasvava.

Tilavaativuus on myös O(n) koska alkioiden määrä kasvaa ja niitä talletetaan kun uusi round olio luodaan. Periaatteessa aluksi on varattu muistia yhdelle oliolle ja sen jälkeen n+1:lle jossa n on olioiden lukumäärä

1.5 **Lähteet**

Tietoa haettu tira:n materiaalista ja ohjelmointiputkasta.

2. Testausdokumentaatio

2.1 **Mitä on testattu**

Metodien toiminnallisuutta on testattu eri syötteillä. Tämä varmistaa sen, ettei virheitä pääse käymään, vaikka syöte jostain syystä olisi väärä. Testeistä on kirjoitettu jUnit testejä mutta ohjelmaa on myös testattu omilla debugerreilla. Esimerkiksi System.out.print(”haluttuja juttuja tänne”) hyödyntäen on voitu testata ohjelmaa samalla kun sitä suoritetaan ja varmistua että oliot saavat oikeat parametrit ja mitä tapahtuu jos tulee virheellisiä syötteitä. Tällä on myös varmistettu että tekoälyn käyttämät laskurit toimivat oikein.

2.2 **Minkälaisilla syötteillä testaus tehtiin**

Testausta on suoritettu netbeansilla joten syöte on valmiiksi annettu testille. Syötteenä on annettu myös virheellistä tietoa esim. väärä siirto ja katsottu että ohjelma osaa reagoida tähän. Esimerkiksi kun testataan sitä ottaako tekoäly oikean siirron kun pelaaja on kolmesti valinnut kiven ja tämän jälkeen valinnut

saksen/paperin jonka jälkeen taas valinnut kiven kolmesti jne. sout debuggerilla pystytään varmistamaan että laskurin saama todennäköisyys putken jatkumiselle on oikea eri syötteillä ja eri tilanteissa. Syöttellä k,k,k,p,k,k,k,s,k,k,? tekoäly havaitsee että kahden saman jälkeen on tullut kolmas sama 100% siirroista ja osaa tällöin pelata tätä siirtoa vastaan.

2.3 **Miten testit voidaan toistaa**

Testit sijaitsevat projektin test kansion alla ja ne voi suorittaa esimerkiksi netbeansilla.

3. Toteutusdokumentaatio

3.1 **Ohjelman rakenne**

Ohjelmassa on käyttäjälle näkyvä käyttöliittymä (tekstipohjainen) joka hallitsee ohjelman muita luokkia ja niiden toimintaa. Käyttöliittymän sisällä luodaan Pelaaja ja AI (tekoäly) oliot sekä voiton tarkastaja (Inspector) olio joka hoitaa taas tarkistuksen voittaako toinen siirto toisen

Pakkaus **Game**

Inspector:

-Tarkistaa kumpi siirroista voittaa

Round:

-Olio jokaiselle kierrokselle

-Pelaajan ja AI:n siirto

-Kumpi voitti

RoundRemember

-Tietorakenne joka hoitaa round olion talletuksen (kaksisuuntainen linkitettylista)

Pakkaus **Player**

Player:

-Pelaajan nimi

-Pelaajan pisteet

-Hoitaa myös botin nimen ja pisteet

Pakkaus **ArtificialIntelligence**

-Tänne toteuteetaan kaikki tekoäly

AI:

-Laskee botille parhaan mahdollisen siirron

StreakCalculator & AfterLostRoundCalculator:

-Sisältää algoritmeja jotka auttavat tekoälyä seuraavan siirron valitsemisessa

Pakkaus **UserInterface**

UI:

-Ilmentymä jokaisesta yllä olevasta luokasta

-Hoitaa ohjelman pyörityksen ja siirtojen kysymisen käyttäjältä

Tekoälyä avattu:
Tekoäly keskittyy havaitsemaan tiettyjä kaavoja pelaajan siirroissa ja pelaamaan tätä vastaan. Esimerkiksi jos pelaaja käyttää samaa siirtoa kahdesti, muttei koskaan kolmesti osaa tekoäly varautua tähän ja valitsee turvallisimman siirron. Tässä tapauksessa se olisi sellainen joka häviää pelaajan siirrolle jos pelaaja käyttäisikin samaa. Esimerkiksi jos pelaajan entisistä siirroista on laskettu että todennäköisyys kolmen saman putkelle on 10%. Pelaaja on syöttänyt s ja s. Tekoäly valitsisi seuraavaksi paperin koska 90% todennäköisyydellä pelaaja valitsee k tai p jolloin paperi on paras vaihtoehto. (Ei häviä kivelle eikä paperille). Jos taas todennäköisyydeksi on laskettu 95% että pelaaja jatkaa putkeaan kahden saman

jälkeen valitaan k jos pelaaja on antanut taas syötteen s ja s. Sama lasku toteutetaan jokaisen ”putken kohdalla”. Eli kun on valittu kahdesti, kolmesti tai neljästi sama. Tämän ylittävät putket tekoäly koittaa lopettaa pelaamalla kokoajan putkea vastaan. Tekoäly myös pitää kokoajan kirjaa pelaajan siirroista ja % osuudesta kaikista siirroista. Esim jos kiveä on pelattu 65% kaikista siirroista. Tekoäly pelaa tätä vastaan valitsemalla paperin. Sama toimii toisinpäin. Jos jokin siirto on alipelattu, valitaan tälle häviävä siirto. Esimerkiksi jos pelaaja on valinnut paperin 14% ajoista, valitaan kivi. Koska kivi ei häviä paperille tai kivelle on tämä turvallisin siirto sillä vain 14% ajoista pelaaja valitsee paperin ja tekoäly häviää.

Käyttöohje:

Peli etenee antamalla netbeanssissa siirto (k,p,s). Kierroksen jälkeen näytetään pelaajalle kumpi pelaajista voitti, mikä oli tekoälyn siirto ja mikä on tilanne pelissä. Peli jatkuu ikuisesti tai kunnes annetaan komento stop. Väärällä komennolla ei tapahdu muuta kuin kehoitus valita k, p tai s.

Suorita ohjelma komentoriviltä Linux/Windows käyttöjärjestelmällä menemällä kansioon projektin kansioon ja suorittamalla java -jar Tiralabra.jar. Ajaaksesi ohjelmaa tarvitset toimivan version javasta.