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() hyödyntäen on voitu testaja ohjelmaa samalla kun sitä suoritetaan ja varmistua että oliot saavat oikeat parametrit ja mitä tapahtuu jos tulee virheellisiä syötteitä

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.

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

Käyttöohje:

Peli etenee antamalla netbeanssissa siirto (k,p,s). Kierroksen jälkeen näytetään pelaajalle kumpi 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.