Toteutusdokumentti

Ohjelman yleisrakenne

Ohjelma yrittää voittaa pelajaan käden kivi-paperi-sakset(-lisko-spock) -pelissä. Toteutus on hyvin suoraviivainen:

- Ensimmäisellä kierroksella kone valitsee satunnaisen käden pelattavaksi
- Jokaisen kierroksen jälkeen kone tallentaa pelatut kädet ja päivittää statistiikan
- Kierroksilla 2-5 kone olettaa pelaajan noudattavan yleistä rotaatiota* ja valitsee sen perusteella itselleen voittavan käden
- Kierroksen viisi jälkeen kone käyttää heuristiikkaa ennustaakseen pelaajan seuraavan käden ja asettaa itselleen tuota kättä voittavan käden

* Mikäli pelaaja voittaa kierroksen tulee pelaaja todennäköisesti pelaamaan seuraavalla kierroksella saman käden. Pelaajan hävitessä pelaaja valitsee todennäköisesti juuri pelattua kättä seuraavan käden rotaatiossa: kivi-paperi-sakset(-lisko-spock)

Saavutetut aika- ja tilavaativuudet

Ohjelma toimii vakioajassa ja -tilassa.

```
// pseudo: pelaaKierros idea
if (kierroksia == 0)
      palauta satunnainen käsi
else if (kierroksia < 6)
      palauta pelaajan rotaatiota voittava käsi
else
      int pelaajanKädet[5]
      listasolmu seuraaja = käsilista.getEkaSolmu
      listasolmu edeltäjä = seuraaja.getSeuraavaSolmu
      listasolmu verrattava = käsilista.getEkaSolmu // viimeisin peli
      while(edeltäjä != null)
            if (edeltäjä == verrattava)
                  pelaajanKädet[seuraava.getPelaajanKasi++]
            seuraaja = seuraaja.getSeuraavaSolmu
            edeltäjä = edeltäjä.getSeuraavaSolmu
      palauta käsi joka voittaa pelaajanKadet[suurin arvo]
```

Analyysi:

- Satunnaisen käden palauttaminen on vakioaikainen operaatio
- Pelaajan rotaation selvittäminen vie huonoimmassa tapauksessa aikaa laajan pelin käsien lukumäärän = 5 + seuraavakäsi aikayksikköä, eli tämäkin on vakioaikainen
- Listan läpikäyminen vie aikaa 20 yksikköä sillä listan maksimikoko on 20 alkiota. Algoritmi seuraa kolmea listasolmua ja päivittää niitä. Huonoimmassa tapauksessa aikaa kuluu <20*3 yksikköä, joten listan läpikäyminen on myös vakioaikaista
- pelaajanKädet on vakiokokoinen (5 alkiota) ja alkioiden arvon kasvattaminen yhdellä on vakioaikaista. Myös suurimman arvon sisältävän alkion etsiminen on vakioaikainen.

- Sekä lista että taulukko vievät vakiomäärän tilaa riippumatta syötteestä tai pelitilanteesta. (Listan koko on kierrosten lukumäärä kunnes on pelattu 20 kierrosta, tämän jälkeen lista on jatkuvasti 20 alkion kokoinen)

Pelin muutkin luokat ja niiden metodit ovat vakioaikaisia. Selvitettäessä pelaajan seuraavaa kättä rotaatiossa, pelaajan kättä vastaavaa kokonaislukua kasvatetaan yhdellä tai saavuttaessa viimeiseen kättä kuvaavaan kokonaislukuun vaihdetaan käden arvo nollaksi.
Pelaajan kättä voittavat kädet on kovakoodattu perinteisessä pelimoodissa. Laajennetussa pelimoodissa arvotaan pelaajan käden voittava käsi kahdesta vaihtoehdosta. (Laajennetussa pelissä on aina kaksi kättä jotka voittavat yhden käden ja samaten yksi käsi voittaa kaksi muuta kättä).

Puutteet ja parannusehdotukset

Heuristiikkaa voi viedä eteenpäin monella tavalla. Tällä hetkellä tekoäly etsii pelihistorian viimeisimmistä 20:stä pelatusta kädestä nykyistä kierrosta vastaavia tilanteita ja merkitsee ylös mitä kättä pelaaja pelasi aiemmin. Jos vaihtoehtoja on useita oletetaan pelaajan pelaavan sitä vaihtoehtoa joka ilmenee useimmin. Tekoälyä voisi parantaa lisäämällä mukaan pelaajan käyttäytymismallin jossa etsitään erilaisia malleja pelaajan käsistä. Esimerkiksi tekoäly voisi löytää toistuvan kuvion jossa pelaaja pelaa aina peräkkäin kaksi kiveä ja niitä seuraa paperi.

Lähteet

Social cycling and conditional responses in the Rock-Paper-Scissors game, http://arxiv.org/pdf/1404.5199v1.pdf, 12.12.2014