

Toteutusdokumentti, Kalaha

- Yleisrakenne
- Aika- ja tilavaativuus
- Puutteet ja parannusehdotukset

Ohjelma koostuu neljästä luokasta:

Main:

Ohjelma suoritetaan tässä luokassa. Luo Board- ja Game-oliot, jotka sisältävät pelin säännöt ja kulun tarvittavine metodeineen ja muuttujineen.

Game:

Pelin kulku menee tämän luokan pelaa()-metodin mukaisesti. Tarvitsee MiniMax-olion minimax-algoritmia tietokoneen optimaalisen siirron selvittämiseen.

MiniMax:

Sisältää minimax-algoritmin, joka saa syötteenä pelaajan (1 tai 2), jonka paras mahdollinen siirto halutaan selvittää, pelilaudan, josta siirto lasketaan, tutkittavan syvyyden, joka määritetään metodin sisällä, sekä parametrit alfa ja beta, joiden avulla karsitaan turhia pelipuun oksia. Palauttaa 6-paikkaisen listan, jonka indeksien 0...5 arvot edustaa kyseisen indeksin siirrolla saavutettavaa pistemäärää. Algoritmi olettaa, että molemmat pelaavat optimaalisesti.

Board:

Tämä luokka huolehtii pelilaudan alustuksesta, tulostuksesta ja päivittämisestä tehtävän siirron mukaan, sääntöjä noudattaen.

Jokaisella luokan Board ilmentymällä on luokkamuuttujana Board olio "temp", jota käytetään simuloivan siirron yhteydessä, jotta varsinainen pelilauta ei muuttuisi sen takia. Simuloiva siirto päivittää siis vain tämän temp-laudan pelitilannetta sekä uusiVuoro-muuttujaa, joka kertoo, saako siirrolla uuden vuoron.

Luokassa on myös tarkistus sille, onko peli ohi, eli onko jomman kumman pelaajan pelikupit tyhjänä.

Aika- ja tilavaativuus:

Pelipuut ovat yleensä aikavaativuudeltaan suuria ja vain pienille, kevyille peleille suoritus onnistuu kohtuullisessa ajassa. Pelkällä minimax-algoritmillä kuusihaaraisen puun läpikäymisen aikavaativuus olisi luokkaa $O(b^d)$, missä b on haarautumisaste, joka Kalahassa on 6 ja d on puun syvyys, joka Kalahassa on arviolta keskimäärin 20-30 (perustuu omaan testailuun).

Alfa-Beta -karsinta, jota tässäkin työssä olen käyttänyt, voi pienentää aikavaativuutta parhaimmillaan luokkaan $O(b^{(d/2)})$, eli puun syvyys puolittuu. Pahimmassa tapauksessa, jossa pienimmät arvot tutkitaan ensin, on aikavaativuus sama kuin minimax:lla ilman alfa-beta -karsintaa. Olen tässä työssä rajoittanut syvyyden **XXX**, jotta ohjelma suorittaisi pelin kohtuujassa. Ensimmäisen siirron etsimiseen menee tällä syvyydellä noin **XXX** sekuntia. (syvyys 15, 5-6 sec)

Tilavaativuus on minimax-algoritmillä $O(bd)$ huolimatta siitä, onko alfa-beta -karsintaa käytetty vai ei.

Puutteet ja parannusehdotukset: