Shakkipeli

Määrittelydokumentti

Tuukka Paukkunen
tuukka.paukkunen@cs.helsinki.fi

Aineopintojen harjoitustyö: Tietorakenteet ja algoritmit
30.7.2015
Helsingin yliopisto, tietojenkäsittelytieteen laitos

Sisällysluettelo

1 Sanasto

Sana	Sanan tarkoitus
Pelitilanne	Shakkilaudalla oleva shakkinappuloiden muodostama shakin sääntöjen mukainen asetelma, eli shakkilaudan sisältö pelin ollessa käynnissä.
Siirto	Siirtyminen pelitilanteesta toiseen, jossa joko tietokone tai ihminen muuttaa shakkinappulan sijaintia laudalla ja mahdollisesti syö toisen nappulan tai tekee tornituksen, shakin sääntöjen mukaan.
Syönti	Shakkinappulan poistaminen laudalta pelin aikana shakin sääntöjen mukaisesti.
Ihminen	Shakkipelisovellusta käyttävä henkilö
Tietokone	Shakkipelisovelluksen muodostama shakkia pelaava tekoäly, pelaa ihmistä vastaan
Voitto	Tietokoneen voitto shakkipelissä
Häviö	Tietokoneen häviö shakkipelissä

Tässä määrittelydokumentissa puhutaan shakkipelin pelaamisesta shakkipelisovelluksen tekoälyn näkökulmasta, eli esimerkiksi voitto on tekoälyn voitto ja häviö on tekoälyn häviö.

2 Määrittely

2.1 Yleistä

Kurssin "Aineopintojen harjoitustyö: Tietorakenteet ja algoritmit (loppukesä)" työni aiheena on shakkipeli, jossa sovelluksen käyttäjä, eli ihminen pelaa tietokonetta vastaan perinteisellä 8x8 ruudukolla perinteisin nappuloin ja säännöin.

Tietorakenteena Shakkipeli käyttää ns. implisiittistä puuta [Flo15], joka syntyy rekursiivisen minimax-algoritmin tuloksena. Puuta ei kokonaisuudessaan tallenneta mihinkään tietorakenteeseen. Pelitilanteet talletetaan kaksiulotteisena taulukkona. Tämän puun juurena on pelin tämän hetkinen pelitilanne, eli se, missä asemassa nappulat pelin kulkiessa kullakin hetkellä ovat. Juuren lapsinoodeina ovat juuren pelitilannetta seuraavat mahdolliset pelitilanteet, eli mihin

asemiin nappuloita on sääntöjen puitteissa mahdollista liikuttaa. Näitä lapsinoodeja taas seuraavat niitä kutakin pelitilannetta seuraavat mahdolliset pelitilanteet ja niin edelleen. Tätä implisiittistä puuta kutsutaan myös pelipuuksi. Pelipuuta pyritään rakentamaan siten, että kulloisesta pelitilanteesta nähdään neljä siirtoa eteenpäin.

Shakkipelin pelimoottori käyttää apunaan minimax-algoritmia [Roo14, Wik15], jonka avulla sovellus tekee päätöksen, minkälainen siirto kussakin pelitilanteessa on hyödyllisintä tehdä. Pelimoottori ei käy koko pelipuuta läpi varsinkaan pelin alkutilanteessa pelipuun valtavan koon vuoksi, vaan pelipuu pyritään käymään läpi neljä siirtoa eteenpäin. Pelipuussa tutkittavien haarojen määrää karsitaan alpha-beta -karsinnan avulla [Roo14, Hel14, Abb13], jonka avulla voidaan jättää tutkimatta sellaiset pelipuun haarat, jotka osoittautuvat huonommiksi kuin aiemmin tutkitut haarat. Kunkin pelitilanteen edullisuus tietokoneelle selvitetään pelitilanteen arviointialgoritmilla, joka pyrkii selvittämään, minkälaisesta pelitilanteesta olisi todennäköisintä päätyä voittoon. Arviointialgoritmi ottaa huomioon laudalla olevat nappulat ja niiden sijainnit toisiinsa nähden. Näiden asioiden perusteella arviointialgoritmi antaa kullekin pelitilanteelle arvon, joka kertoo, kuinka todennäköistä ko. Pelitilanteesta on päästä voittoon. Jos pelitilanteesta pääsee seuraavalla siirrolla tietokoneen varmaan voittoon, tällöin arvo on positiivinen ääretön. Jos pelitilanteesta pääsee seuraavalla siirrolla vastustajan varmaan voittoon, tällöin pelitilanteen arvo on negatiivinen ääretön. Varman tasapelin arvo on 0. Nämä algoritmit ovat perinteinen ja tunnettu toteutustapa shakkipelille, joten valinta osui sen vuoksi näihin.

Shakkipelisovelluksessa tietokone pelaa aina valkoisilla mustilla ja ihminen aina valkoisilla nappuloilla. Shakkipelissä sovelletaan maailman shakkijärjestön sääntöjä [Fid14a]. Pelissä jo tehtyjä siirtoja ei voi perua, eli undo-toimintoa ei ole. Aloitettua peliä ei voi myöskään tallentaa, vaan peli pelataan loppuun yhdeltä istumalta.

2.2 Käyttö ja käyttöliittymä

Käyttöliittymä pidetään sovelluksessa varsin yksinkertaisena. Sovelluksella tulee olemaan tekstipohjainen käyttöliittymä, jossa shakkilauta piirretään käyttäjän nähtäville ASCII-merkein. Tietokoneen pelaamat valkoiset nappulat ovat ylhäällä ja ihmisen pelaamat mustat nappulat alhaalla. Shakkinappuloiden symboleina käytetään englanninkielen mukaisia shakkinappuloiden yksikirjaimisia lyhenteitä [Fid14b].

Siirrot tehdään kirjoittaen komentokehotteeseen komentoja, jossa annetaan lähtökoordinaatit ja maalikoordinaatit, esimerkiksi: ”d2 d4”, jolla siirretään sotilasta ruudusta d2 ruutuun d4. Mikäli komennossa annetaan siirto, joka ei shakin sääntöjen puitteissa ole mahdollinen, lähtökoordinaatti osoittaa tyhjään ruutuun tai maalikoordinaatti osoittaa laudan ulkopuolelle, sovellus antaa tästä käyttäjälle virheilmoituksen.

Sallittujen koordinaattien annon jälkeen sovellus päivittää shakkilaudan piirtämällä sen uudelleen, jolloin laudalla siirretyn nappulan sijainti on muuttunut, sekä mahdollisesti vastustajan nappula syöty. Tämän jälkeen sovellus tekee siirron omilla, mustilla nappuloillaan, jonka jälkeen taas on käyttäjän vuoro.

Alla esimerkki sovelluksen tulostamasta shakkilaudasta nappuloineen:

```
   ---------------------------------
8 | R | N | B | Q | K | B | N | R | MUSTA
   |-------------------------------|
7 | p | p | p | p | p | p | p | p |
   |-------------------------------|
6 |   |   |   |   |   |   |   |   |
   |-------------------------------|
5 |   |   |   |   |   |   |   |   |
   |-------------------------------|
4 |   |   |   |   |   |   |   |   |
   |-------------------------------|
3 |   |   |   |   |   |   |   |   |
   |-------------------------------|
2 | p | p | p | p | p | p | p | p |
   |-------------------------------|
1 | R | N | B | Q | K | B | N | R | VALKOINEN
   ---------------------------------
     A   B   C   D   E   F   G   H
```

3 Aika- ja tilavaativuudet

Minimax-algoritmin aikavaativuus on O(b^m) ja tilavaativuus on O(bm), jossa b on sallittujen siirtojen määrä (eli siirrettävissä olevien nappuloiden määrä) kussakin pelitilanteessa ja m on pelipuun korkeus (tarvittaessa rajattu tiettyyn korkeuteen).

3.1 Aikavaativuus Alpha-beta -karsinnan kera

Kun algoritmiin otetaan mukaan Alpha-Beta karsinta (Alpha-Beta pruning), tällöin huonoimmassa tapauksessa pelipuusta ei voida karsia mitään, jolloin koko pelipuu, tai sovittu määrä pelipuun tasoja, käydään edelleen läpi, jolloin aika- ja tilavaativuus säilyvät ennallaan.

Parhaassa tapauksessa kunkin solmun kohdalla tarkistetaan 2b-1 lapsisolmua. Huonoimmassa tapauksessa tarkistetaan b^2 lapsisolmua. Tämän perusteella voidaan todeta, että Minimax-algoritmin aikavaativuus Alpha-Beta karsinnan kera on O(b^(m/2)).

4 Lähteet

Flo15	Luentomoniste, Tietorakenteet ja algoritmit -kurssi, Tietojenkäsittelytieteen laitos, Helsingin yliopisto, 2015. Sivut 364-374. www.cs.helsinki.fi/u/floreen/tira2015/sivut351-638.pdf [30.7.2015]
Wik15	Minimax-algoritmia koskeva Wikipedian artikkeli. https://en.wikipedia.org/wiki/Minimax [28.7.2015]
Roo14	”Johdatus tekoälyyn” -kurssin luentomoniste, Tietojenkäsittelytieteen laitos, Helsingin yliopisto, Helsinki, 2014. Sivut 6-9. https://www.cs.helsinki.fi/webfm_send/1429 [28.7.2015]
Hel14	”Johdatus tekoälyyn 22.9.2011 – Alphabeta” -video, Tietojenkäsittelytieteen laitos, Helsingin yliopisto, Helsinki 2014. http://www.cs.helsinki.fi/video/alphabeta [28.7.2015]
Abb13	CS188 Artificial Intelligence, UC Berkeley, Spring 2013, Instructor: Prof. Pieter Abbeel, Step by Step: Alpha Beta Pruning, Youtube 2013. https://www.youtube.com/watch?v=xBXHtz4Gbdo. [28.7.2015]
Fid14a	Shakin säännöt maailman shakkijärjestön (World Chess Federation) mukaan, voimassa 1.7.2014 lähtien. http://www.fide.com/fide/handbook.html?id=171&view=article [28.7.2015]
Fid14b	Shakkinappuloiden yksikirjaimiset symbolit, kappale 2.2. http://www.fide.com/fide/handbook.html?id=171&view=article [28.7.2015]