V. KÉTSZEMÉLYES JÁTÉKOK

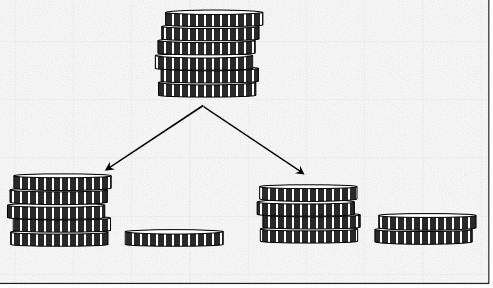
1

Teljes információjú, véges, zéró összegű kétszemélyes játékok

- Két játékos lép felváltva adott szabályok szerint.
- Mindkét játékos ismeri a maga és az ellenfele összes választási lehetőségét, és azok következményeit.
- Mindkét játékos minden lépésében véges számú lehetőség közül választhat;
  minden játszma véges lépésben véget ér.
- Amennyit az egyik játékos nyer, annyit veszít a másik. (Legegyszerűbb változatban: egyik nyer, másik veszít, esetleg lehet döntetlen is)

2

Grundy mama játéka



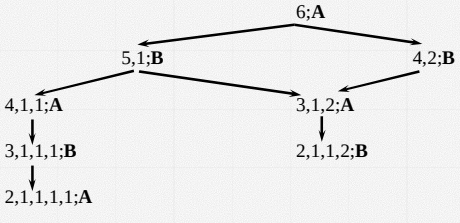
3

Állapottér-reprezentáció

- állapot - állás + soron következő játékos
- művelet - lépés
- kezdő állapot - kezdőállás-kezdő játékos
- célállapot - végállás
  (nyerő, vesztő vagy döntetlen)
- műveletsorozat - játszma (mindig véges)

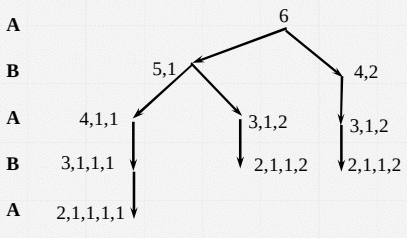
4

Grundy mama játékgráfja



5

Grundy mama játékfája



6

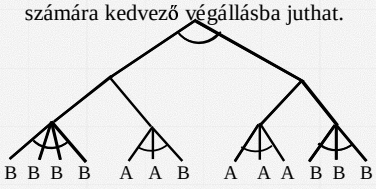
Játékfa

- csúcs - állás (egy állás több csúcs)
- szint - játékos
- él - lépés (szintről szintre)
- gyökér - kezdőállás (kezdő játékos)
- levél - végállások
- ág - játszma

7

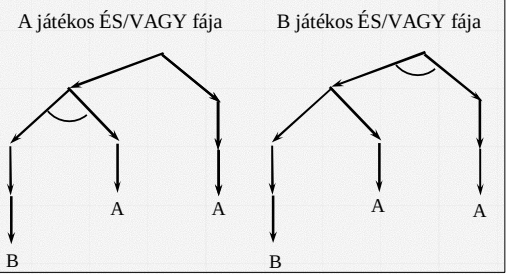
Nyerő stratégia

- Az egyik játékosnak akkor van nyerő stratégiája (vagy nem vesztő stratégiája), ha mindig tud olyat lépni, hogy ellenfele bármilyen játéka esetén számára kedvező végállásba juthat.



8

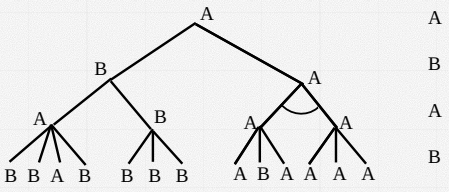
Nyerő stratégia ábrázolása



9

Tétel

- Egyik játékos számára mindig létezik nyerő stratégia (nem vesztő stratégia)



10

Részleges játékfa-kiértékelés

- Minimax algoritmus
- Negamax algoritmus
- Átlagoló kiértékelés
- Váltakozó mélységű
- Szelektív kiértékelés
- Alfabéta algoritmus

11

Kiértékelő függvény

- Minden esetben szükségünk van egy olyan heurisztikára, amely megbecsüli, hogy egy állás mennyire ígéretes.
- Sokszor ez egy *f*:állás → [-100..100] függvény

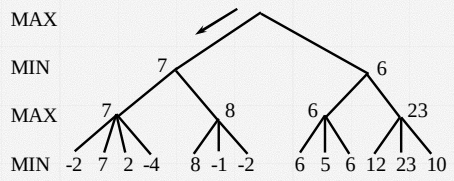
12

Minimax algoritmus

- Adott állásból indulva felépítjük a játékfa néhány szintjét
- A részfa leveleit kiértékeljük aszerint, hogy azok számunkra kedvező, vagy kedvezőtlen állások.
- Az értékeket felfuttatjuk a fában. (Saját szintjeink csúcsaihoz azok gyermekeinek maximumát, ellenfél csúcsaihoz azok gyermekeinek minimumát rendeljük.)
- Soron következő lépésünk ahhoz az álláshoz vezet, ahonnan a gyökérhez felkerült a legnagyobb érték.

13

Példa



14

Megjegyzés

- Az algoritmust minden alkalommal, valahányszor mi következünk, megismételjük, hiszen lehet, hogy az ellenfél nem az általunk várt legerősebb lépésekkel válaszol, mert:
  - eltérő mélységű részfával dolgozik
  - más kiértékelő függvényt használ
  - nem minimax eljárást alkalmaz
  - hibázik

15

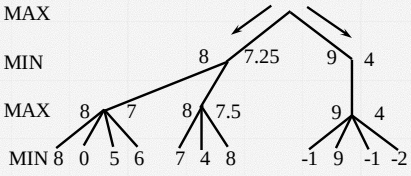
Negamax algoritmus

- Negamax eljárást könnyebb implementálni.
  - Az ellenfél szintjén levő levelek értékének vesszük a (-1)-szeresét, majd
  - minden szinten szülő=max(-gyerek$_1$,..., -gyerek$_n$)

16

Átlagoló kiértékelés

- Az (*m*,*n*) átlagolás célja a kiértékelő függvény esetleges tévedéseinek simítása.
- Legyen például *n=2* és *m=2*.



17

Váltakozó mélységű kiértékelés

- Célja, hogy a kiértékelő függvény minden ágon reális értéket mutasson.
- A részfa felépítését módosítjuk úgy, hogy egy adott szinttől kezdve, akkor vesszük bele egy csúcs utódait a részfába, ha minden utódra teljesül a nyugalmi teszt:
  - $|f(\text{szülő}) - f(\text{gyerek})| < K$

18

Szelektív kiértékelés

- Célja a memória-igény csökkentése.
- Elkülönítjük a lényeges és lényegtelen lépéseket, és csak a lényeges lépéseknek megfelelő részfát építjük fel.

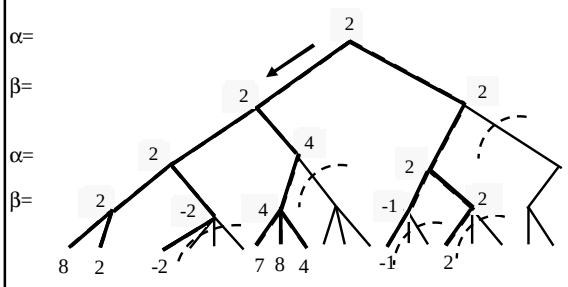
19

Alfa-béta algoritmus

- Visszalépéses algoritmus által járjuk be a részfát. (mélységi bejárás) Az aktuális úton fekvő csúcsokat:
  - a mi szintünkön α értékkel (ennél rosszabb értékű állásba innen már nem juthatunk),
  - az ellenfelén β értékkel(ennél jobb értékű állásba onnan már nem juthatunk) látjuk el.
- Lefelé haladva $\alpha=-\infty$, és $\beta=+\infty$.
- Ezek visszalépéskor a felhozott értékre változnak, ha az nagyobb, mint az α, illetve kisebb, mint a β.
- Vágás: ha az úton van olyan α és β, hogy $\alpha \geq \beta$.

20

Példa



Eredmény

- Több egyformán jó kezdőirány esetén a baloldalit kell választani.
- Ekkor ugyanazt a kezdőlépést kapjuk eredményül, mint a minimax algoritmussal talált baloldali legjobb kezdőlépés.

22

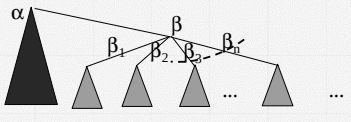
Hatékonyság

- Tárigény: csak egy utat tárol
- Futási idő: a vágások miatt sokkal jobb, mint a minimax módszeré
  - Optimális eset: egy *d* mélységű *b* elágazású fában kiértékelt levélcsúcsok száma: $\sqrt{b^d}$
  - Átlagos eset: egy csúcs alatt, két belőle kiinduló ág megvizsgálása után már vághatunk.
  - Jó eset: A bejárt részfa megfelelő rendezésével érhető el. („cáfoló lépés elve”)

23

Tétel

- Egy csúcsból kiinduló ágak közül várhatóan elég kettőt megvizsgálni, után már vághatunk.



24

Bizonyítás

- $M(n$ ág esetén hányadik ág vizsgálata után vágunk$) =$
  $= \Sigma_{k=1..n}\ k*P(k$-adik ág után $\alpha \geq \beta_k) + n*P($nincs vágás$)$
  - $P(k$-adik ág után $\alpha \geq \beta_k) = P(\alpha < \beta_1) \ldots P(\alpha < \beta_{k-1})\ P(\alpha \geq \beta_k)$
- ha $P(\alpha < \beta_i) = p \quad (p<1)$ akkor
  - $P(k$-adik után vágás$) = p^{k-1}(1-p) \quad - P($nincs vágás$) = p^n$
- $M(p, n) = \Sigma_{k=1..n-1}\ k\ p^{k-1}(1-p) + n\ p^n$
- $M(p, \infty) = \Sigma_{k=1..\infty}\ k\ p^{k-1}(1-p) = (1-p)\Sigma\ k\ p^{k-1} = (1-p)*sum =$
  - $sum = \Sigma_{k=1..\infty}\ k\ p^{k-1} = \Sigma_{k=0..\infty}\ (k+1)\ p^k$
  - $p*sum = \Sigma_{k=1..\infty}\ k\ p^k = \Sigma_{k=0..\infty}\ k\ p^k$
  - $= sum - p*sum = \Sigma_{k=0..\infty}\ p^k = 1/(1-p)$
- $M(0.5, \infty) = 2$

Tétel

- Egy d mélységű b elágazású fában a kiértékelt levélcsúcsok száma
  - $d=2k-1$ esetén $b^k + b^{k-1} - 1$
  - $d=2k$ esetén $2b^k - 1$
- Bizonyítás.
- Teljes indukció k szerint

k=1

$d=1$ illetve $d=2$ esetén

$d=1$: A szint
B szint

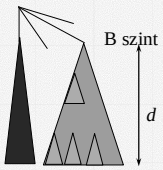
Itt a B szint összes (b darab) levelét ki kell értékelni
$b^k + b^{k-1} - 1 = b^1 + b^{1-1} - 1 = b$

$d=2$: A szint
B szint

Itt a baloldali ág összes levelét (b darab), majd a többi ágnakegy-egy levelét ($b-1$ darab) kell kiértékelni.
$2b^k - 1 = 2b^1 - 1$

Lemma

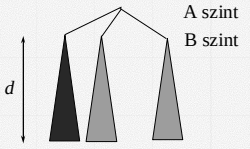
Egy olyan d mélységű részfában, amelynek gyökere felett van már ideiglenes érték, optimális esetben csak b^k darab levélcsúcsot kell kiértékelni.



Az A szinteken az összes (b darab) ágat ellenőrizni kell, a B szinteken - optimális esetben - csak az elsőt.

Ha $d=2k$, akkor $d/2$,
azaz k darab A jelű szint van
ha $d=2k-1$, akkor $(d+1)/2$,
azaz szintén k darab.

k→k+1



Kiértékelt levelek száma = ind. felt.+ $(b-1)$*lemma
$d = 2*(k+1)-1 = 2*k+1$, ekkor az előző szint páros
$2*b^k - 1 + (b-1)*b^k = b^{k+1} + b^k - 1$
$d = 2*(k+1) = 2*k+2$, ekkor az előző szint páratlan
$b^{k+1} + b^k - 1 + (b-1)*b^k = 2*b^{k+1} - 1$

Kétszemélyes játékot játszó program

- Váltakozó mélységű, szelektív, (m,n) átlagoló, negamax alfa-béta kiértékelést végez
- Keretprogram, amely váltakozva fogadja a felhasználó lépéseit, és generálja a számítógép lépéseit
- Kiegészítő funkciók (beállítások, útmutató, segítség, korábbi lépések tárolása, mentés stb.)
- Felhasználói felület, grafika
- Heurisztika megválasztása (kiértékelő függvény, szelekció, kiértékelés sorrendje)