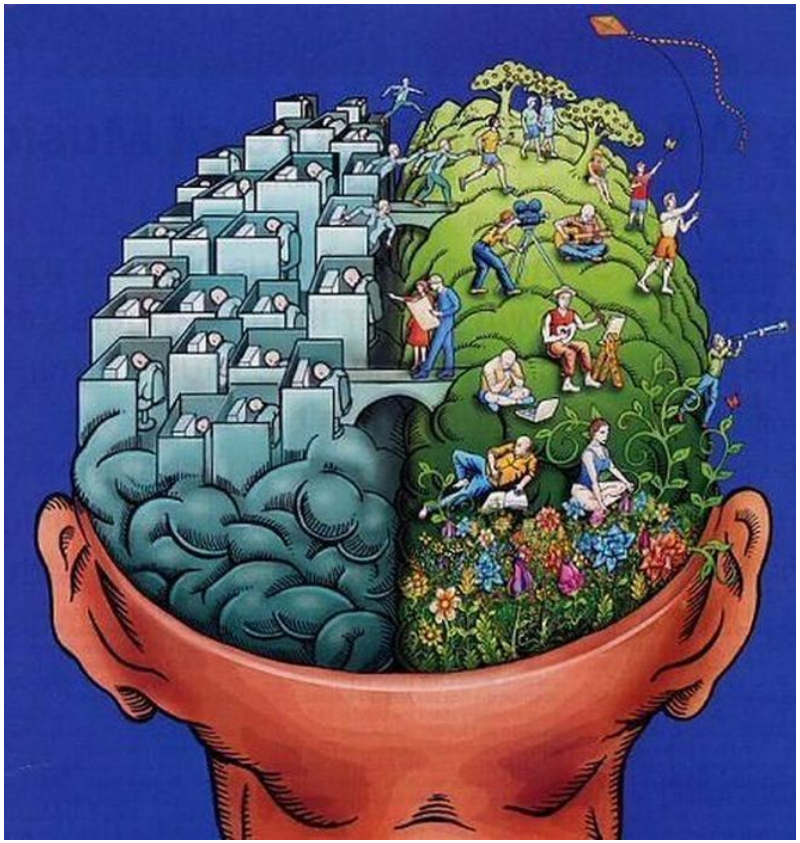
Mesterséges Intelligencia MI

Keresés ellenséges környezetben



Dobrowiecki Tadeusz
Eredics Péter, és mások

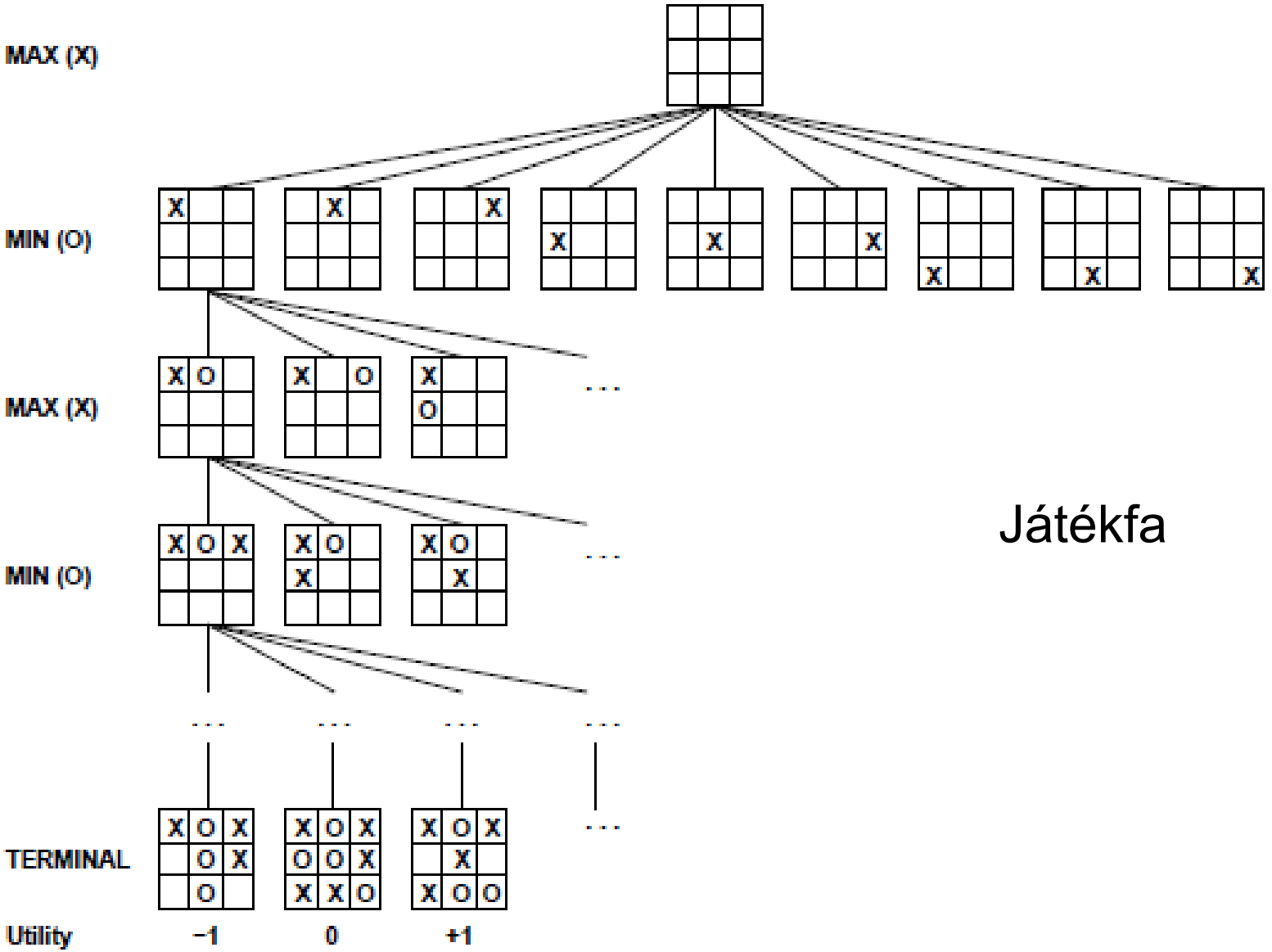
BME I.E. 437, 463-28-99
dobrowiecki@mit.bme.hu,
http://www.mit.bme.hu/general/staff/tade

Ellenség elleni játékok, kétszemélyes játékok, ...

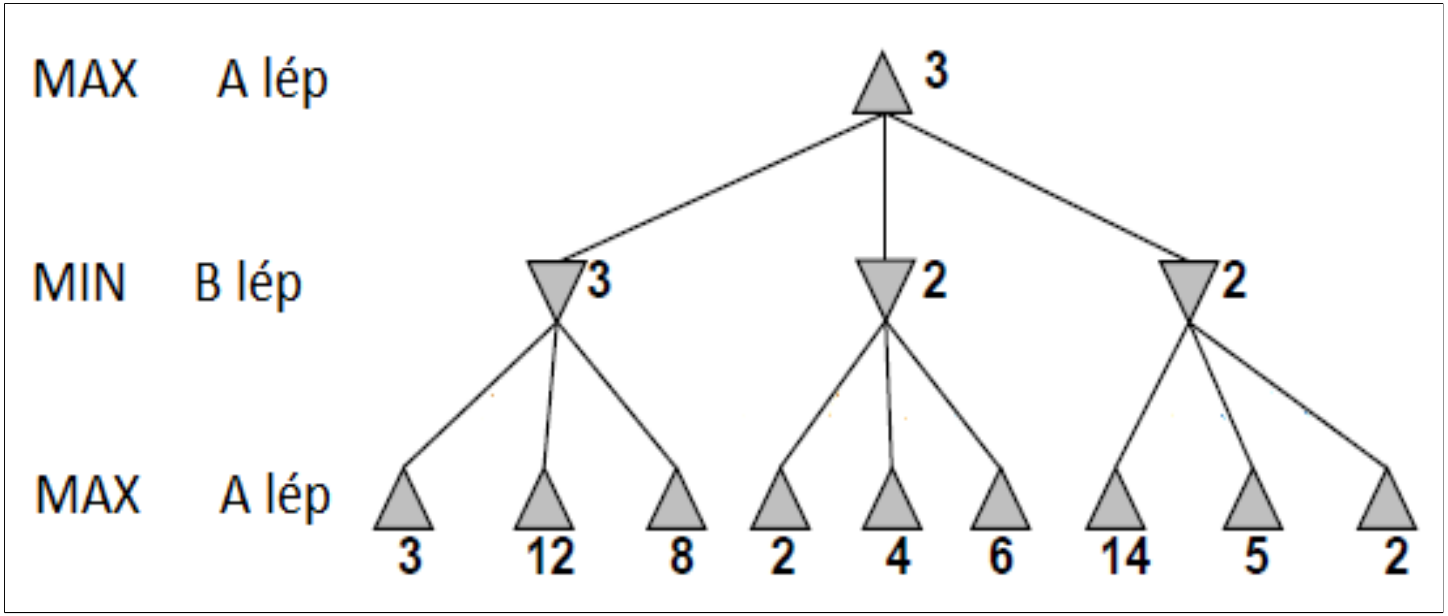
Probléma: nem csak mi lépünk, az ellenség is lép
nem tudjuk, mit fog lépni
mindenre fel kell készülni megfelelő válaszlépéssel,
azaz stratégiával

	determinisztikus	véletlen
teljes információ	**sakk, dáma, go, …**	**ostábla, Monopoly ...**
nem teljes információ	**csatahajók, ...**	**bridzs, póker, atomháború, ...**

(l. korábban az „eshetőségi” problématípus)
zérus-összegű, vagy sem?
keresés eredménye = stratégia
= opt/jó válasz az ellenség minden lépésére



Játékfa



Minimax gondolata

Tulajdonságai

Teljes? Igen, ha a fa véges.

Optimális? Igen, egy optimálisan játszó ellenféllel szemben.

Különben?

Időkomplexitás? $O(b^m)$

Tár? O(bm) (mélységi jelleg miatt)

(sakk: b = 35, m = 100 ? $O(10^{154})$

(de valóban szükséges minden ágat feltárni?)

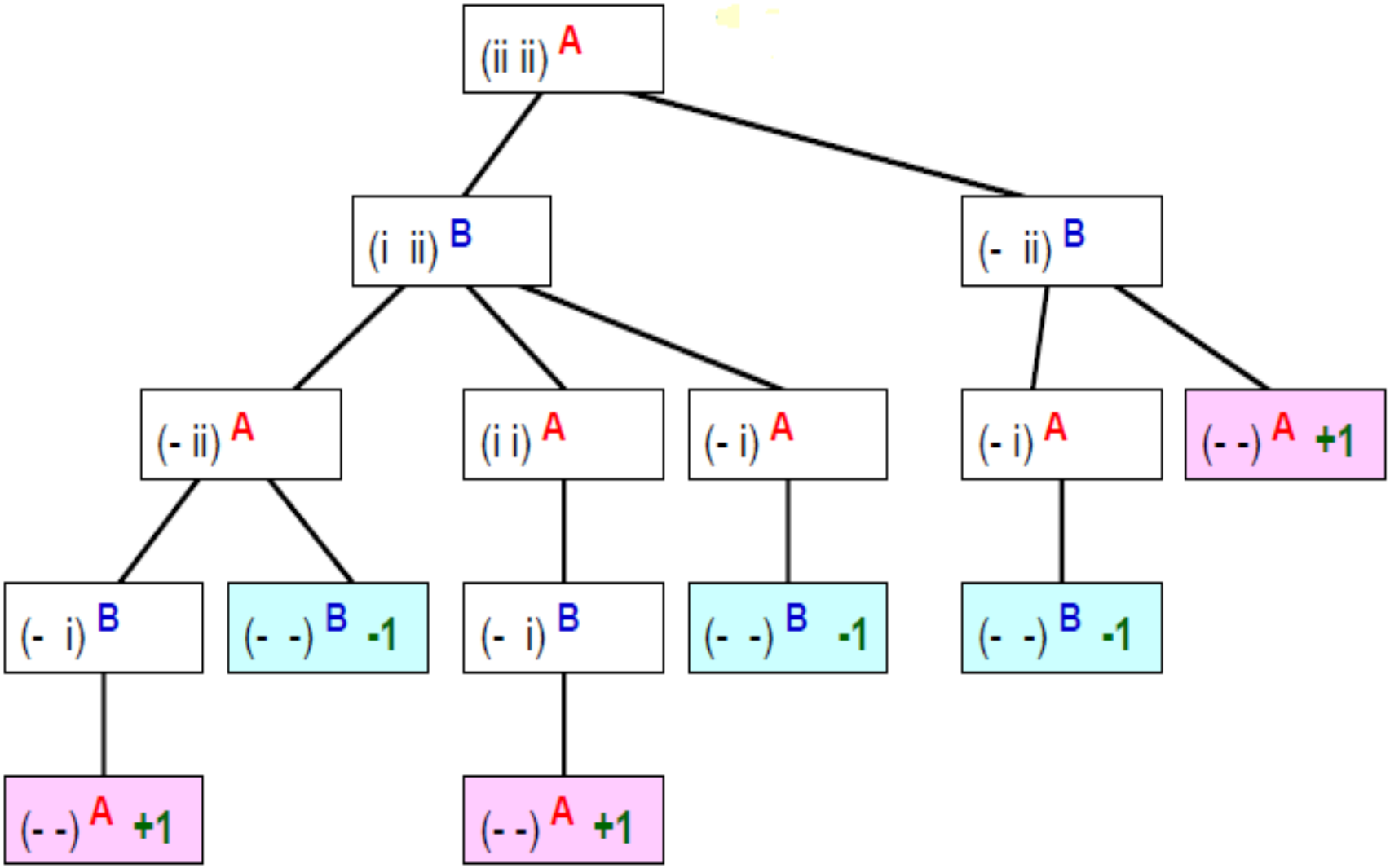
```
MinimaxValue(S)=
    If (S is a terminal)
        return V(S)
    Else
        Let { S1, S2, … Sk } = Succs(S)
        Let vi = MinimaxValue(Si) for each i
        If Player-to-move(S) = A
            return max_{i∈{1,2…k}} Vi
        else
            return min_{i∈{1,2…k}} Vi
```

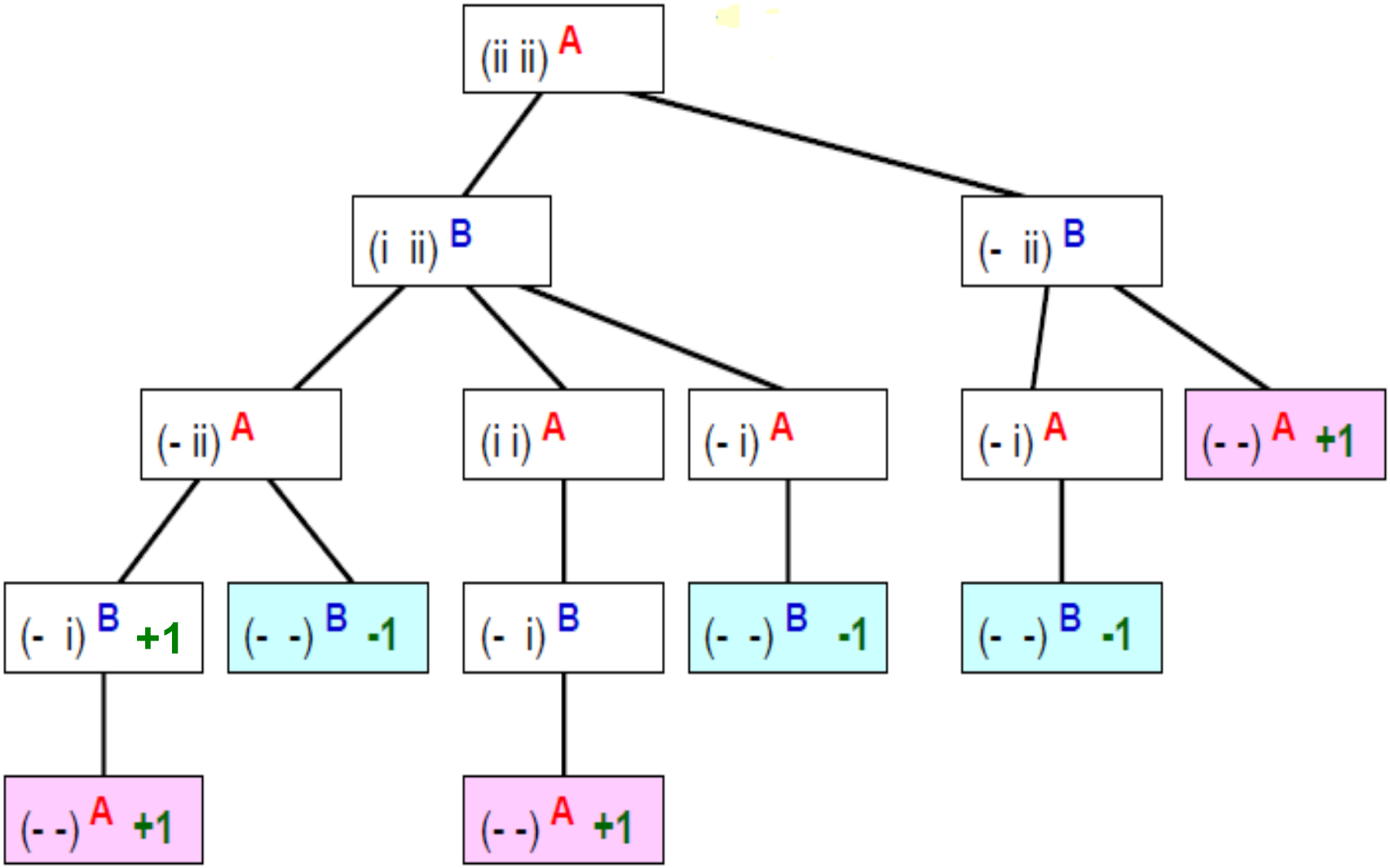
Nim

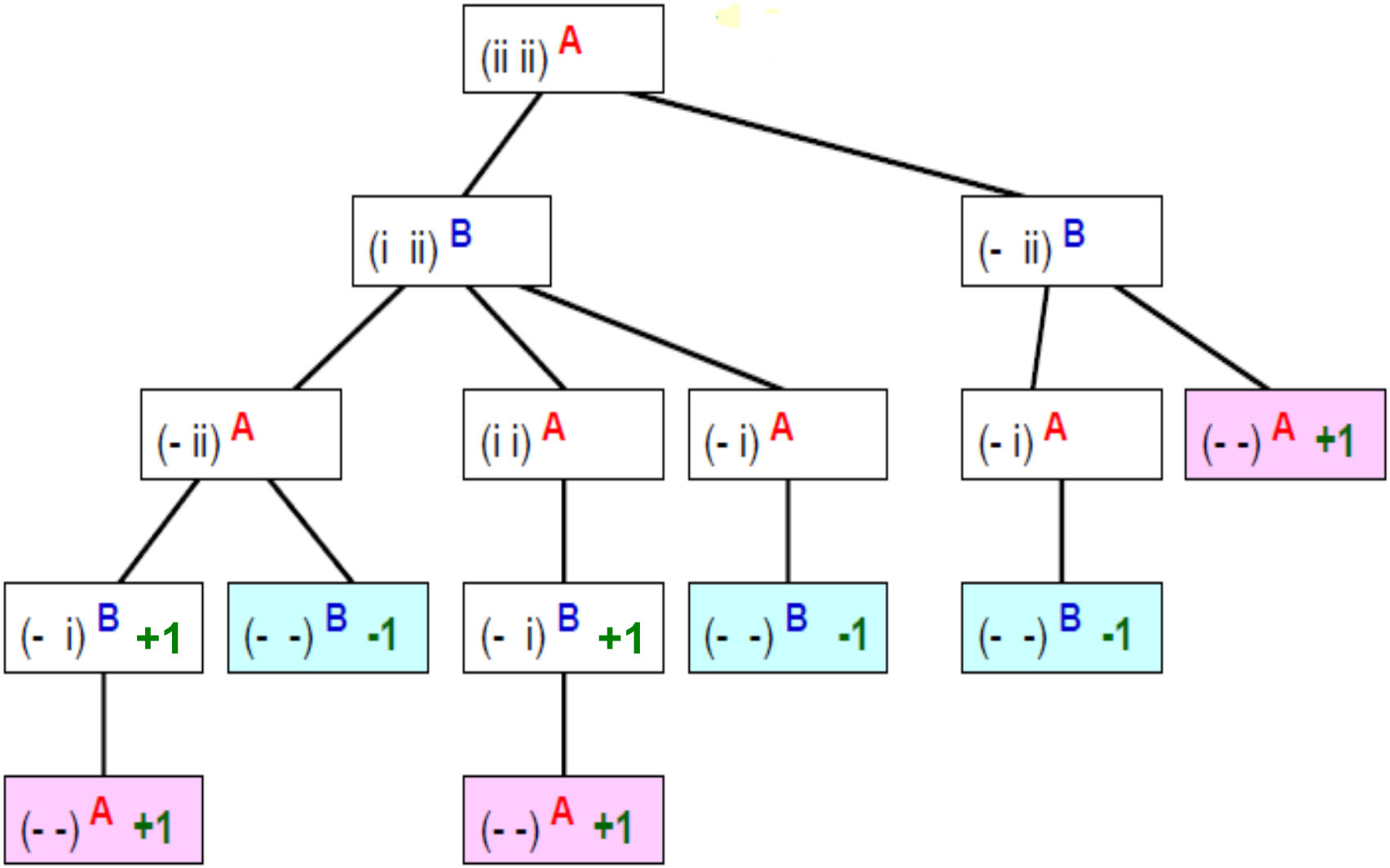
1. Bizonyos számú gyufakupac.
2. Egy lépésben egy játékosnak szabad tetszőleges sz. gyufát, de csakis egyetlen egy kupacról leemelni.
3. Aki az utolsó gyufát emel el, veszít.

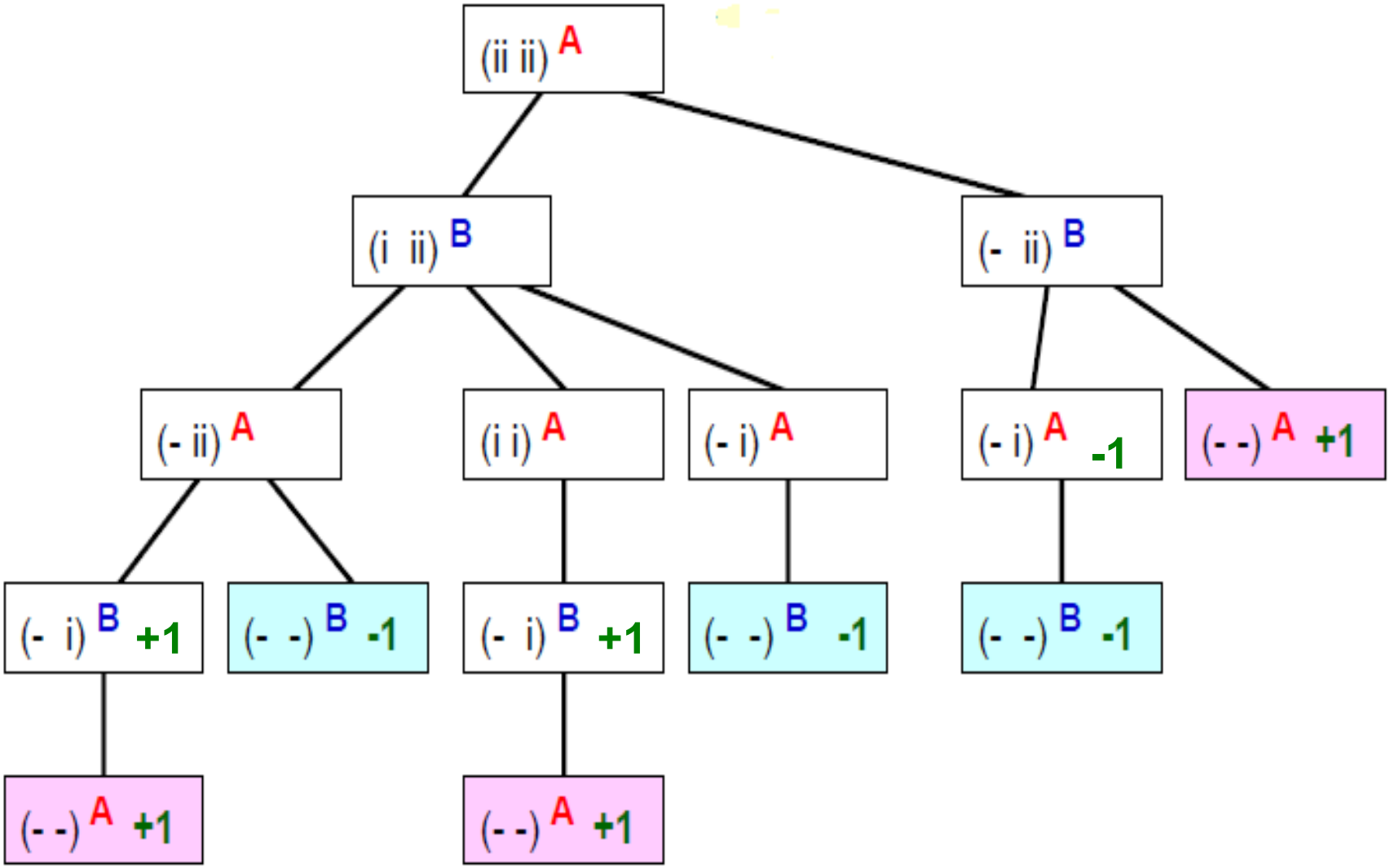
II-Nim

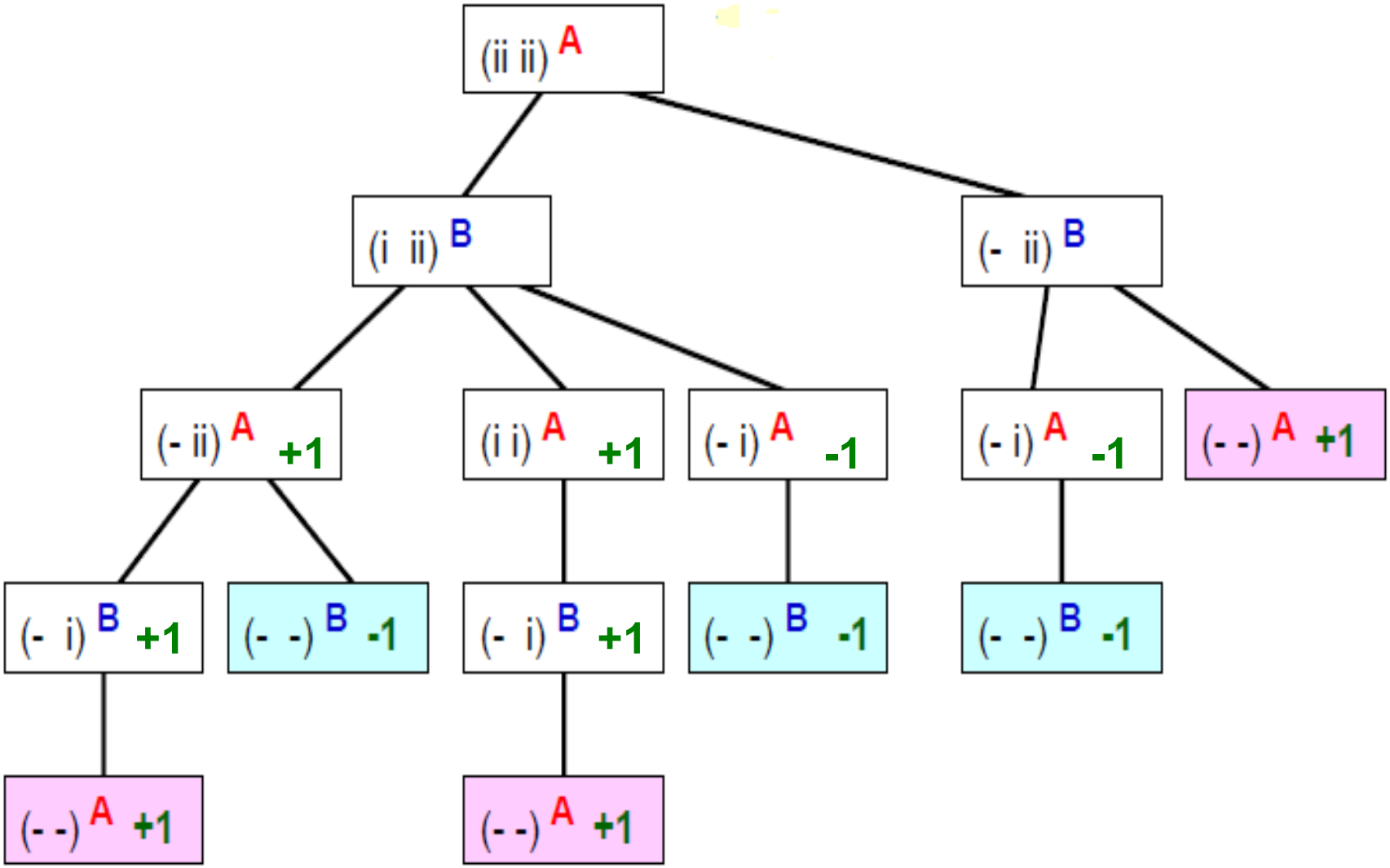
S	=	a finite set of states (note: state includes information sufficient to deduce who is due to move)	(_ , _)-A (_ , i)-A (_ , ii)-A (i , i)-A (i , ii)-A (ii , ii)-A (_ , _)-B (_ , i)-B (_ , ii)-B (i , i)-B (i , ii)-B (ii , ii)-B	
I	=	the initial state	(ii , ii)-A	
Succs	=	a function which takes a state as input and returns a set of possible next states available to whoever is due to move	Succs(_,i)-A = { (_,_)-B } Succs(_,ii)-A = { (_,_)-B , (_,i)-B } Succs(i,i)-A = { (_,i)-B } Succs(i,ii)-A = { (_,i)-B (_,ii)-B (i,i)-B} Succs(ii,ii)-A = { (_,ii)-B , (i,ii)-B }	Succs(_,i)-B = { (_,_)-A } Succs(_,ii)-B = { (_,_)-A , (_,i)-A } Succs(i,i)-B = { (_,i)-A } Succs(i,ii)-B = { (_,i)-A , (_,ii)-A (i,i)-A } Succs(ii,ii)-B = { (_,ii)-A , (i,ii)-A }
T	=	a subset of S. It is the terminal states	(_ , _)-A	(_ , _)-B
V	=	Maps from terminal states to real numbers. It is the amount that A wins from B.	V(_ , _)-A = +1	V(_ , _)-B = -1

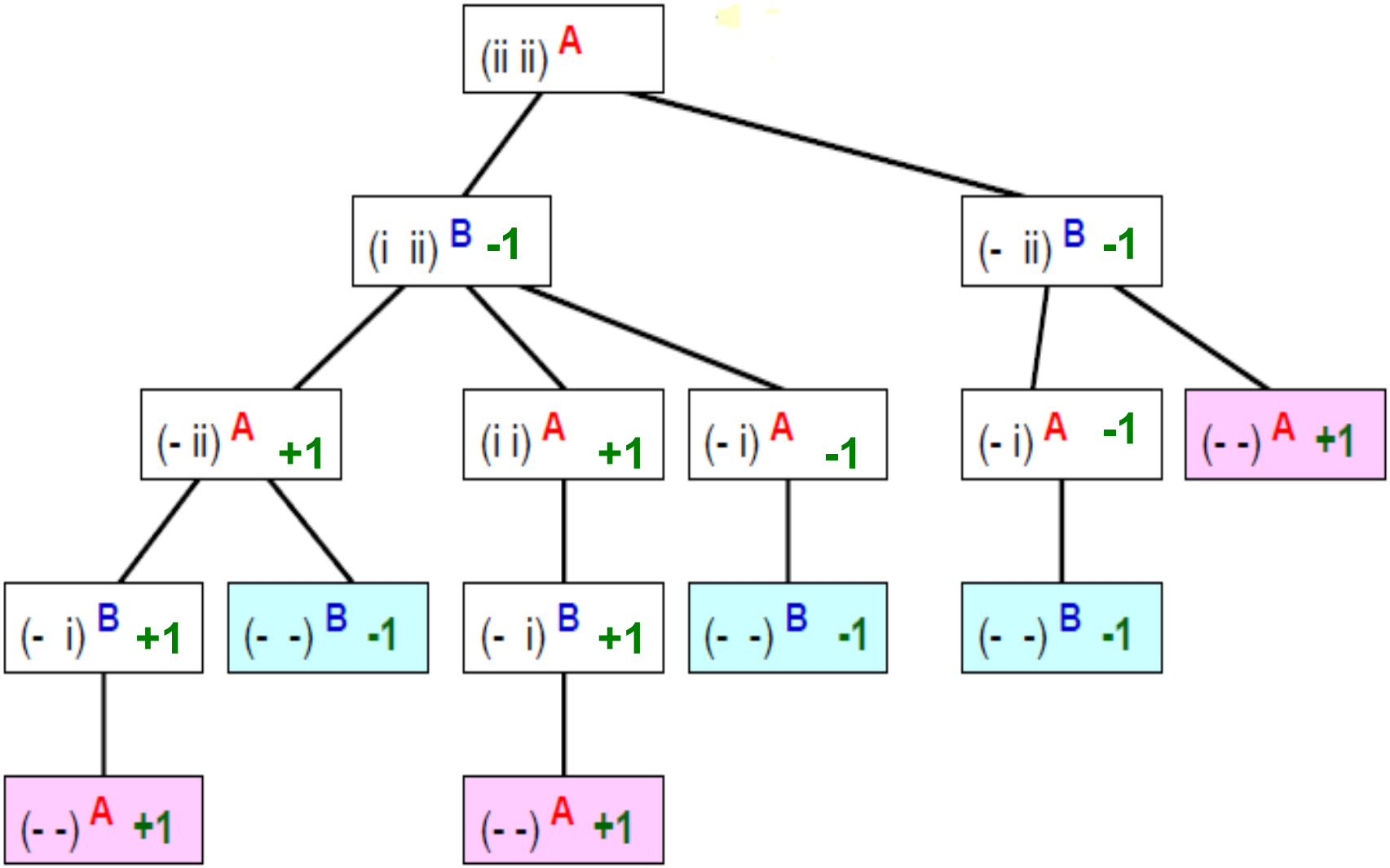

(ii ii) A
(i ii) B
(- ii) B
(- ii) A
(i i) A
(- i) A
(- i) A
(- -) A +1
(- i) B
(- -) B -1
(- i) B
(- -) B -1
(- -) B -1
(- -) A +1
(- -) A +1

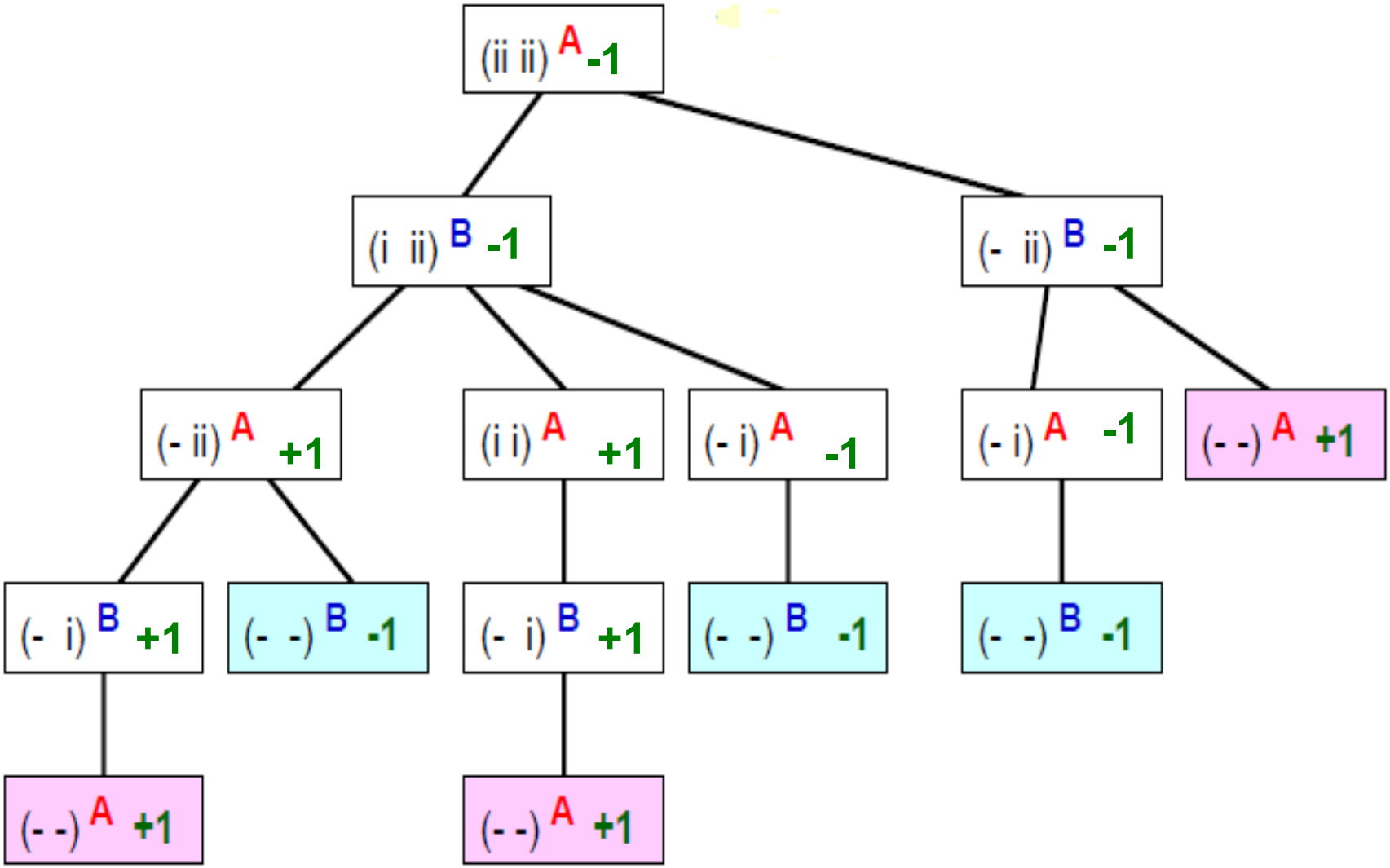

(ii ii) A
(i ii) B
(- ii) B
(- ii) A
(i i) A
(- i) A
(- i) A
(- -) A +1
(- i) B +1
(- -) B -1
(- i) B
(- -) B -1
(- -) B -1
(- -) A +1
(- -) A +1


(ii ii) A
(i ii) B
(- ii) B
(- ii) A
(i i) A
(- i) A
(- i) A
(- -) A +1
(- i) B +1
(- -) B -1
(- i) B +1
(- -) B -1
(- -) B -1
(- -) A +1
(- -) A +1


(ii ii) A
(i ii) B
(- ii) B
(- ii) A
(i i) A
(- i) A
(- i) A -1
(- -) A +1
(- i) B +1
(- -) B -1
(- i) B +1
(- -) B -1
(- -) B -1
(- -) A +1
(- -) A +1

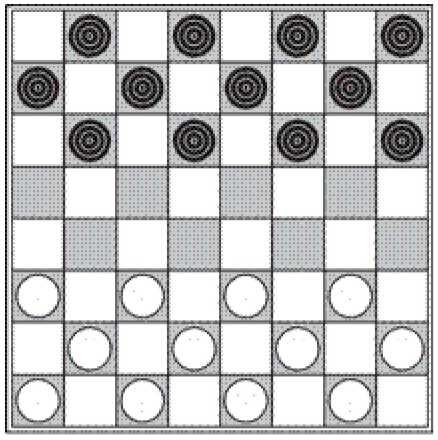

(ii ii) A
(i ii) B
(- ii) B
(- ii) A +1
(i i) A +1
(- i) A -1
(- i) A -1
(- -) A +1
(- i) B +1
(- -) B -1
(- i) B +1
(- -) B -1
(- -) B -1
(- -) A +1
(- -) A +1


(ii ii) A
(i ii) B -1
(- ii) B -1
(- ii) A +1
(i i) A +1
(- i) A -1
(- i) A -1
(- -) A +1
(- i) B +1
(- -) B -1
(- i) B +1
(- -) B -1
(- -) B -1
(- -) A +1
(- -) A +1



(A játék megoldása: Aki kezd, az veszít, ha az ellenfél optimálisan játszik)

Dámajáték megoldva, **hibátlan játék döntetlenhez** vezet!
Dámajáték keresési tere kb. 5 x 10^{20} pozició.



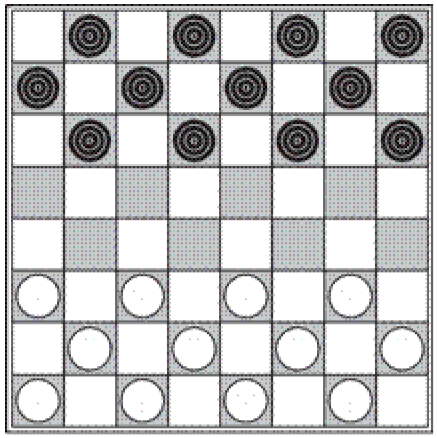
Történelem

1950 - Arthur Samuel öntanuló programja
1963 – 1 győztes parti tehetséges emberjátékos ellen
1989 - Chinook projekt, legyőzni emberi világbajnokot
1990 - Chinook jogosult Világbajnokságban részt venni
1992 - Marion Tinsley világbajnok a világbajnoki címért folytatott mérkőzésen nehezen felül kerekedik
1994 – visszavágó, Tinsley egészségi állapota miatt visszalép, rövidesen meghal
1996 - Chinook minden emberi játékosnál erősebb

A számítások története

1989 – 2007
1992 csúcs: több, mint 200 processzor
2006-2007 átlagosan 50 processzor

Az egyik leghosszabb folyamatosan futtatott elosztott számítás ezidáig



Számítások és az informatikai rendszerek megbízhatósága

Nagy méretű adatfájlok gyakori mozgatása helyi diszkre, hálózaton más gépre hibák évente többször (duplikálás, ellenőrzés)

Kiszámított kész adatbázisok periodikus ellenőrzése adatvesztés veszélye miatt (“bit rot”), másolatmentés, újraszámítás

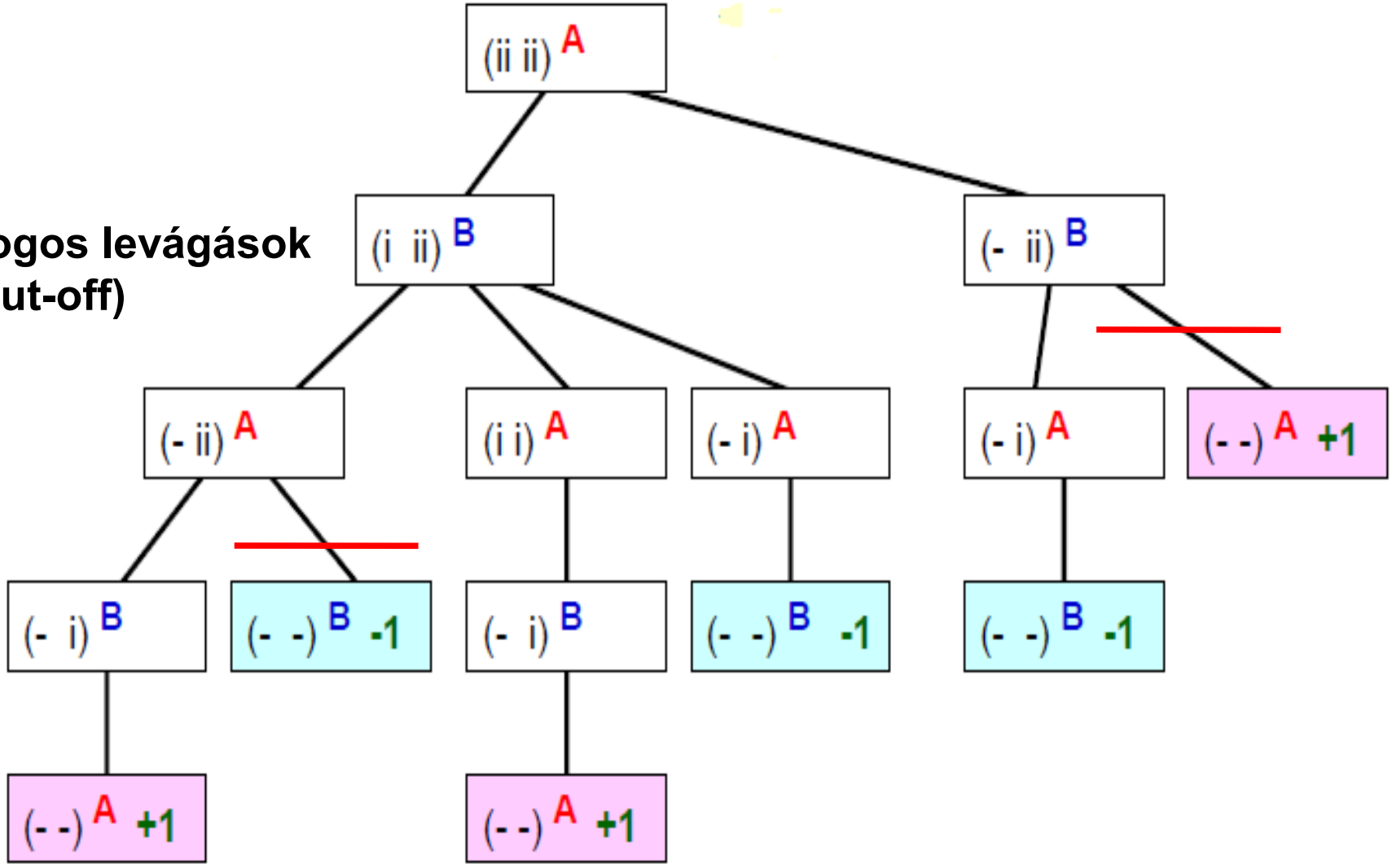
diszk gyártók jótállása – 10^{-13} hibaarány számítások annál sokkal bonyolultabbak

http://webdocs.cs.ualberta.ca/~chinook/

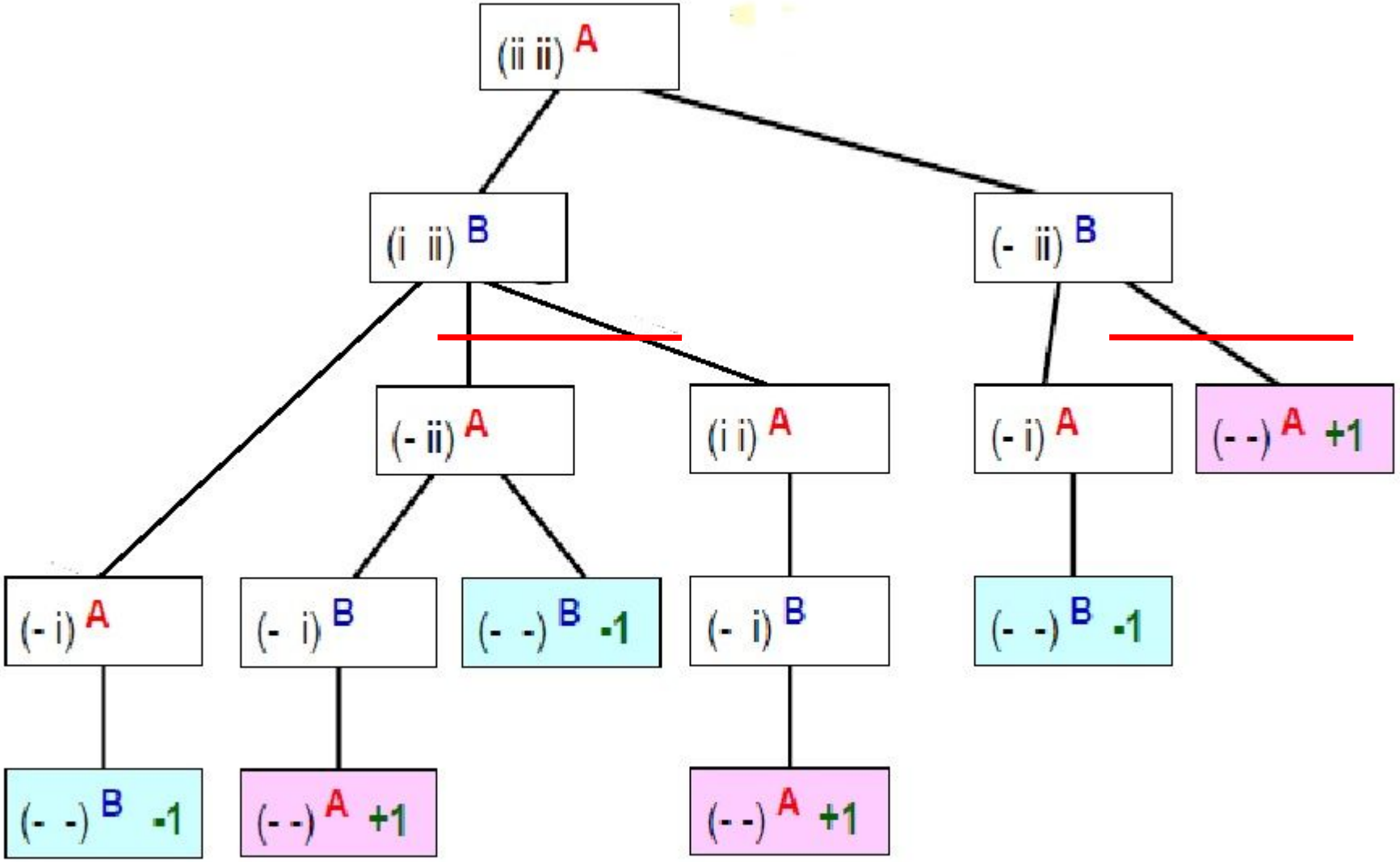
Minimax algoritmus

Tegyük fel, hogy tudjuk, hogy a játék kimenetele csakis -1 és 1.
Spórolhatunk-e a számításokból? És ha valós?

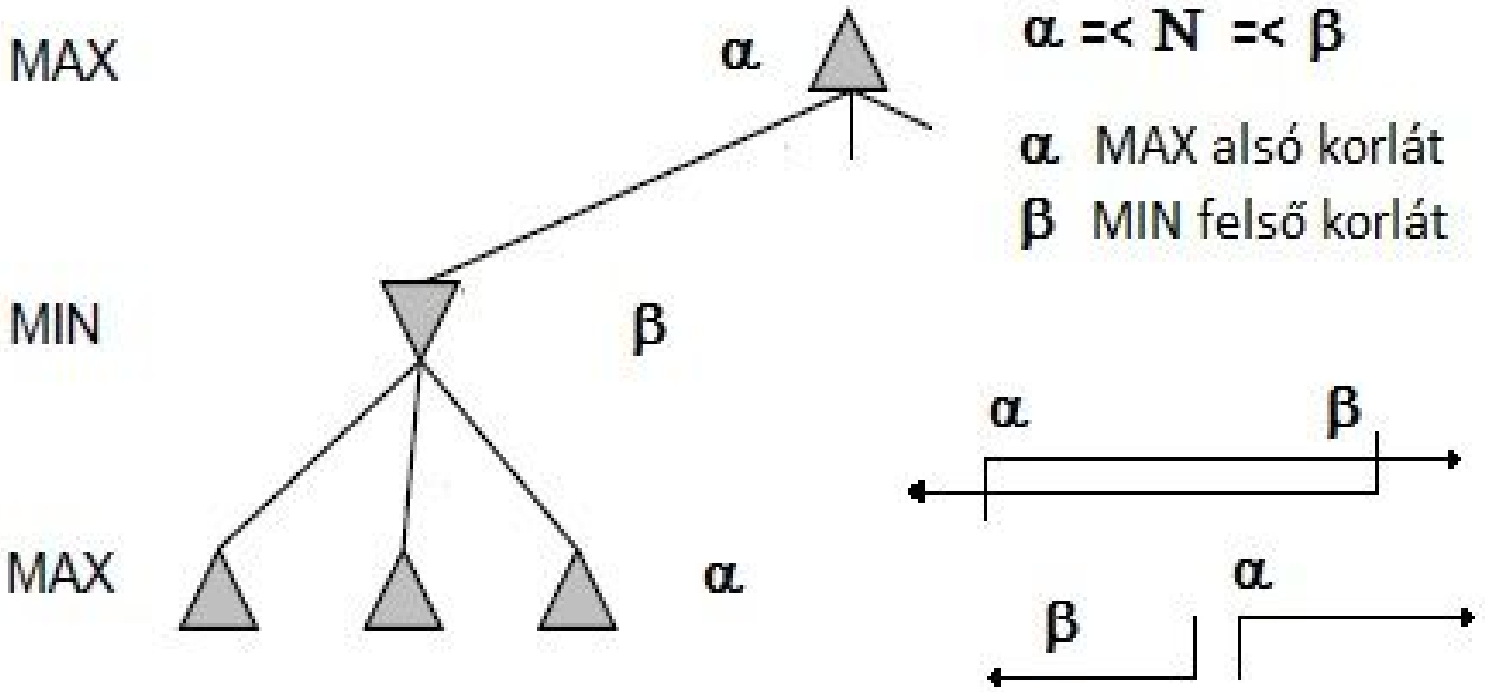
Jogos levágások (cut-off)



Minimax algoritmus



Minimax és alfa-béta nyesés



Minimax és alfa-béta nyesés

Nyesés a végeredményt nem befolyásolja.

Jó lépésrendezés fokozza a nyesés hatékonyságát

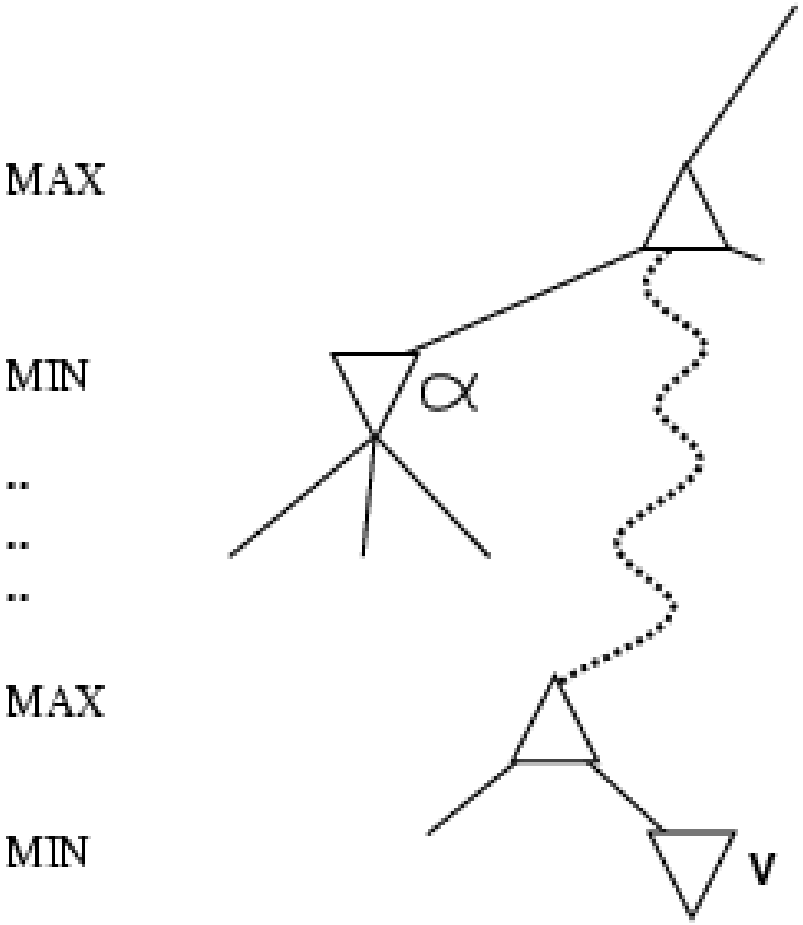
"Tökéletes rendezéssel" az időkomplexitás = $O(b^{m/2})$

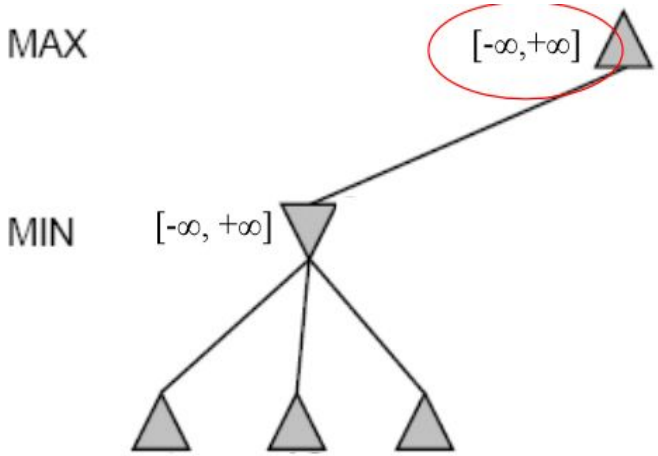
→ a keresés mélységét duplázza
(sakknál b = 35 … b = kb. 6)

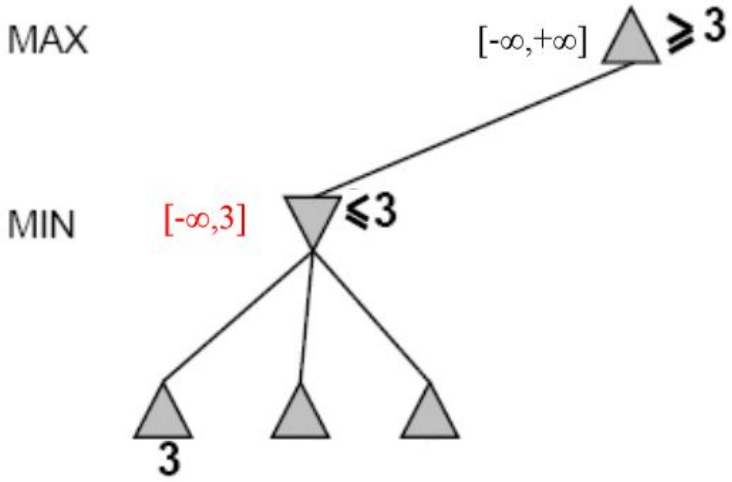
α a MAX számára eddig megtalált legjobb (legmagasabb) választás a MAX-hoz vezető pálya mentén

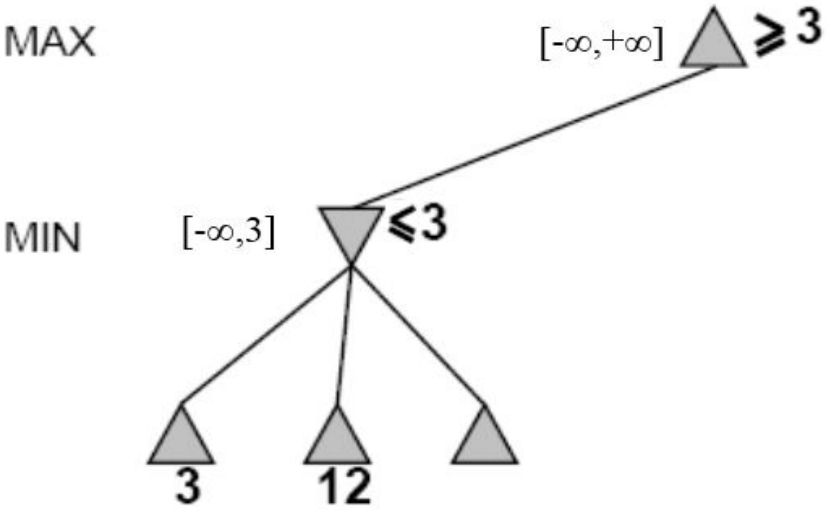
Ha *v* az α-nál rosszabb, MAX ezt az ágat elkerüli → nyesés

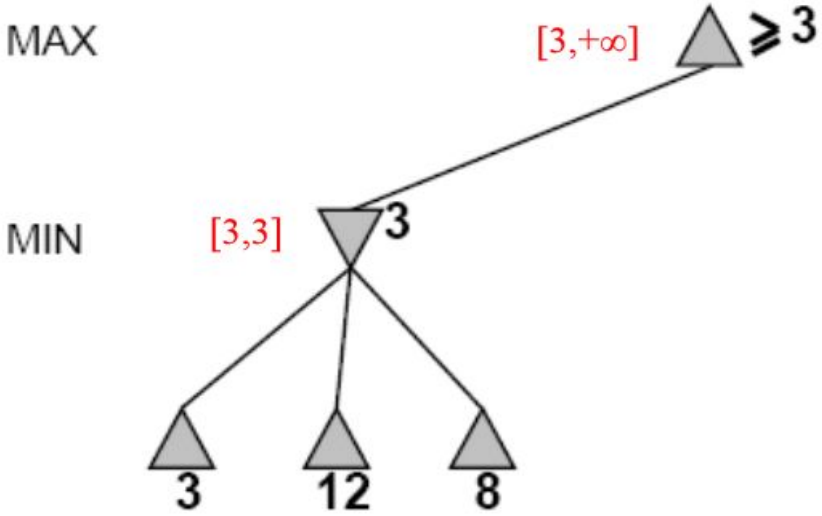
β hasonlóan a MIN számára



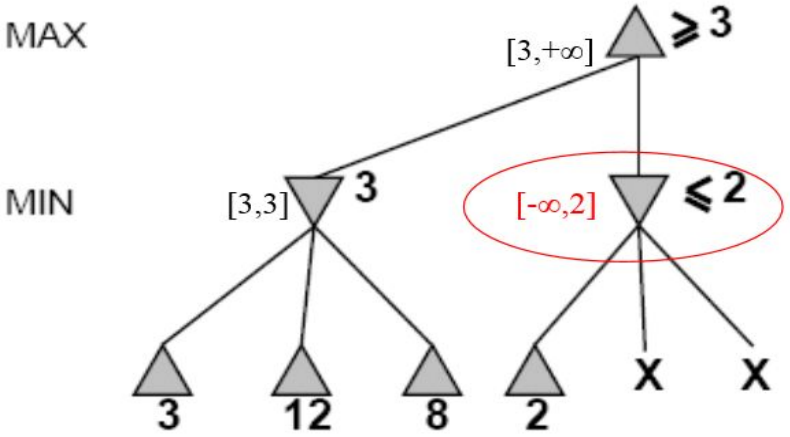


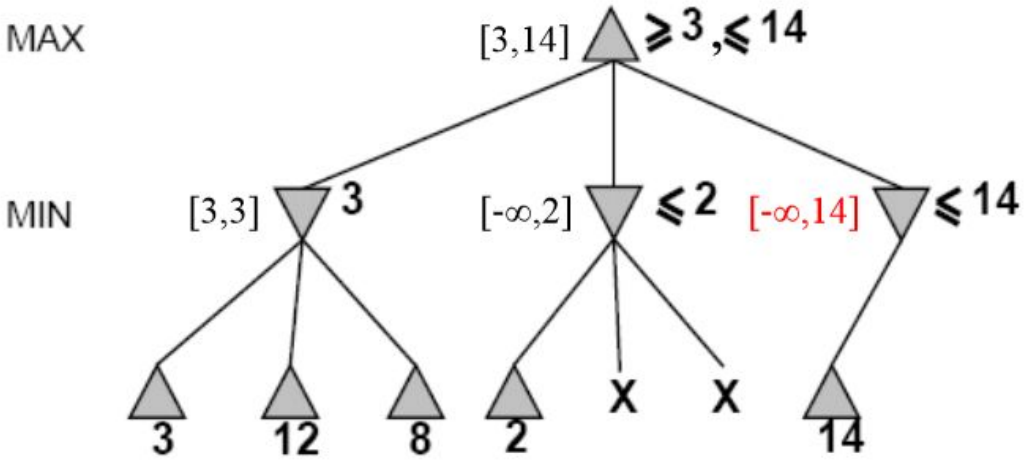


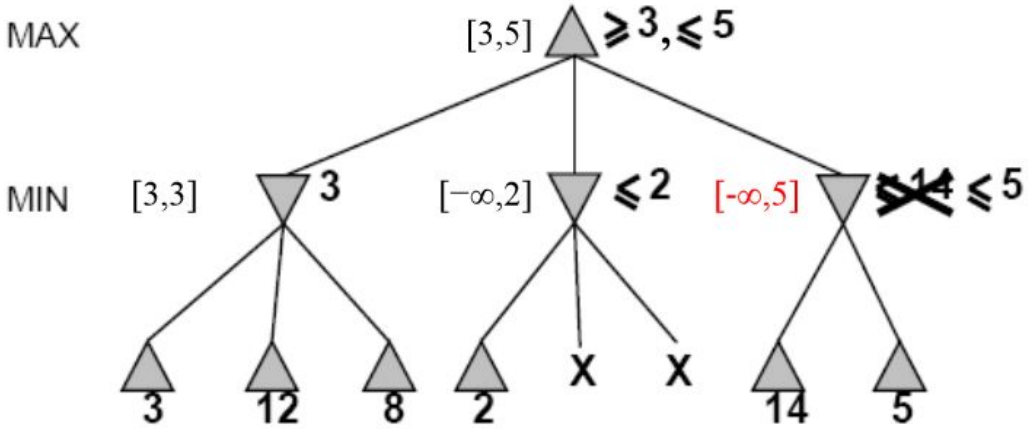


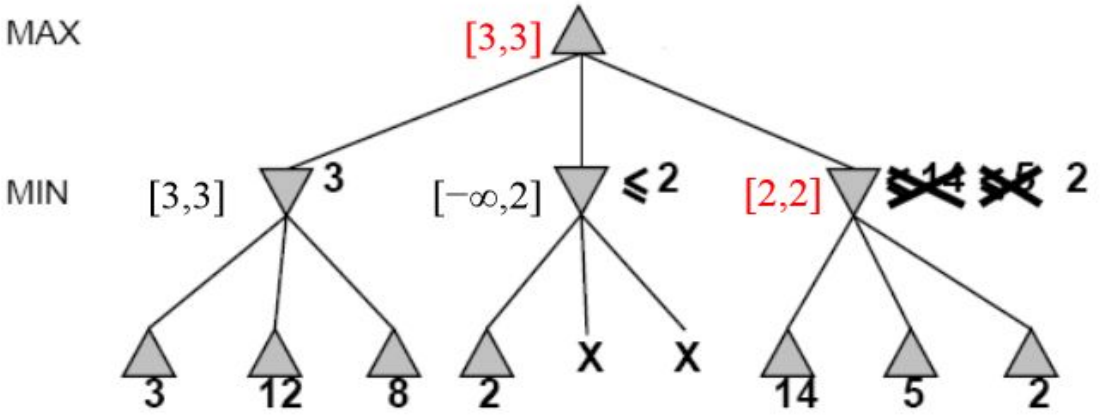


http://www.cs.ucla.edu/~rosen/161/notes/alphabeta.html


MAX
MIN
[3,+∞]
⩾3
[3,3]
3
[-∞,2]
⩽2
3
12
8
2
X
X


MAX
MIN
[3,14]
⩾3,⩽14
[3,3]
3
[-∞,2]
⩽2
[-∞,14]
⩽14
3
12
8
2
X
X
14

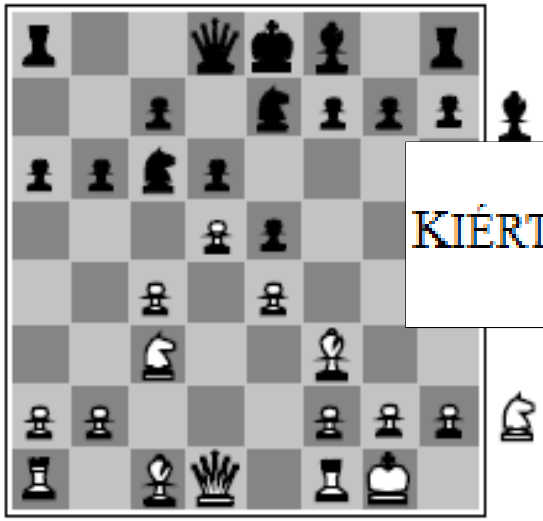

MAX
MIN
[3,5]
⩾3,⩽5
[3,3]
3
[−∞,2]
⩽2
[-∞,5]
⩽5
3
12
8
2
X
X
14
5


MAX
MIN
[3,3]
[3,3]
3
[−∞,2]
⩽2
[2,2]
2
3
12
8
2
X
X
14
5
2

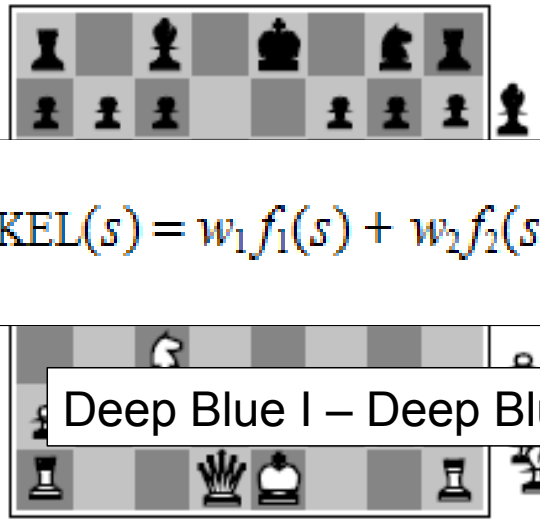
Kiértékelő függvények

Állapot → numerikus értékleképezés. Minél nagyobb a szám, annál értékesebb az állapot (pozíció).
Adott állapot minősítése – egy bizonyos mélységű keresési fa bejárása, ahol a levelek nem a játék végállapotai, hanem a kiértékelő függvénnyel minősített közbülső állapotok.

Játék elején (a célállapottól messze a kiértékelő függvény pontatlan, valamilyen mélységű kereséssel kell párosítani.
Játék végén (a célállapothoz közel) a kiértékelő függvény olyan pontos is lehet, hogy az állapot jellemzésére akár közvetlenül is alkalmazható.



Black to move
White slightly better



White to move
Black winning

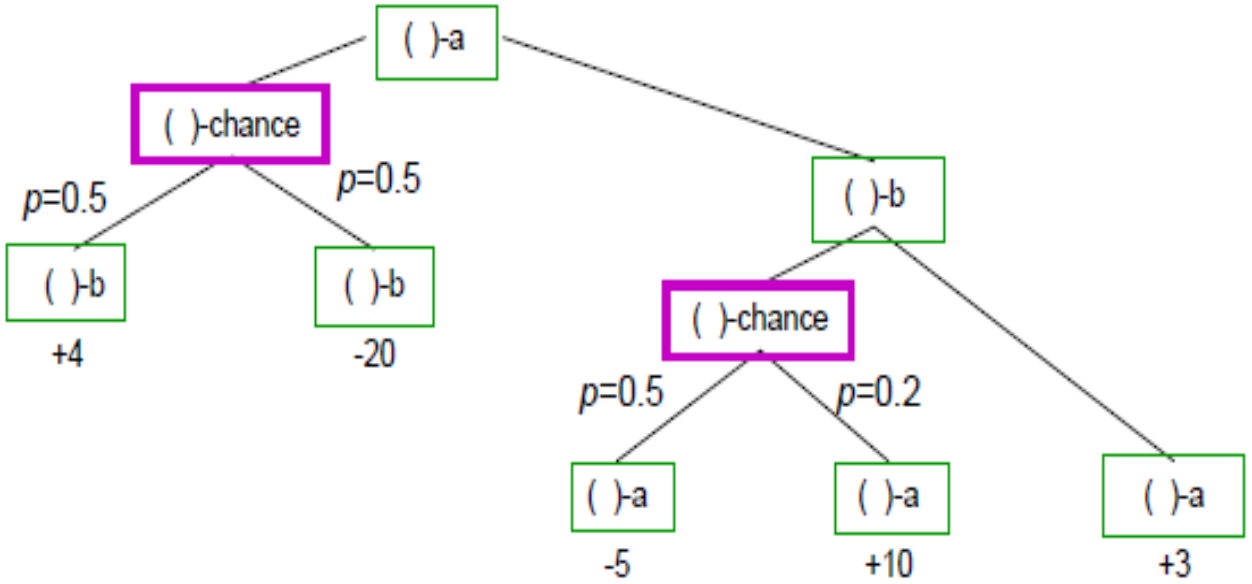
$$\text{KIÉRTÉKEL}(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s) = \sum_{i-1}^{n} w_i f_i(s)$$

Deep Blue I – Deep Blue II, 6400 – 8000 jellegfüggvény.

Játékfa véletlen elemet tartalmazó teljes információjú játékoknál

Játékosak döntésállapotai + a „természet” véletlen állapotai = esély-csomópontok (chance nodes), a problémára jellemző valószínűségekkel.

Állapotértékelés = végleges várható értékeke. ...



Egy ágenses problémák = speciális 2 ágenses játékproblémák
a másik ágens a környezet: … nem törődik semmivel (nem ellenséges)
… a nyeresége az ágens költsége (ellenséges)