5. Metody hraní her (minimax, alfa-beta, expectimax)

Metody hraní her

Cílem všech metod hraní her je **určit tah hráče na tahu**, který povede k jeho vítězství, nebo pro který je pravděpodobnost jeho vítězství nejvyšší.

Nejprve budeme uvažovat pouze hry, které hrají dva pravidelně se střídající hráči, které označíme symboly A a B. Každý z nich chce vyhrát a oba mají úplný přehled o aktuálním stavu hry.

Pro takové hry obecně platí:

- Hráč na tahu (označuje se vždy jako hráč A) zvítězí, vede-li k jeho vítězství alespoň jeden tah – problém OR.
- Hráč na tahu zvítězí, vedou-li po jeho tahu k jeho vítězství všechny možné tahy protihráče (hráče B)– problém AND.

Rozdělení her dvou protihráčů pro následující výklad:

1. Jednoduché hry – lze prozkoumat/vyšetřit všechny možné tahy (NIM).
2. Složité hry – zkoumá se pouze několik následujících tahů (Šachy).
3. Hry s neurčitostí (hod kostkou) – zkoumá se pouze několik následujících tahů s respektováním neurčitosti (Vrhcáby).

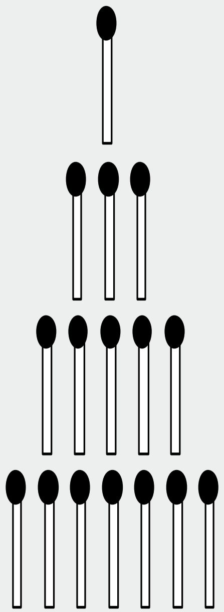
Jednoduché hry:

K řešení lze použít dříve uvedené AND/OR algoritmy, v nichž řešitelný uzel (SOLVED) znamená vítězství a neřešitelný uzel (UNSOLVED) prohru hráče, který je právě na tahu.

Hra NIM:

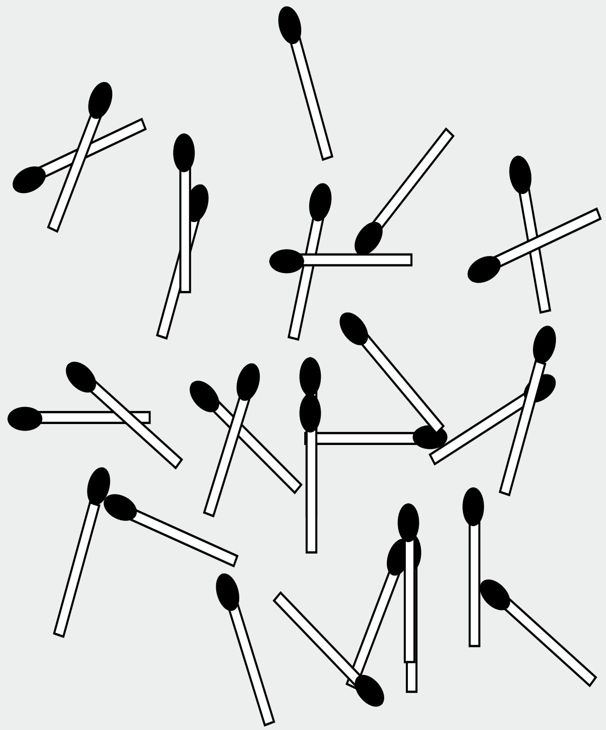
Hráči střídavě odebírají libovolný nenulový počet zápalek v jednom tahu z jediné řady. Prohrává ten hráč, který již nemá co odebrat (v jiné variantě hry prohrává hráč, který musí odebrat poslední zápalku).

Výchozí stav je např.:

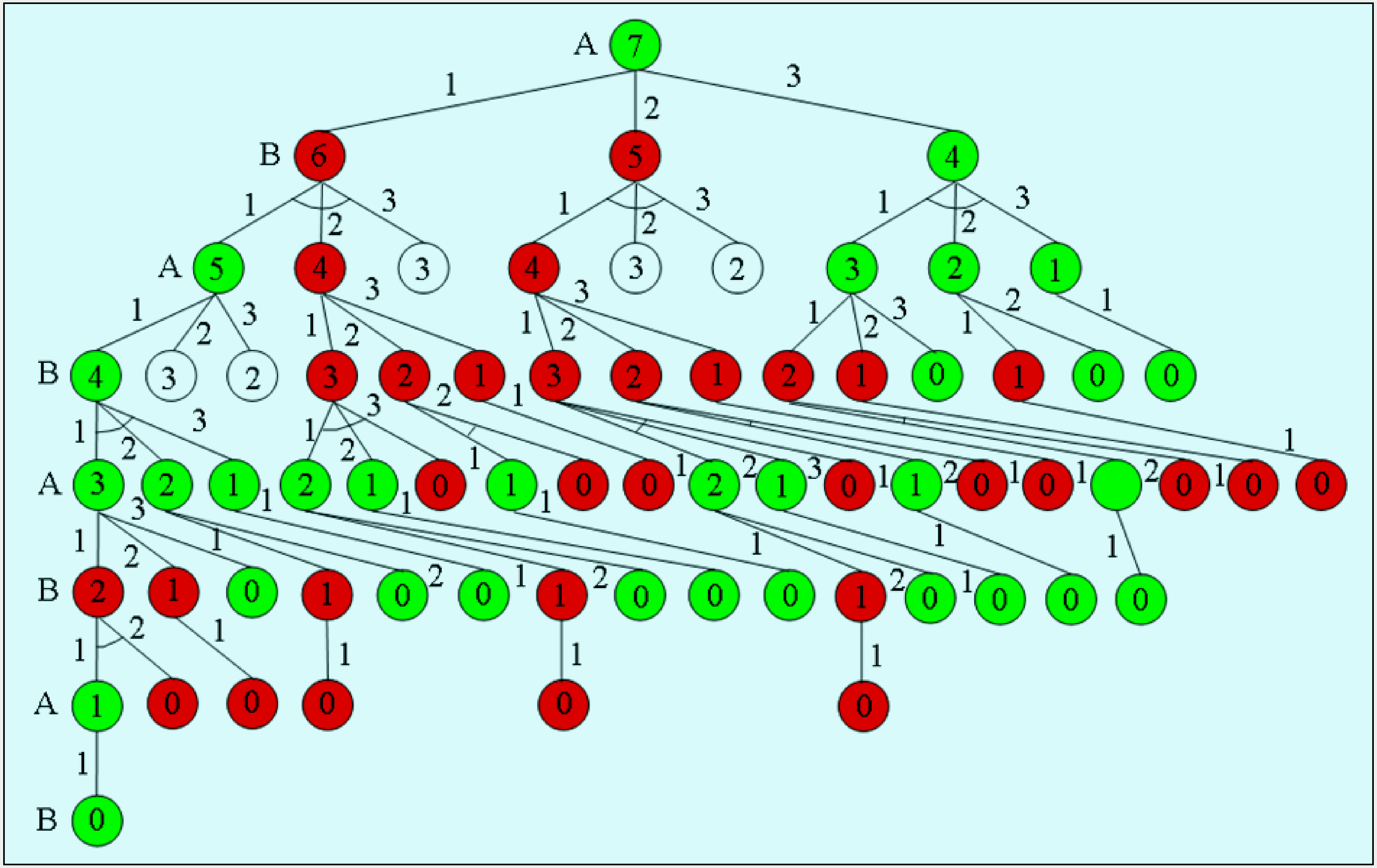


Hru NIM si pro další výklad zjednodušíme a tuto zjednodušenou hru nazveme Zápalky:

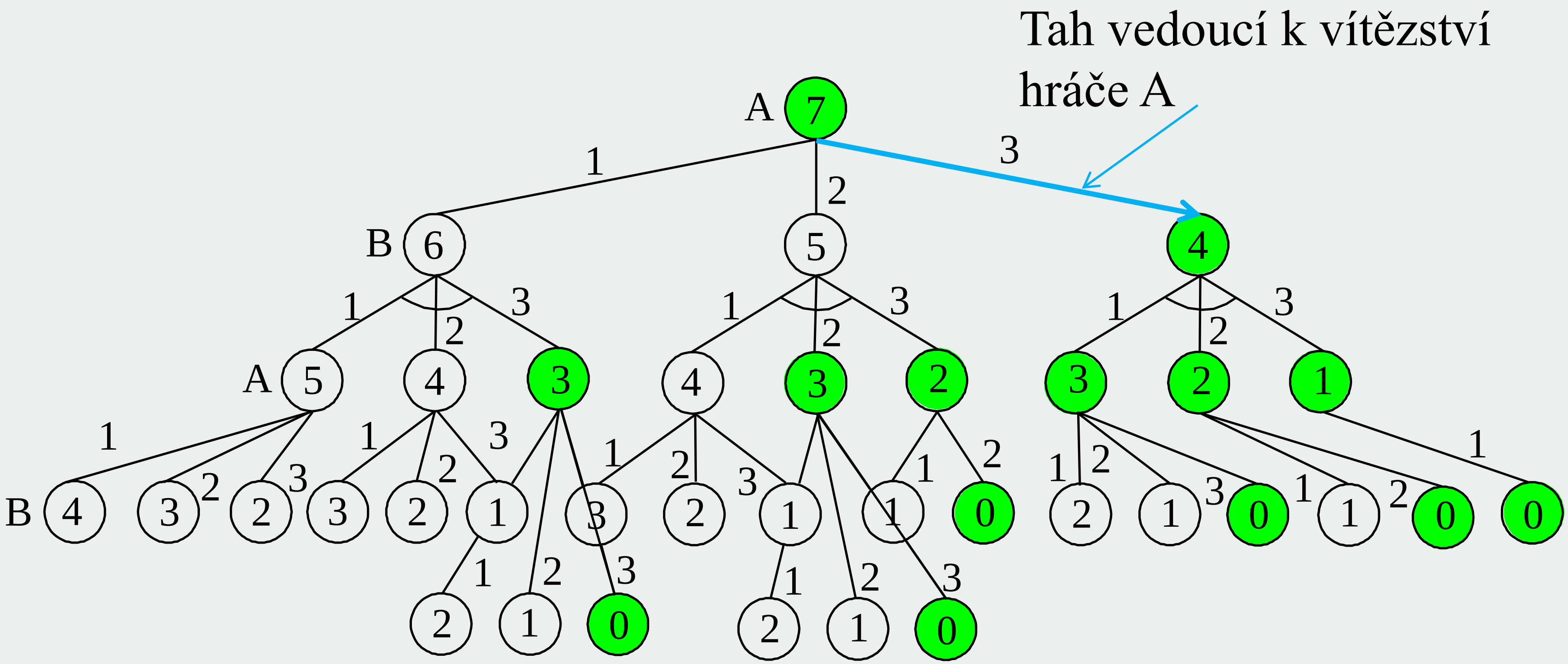
Hráči střídavě odebírají z hromádky zápalek buď jednu, nebo dvě, nebo tři zápalky. Prohrává ten hráč, který již nemá co odebrat. Výchozím stavem je libovolný počet zápalek.



Hra Zápalky – úplný AND/OR graf (stavy jsou dány aktuálními počty zápalek na hromádce, čísla u hran označují počty odebraných zápalek):



Hra Zápalky – řešení úlohy pomocí AO (AND-OR) algoritmu, BFS:



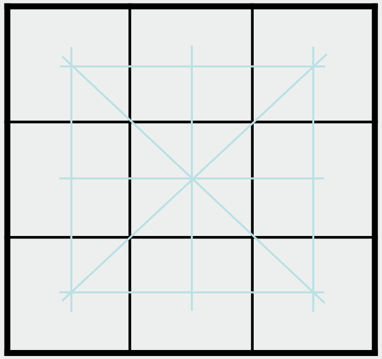
Složité hry – algoritmus/procedura MiniMax

- Každý stav hry je ohodnocen hodnotící funkcí.
- Kladné hodnoty této funkce musí znamenat příznivý stav pro hráče na tahu (hráče A), záporné pak příznivý stav pro protihráče (hráče B). Vítězství, resp. prohra jsou hodnoceny maximálními hodnotami (teoreticky $\pm\infty$, prakticky, například pro 8-mi bitová celá čísla hodnotami ± 127).
- Je zřejmé, že hráč A si vybírá tahy vedoucí do stavů s maximálním ohodnocením, protihráč si vybírá naopak tahy vedoucí do stavů s minimálním ohodnocením – odtud název MiniMax.
- Řešením problému <u>v daném stavu</u> hry je určení **nejvýhodnějšího tahu hráče A**.

Procedura MiniMax je rekurzivní a zahajuje se (poprvé se volá), když je na tahu hráč A. Vstupními parametry procedury jsou stav hry *X*, maximální hloubka prohledávání a informaci o hráči, který je právě na tahu (A nebo B). Procedura vrací ohodnocení aktuálního stavu a tah, který k tomuto ohodnocení vede:

- Je-li uzel *X* listem (konečným stavem hry, nebo uzlem v maximální hloubce) vrací ohodnocení tohoto uzlu.
- Je-li na tahu hráč A, tak postupně pro všechny jeho možné tahy (bezprostřední následníky uzlu *X*, tj. stavy před tahem hráče B) volá proceduru MiniMax pro hráče B, vrací maximální z navrácených hodnot a tah, který k nejlépe ohodnocenému bezprostřednímu následníkovi vede (tento tah má význam pouze u kořenového uzlu, kdy představuje nejvýhodnější reálný tah hráče A).
- Je-li na tahu hráč B, tak postupně pro všechny jeho možné tahy (bezprostřední následníky uzlu *X*, tj. stavy před tahem hráče A) volá proceduru MiniMax pro hráče A a vrací minimální z navrácených hodnot.

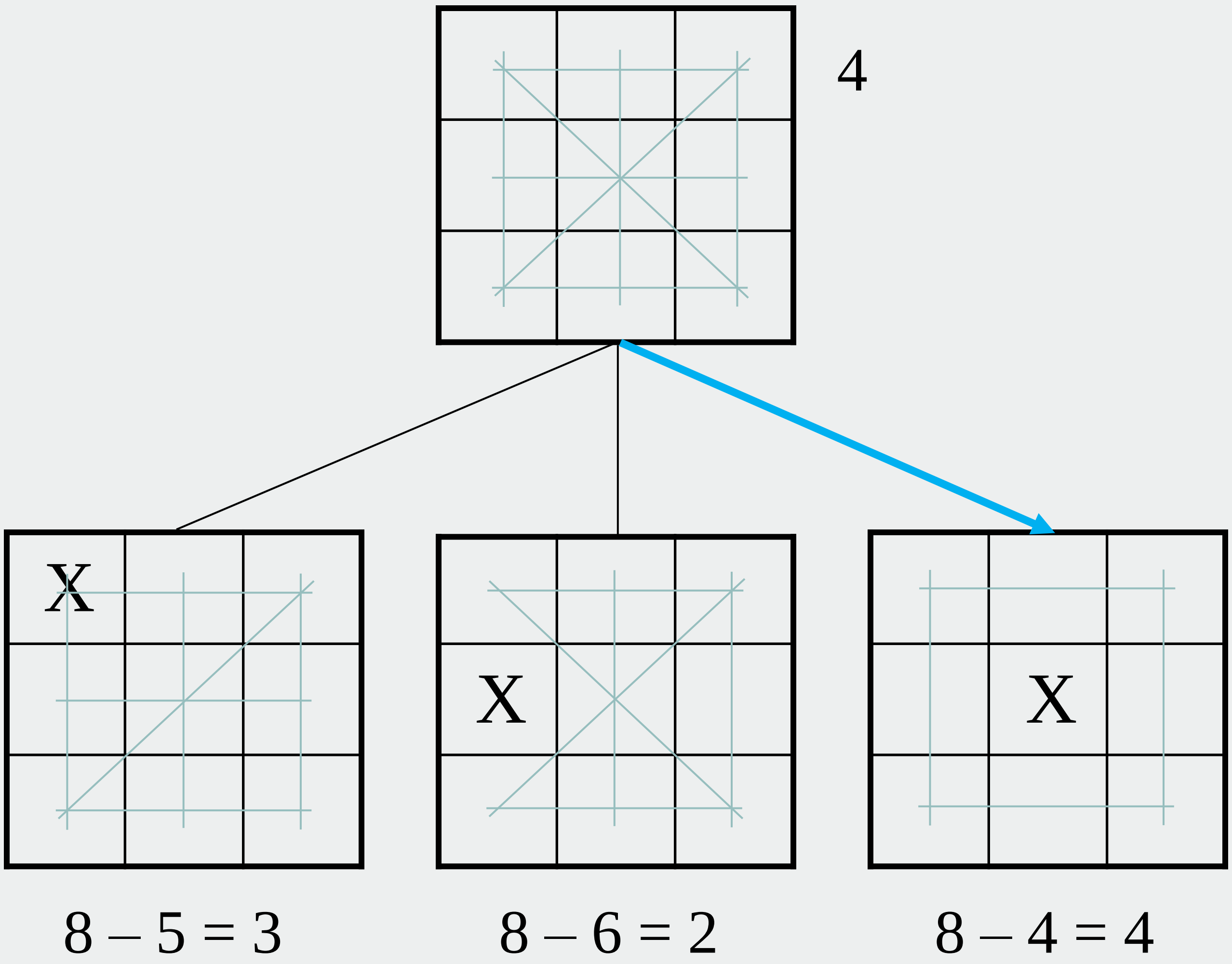
Příklad na MiniMax: Tic-Tac-Toe (křížky - kolečka).
Uvažujme velmi jednoduché základní pole o rozměrech 3 x 3, ve kterém je teoreticky možné vytvořit 8 „linek“:



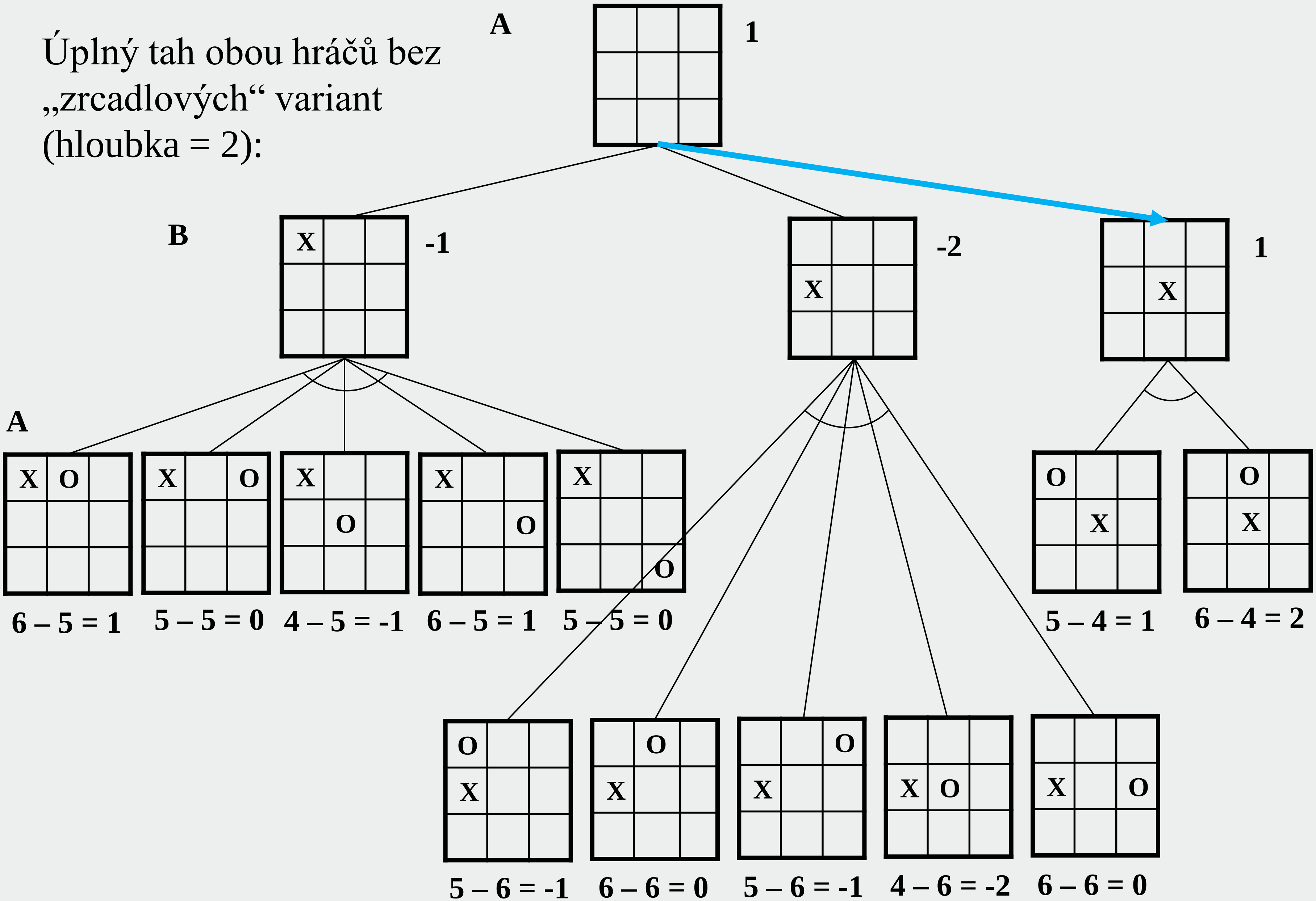
Nechť hráč A obsazuje pole křížky a hráč B kolečky, a nechť na tahu je hráč A.

Ohodnocení každého tahu nechť je dáno rozdílem dosažitelných linek hráčem A a dosažitelných linek hráčem B; ohodnocení počátečního stavu je tedy 8 – 8 = 0.

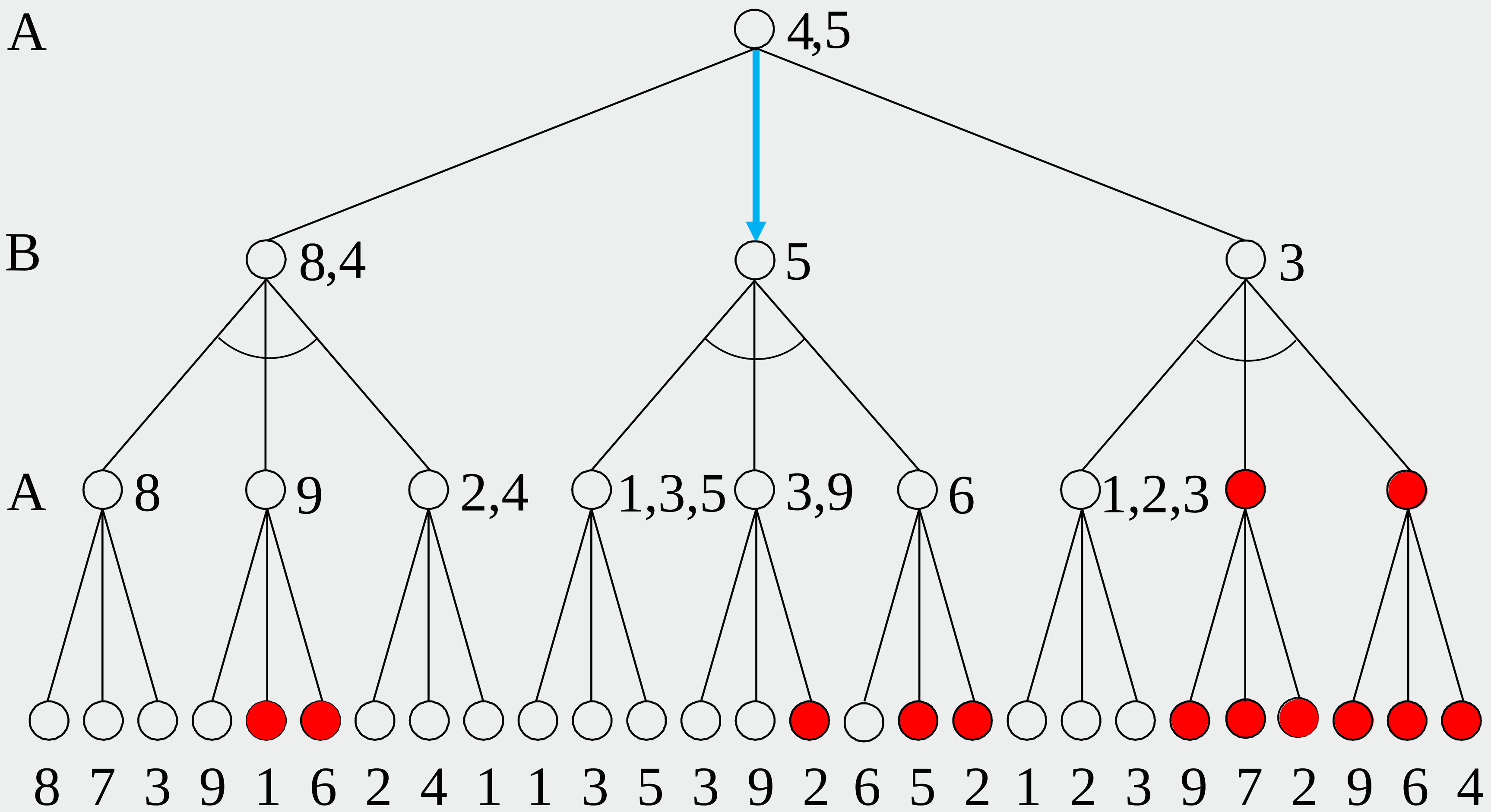
Příklad na MiniMax pro první tah hráče A, max hloubka = 1.



Úplný tah obou hráčů bez „zrcadlových“ variant (hloubka = 2):



Procedura Minimax však není příliš efektivní, protože řeší i podproblémy, které vůbec není nutné řešit:



Procedura AlfaBeta je velmi podobná proceduře MiniMax - je rekurzivní, poprvé se volá když je na tahu hráč A, vrací ohodnocení aktuálního stavu a tah, který k tomuto ohodnocení vede. Používá však navíc dva parametry, označené standardně symboly α a β, které se při jejím prvním volání nastavují na hodnoty $\alpha = -\infty$, $\beta = \infty$ (v praxi na minimum a maximum):

- Je-li uzel *X* listem (konečným stavem hry, nebo uzlem v maximální hloubce) procedura vrací ohodnocení tohoto uzlu.
- Je-li na tahu hráč A:
  a) Dokud platí nerovnost $\alpha < \beta$, tak postupně pro první/další tah (tj. pro prvního/dalšího bezprostředního následníka uzlu *X*) volá proceduru AlfaBeta s aktuálními hodnotami parametrů α a β. Po každém vyšetřeném tahu nastaví hodnotu parametru α na maximum z aktuální a navrácené hodnoty.

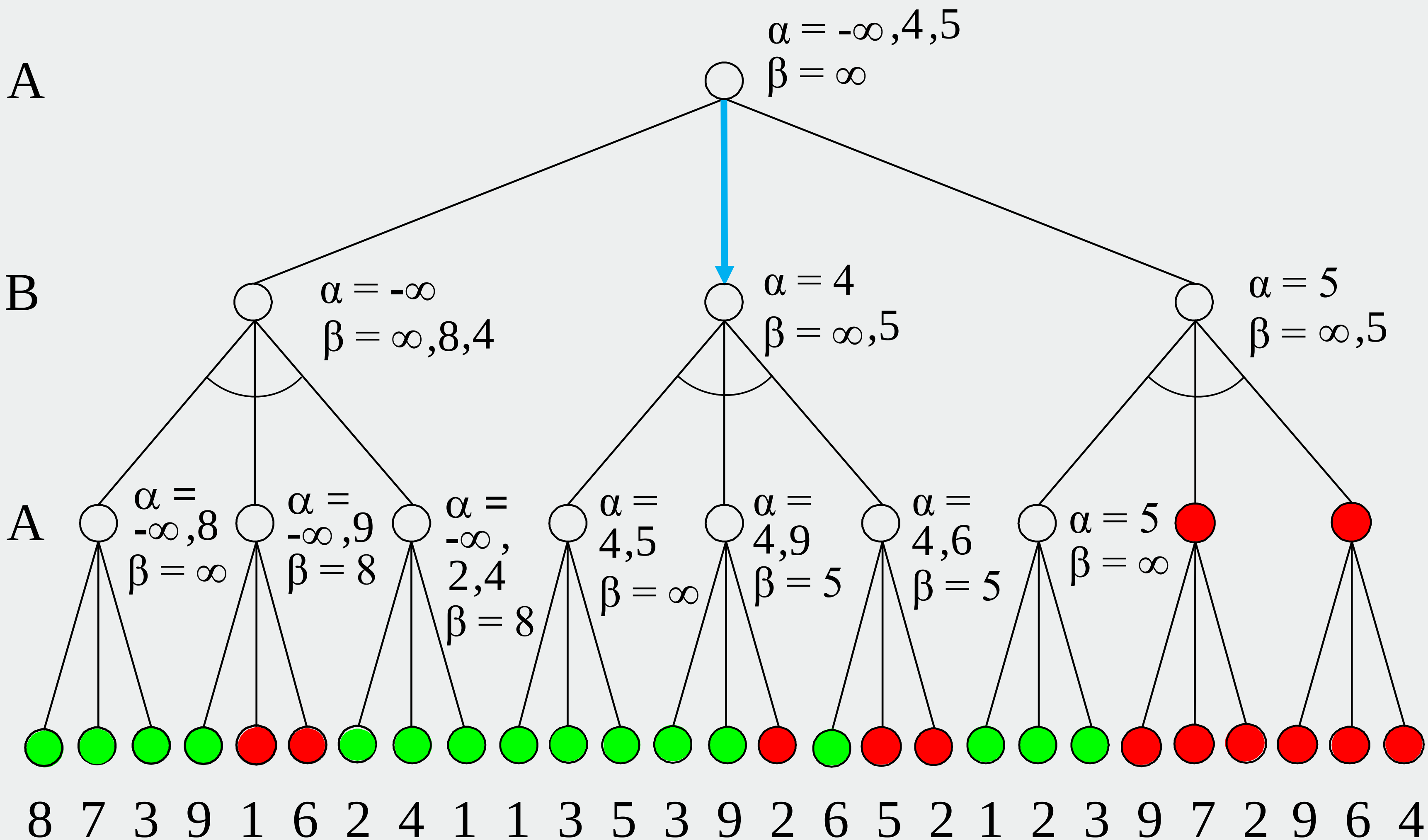
b) Je-li $\alpha \geq \beta$, nebo nemá-li uzel X žádného dalšího bezprostředního následníka, procedura vrací aktuální hodnotu parametru α a tah, který vede k nejlépe ohodnocenému bezprostřednímu následníkovi (tento tah má opět význam pouze u kořenového uzlu, kdy představuje nejvýhodnější reálný tah hráče A; pokud má ale stejné ohodnocení více možných tahů, je relevantní pouze tah, který byl takto ohodnocen jako první z těchto tahů !!!).

- Je-li na tahu hráč B:

a) Pokud platí nerovnost $\alpha < \beta$, tak postupně pro první/další tah (tj. pro prvního/dalšího bezprostředního následníka uzlu X) volá proceduru AlfaBeta s aktuálními hodnotami parametrů α a β. Po každém vyšetřeném tahu nastaví hodnotu parametru β na minimum z aktuální a navrácené hodnoty.

b) Je-li $\alpha \geq \beta$, nebo nemá-li uzel X žádného dalšího bezprostředního následníka, vrací aktuální hodnotu parametru β.

Ukázka činnosti AlfaBeta procedury



Hry s neurčitostí (vrh kostkou)

Hráči nyní nemohou vybírat minimální nebo maximální hodnoty, protože neznají předem výsledky hodů kostkou. Mohou však pracovat s hodnotami očekávanými (expectimin a expectimax), které se vypočítají pomocí pravděpodobností jednotlivých hodů takto:

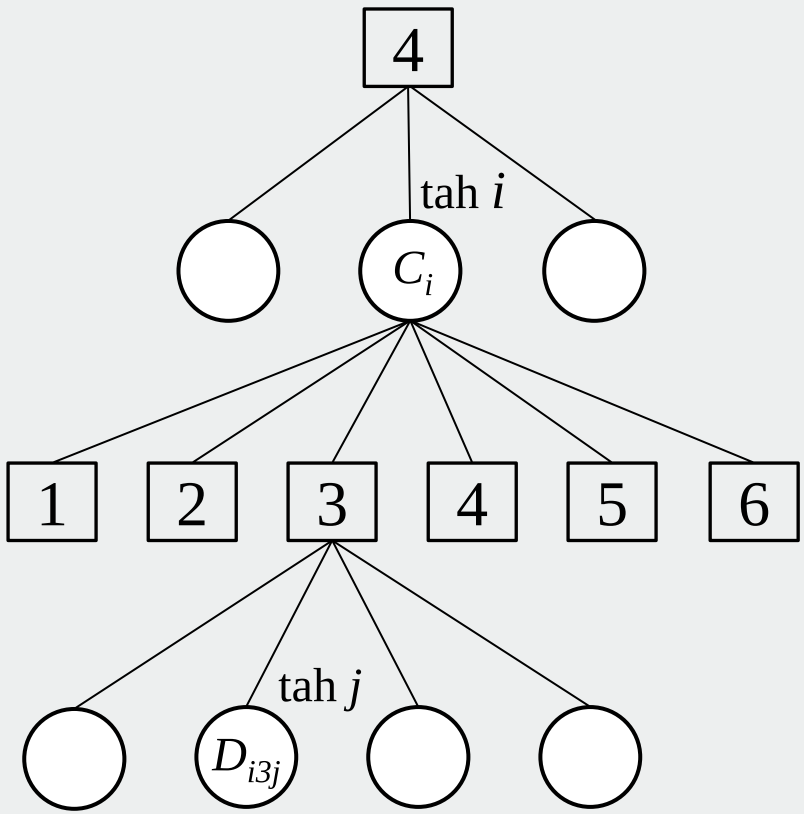
$$expectimin\left(C_i\right) = \sum_k P(h_k) * \min_j \left(D_{ikj}\right)$$

$$expectimax\left(D_j\right) = \sum_k P(h_k) * \max_i \left(C_{jki}\right)$$

V těchto vzorcích značí:

C_i	*i*-tý možný tah hráče A po jeho hodu kostkou
D_{ikj}	ohodnocení stavu po *i*-tém tahu hráče A, hodu h_k a *j*-tém tahu hráče B
D_j	*j*-tý možný tah hráče B po jeho hodu kostkou
C_{jki}	ohodnocení stavu po *j*-tém tahu hráče B, hodu h_k a *i*-tém tahu hráče A
$P(h_k)$	pravděpodobnost hodu h_k

$$expectimin(C_i) = \sum_k P(h_k) * \min_j (D_{ikj})$$



Hráč A hodil kostkou například h_4 (pak $P(h_4) = 1$ a $P(h_k) = 0$ pro k # 4)

Možné tahy hráče A (vybírá maximum)

Stav hry po možném tahu i hráče A

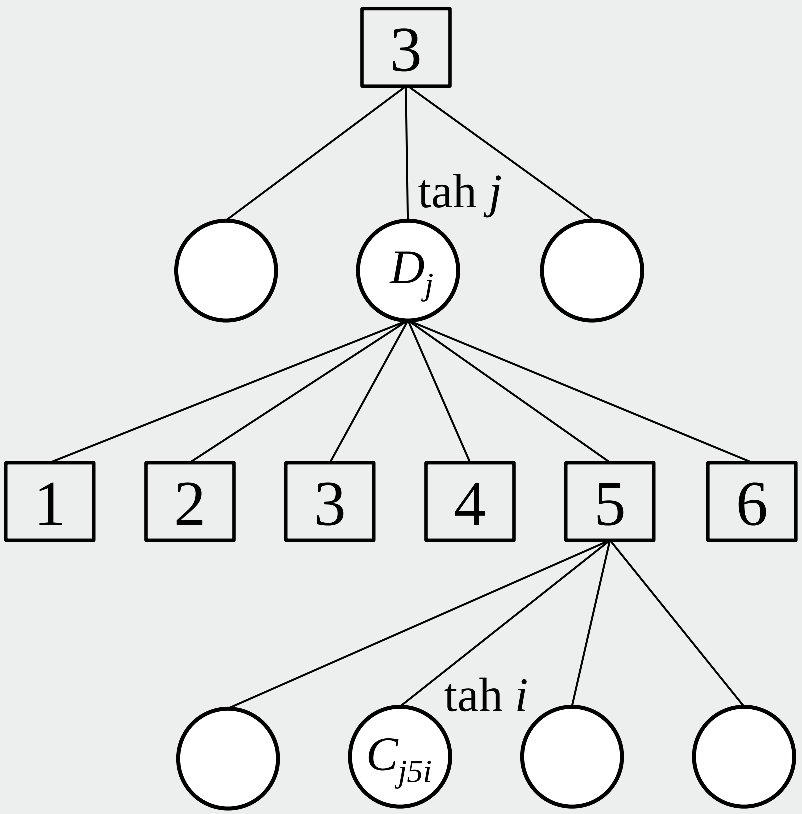
Hráč B hází kostkou

Rozhodování hráče B (např. po hodu h_3)

Možné tahy hráče B (vybírá minimum)

Stav hry po tahu j hráče B (tj. po tahu i hráče A, hodu h_3 hráče B a jeho tahu j)

$$expectimax\left(D_j\right) = \sum_k P(h_k) * \max_i \left(C_{jki}\right)$$



Hráč B hodil kostkou (například h_3)

Možné tahy hráče B (vybírá minimum)

Stav hry po možném tahu j hráče B

Hráč A hází kostkou

Rozhodování hráče A po hodu h_5

Možné tahy hráče A (vybírá maximum)

Stav hry po tahu i hráče A (tj. po tahu j hráče B, hodu h_5 hráče A a jeho tahu i)

Procedura ExpectiMiniMax je velmi podobná proceduře MiniMax:

- Je-li uzel *X* listem (konečným stavem hry, nebo uzlem v maximální hloubce) procedura vrací ohodnocení tohoto uzlu.
- Je-li na tahu hráč A, tak postupně pro všechny jeho možné tahy (bezprostřední následníky uzlu *X*) volá proceduru ExpectiMiniMax pro hráče B, vrací maximální hodnotu z navrácených hodnot *expectimin* a tah, který vede k nejlépe ohodnocenému bezprostřednímu následníkovi (tento tah má opět význam pouze u kořenového uzlu, kdy představuje nejvýhodnější reálný tah hráče A).
- Je-li na tahu hráč B, tak postupně pro všechny jeho možné tahy (bezprostřední následníky uzlu *X*) volá proceduru ExpectiMiniMax pro hráče A a vrací minimální hodnotu z navrácených hodnot *expectimax*.

Určování pravděpodobností hodů $P(h)$ kostkou s čísly 1 až 6:

- Při hodu jedinou kostkou existuje 6 stejných výsledků s $P(h_i) = 1/6$

$$P(h_1) = P(h_2) = P(h_3) = P(h_4) = P(h_5) = P(h_6)$$

- Při hodu dvěma kostkami existuje 21 různých výsledků:
  - stejné hodnoty na obou kostkách (6 možností, $P(h_{ii}) = 1/36$):

$$P(h_{11}) = P(h_{22}) = P(h_{33}) = P(h_{44}) = P(h_{55}) = P(h_{66})$$

  - různé hodnoty na obou kostkách (15 možností, $P(h_{ij}) = 1/18$, $i \# j$, výsledky xy a yx jsou stejné!):

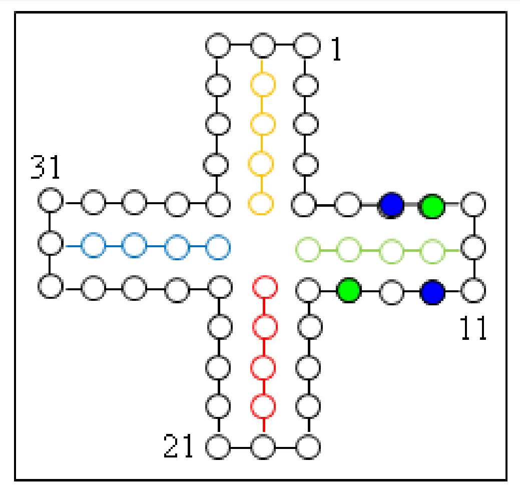
$$P(h_{12}) = P(h_{13}) = P(h_{14}) = P(h_{15}) = P(h_{16}) = P(h_{23}) = P(h_{24}) =$$

$$= P(h_{25}) = P(h_{26}) = P(h_{34}) = P(h_{35}) = P(h_{36}) = P(h_{45}) =$$

$$= P(h_{46}) = P(h_{56})$$

Člověče nezlob se – uvažujme hru s pouze dvěma hráči!

Nechť aktuální stav hry je stav ukázaný na obrázku a nechť na tahu je hráč se zelenými figurkami.
Oba hráči mají ve hře po dvou figurkách, které jsou rozmístěny kolem „domečku“ hráče se zelenými figurkami. Hráč se zelenými figurkami má své dvě figurky na pozicích 8 a 14 (tj. jednu figurku před „domečkem“ a jednu na začátku cesty). Protihráč má modré figurky na pozicích 7 a 12.



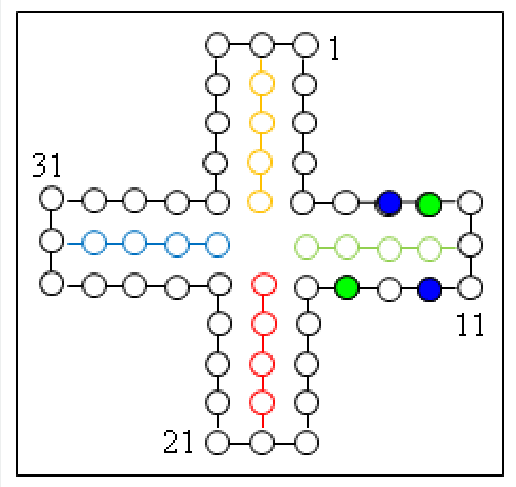
Listové stavy hry můžeme ohodnotit například takto:

hodnota = (dZ – dM) ∗ 15 + (pZ – pM) ∗ 5 + (ddZ – ddM) ∗ 2 + (pZM – pMZ)

kde jednotlivé symboly značí počty:

dZ/dM	zelených/modrých figurek v domečcích
pZ/pM	zelených/modrých figurek na hracím poli mimo domeček
ddZ/ddM	zelených/modrých figurek, které se mohou dostat po jednom hodu kostky do domečku
pZM/pMZ	bezprostředních ohrožení modrých/zelených figurek zelenými/modrými figurkami

Označíme-li stav hry seznamem [Z,M,h], kde Z značí seznam pozic zelených figurek ve hře, M seznam pozic modrých figurek ve hře a h ohodnocení stavu, pak výchozí stav uvedený na obrázku popisuje seznam [[8,14],[7,12],0]:

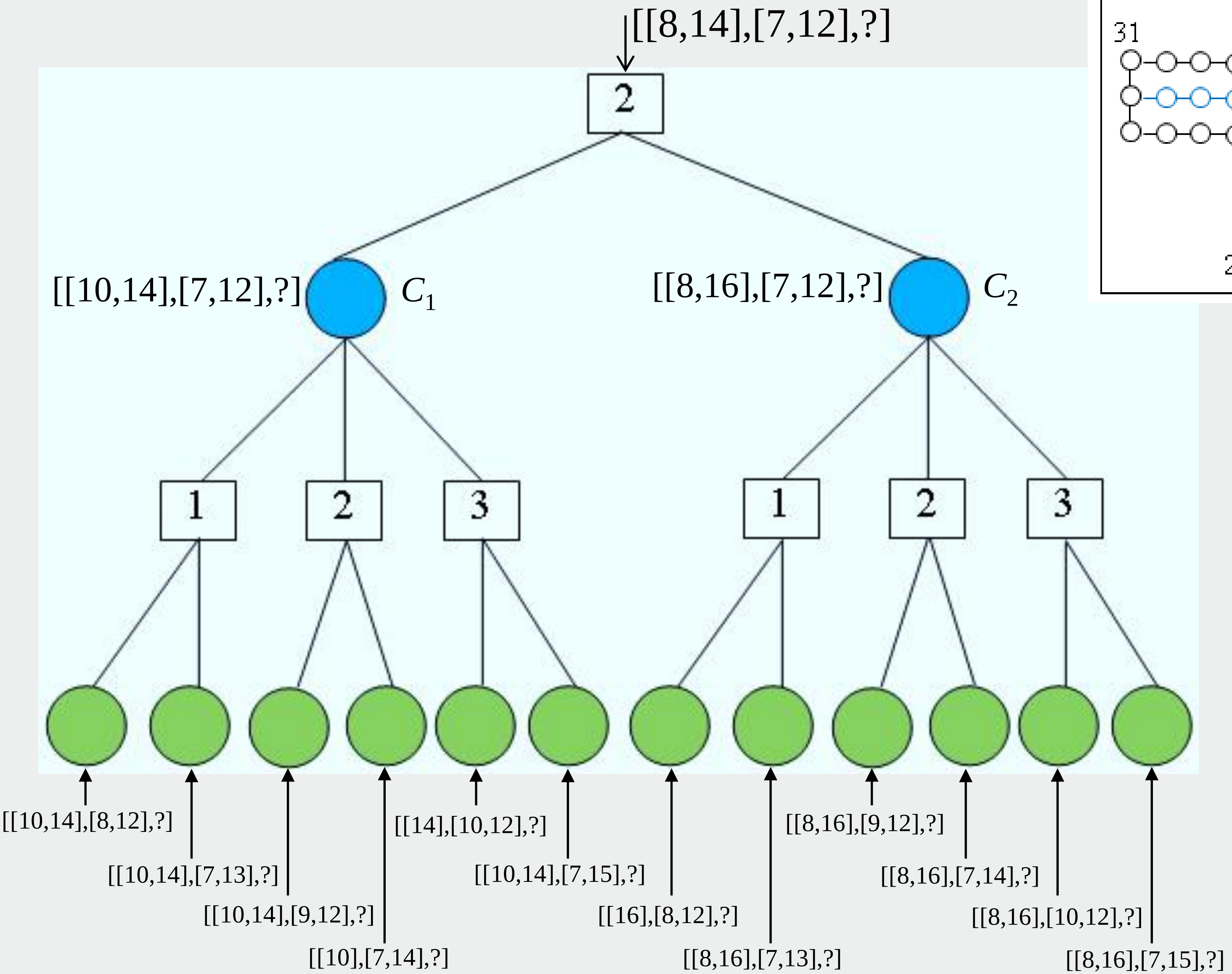


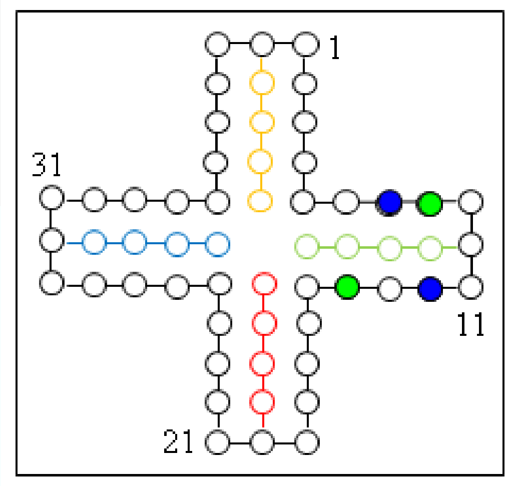
h = (dZ – dM) ∗ 15 + (pZ – pM) ∗ 5 + (ddZ – ddM) ∗ 2 + (pZM – pMZ) = (0 – 0) ∗ 15 + (2 – 2) ∗ 5 + (1 – 0) ∗ 2 + (0 – 2) = 0

Uvažujme šestistrannou hrací kostku s čísly 1,1,2,2,2,3, pro kterou jsou pravděpodobnosti hodů jednotlivých číslic $P(h_1) = 2/6$, $P(h_2) = 3/6$, $P(h_3) = 1/6$.

Nechť je na tahu hráč se zelenými figurkami a nechť právě hodil kostkou číslo 2. Nyní se musí rozhodnout mezi dvěma možnými tahy – první tah vede na stav [[10,14],[7,12],0] a druhý na stav [[8,16],[7,12],1]. Hráč na tahu si vybírá tah do lépe ohodnoceného stavu, a proto (pokud by neuvažoval další průběh hry) by si vybral tah do stavu [[8,16],[7,12],1].

Pokud bude hráč na tahu uvažovat následující tah hráče s modrými figurkami, pak bude situace tato:



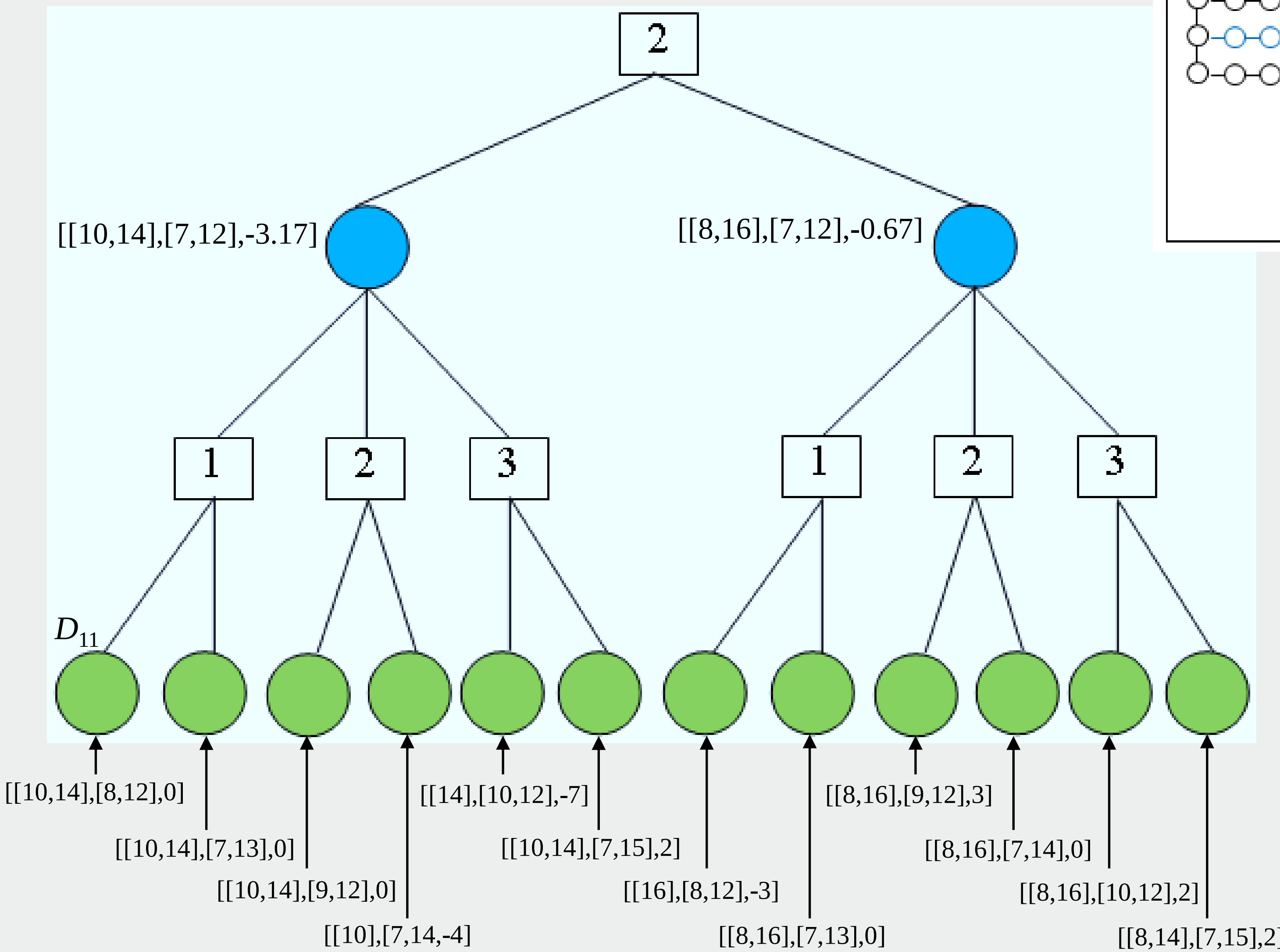


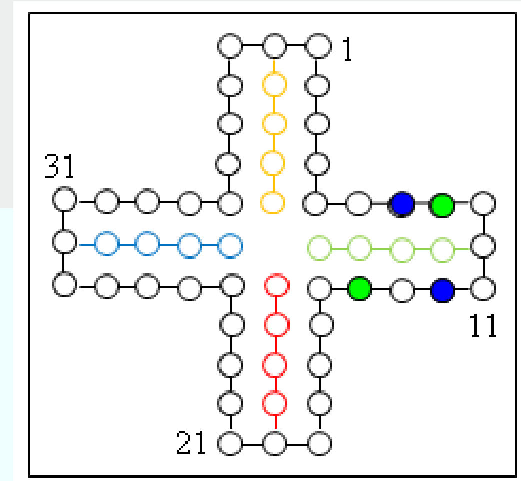
Výpočet ohodnocení stavů C_1 ([10,14],[7,12]) a C_2 ([8,16],[7,12]),

$$expectimin(C_i) = \sum_k P(h_k) * \min_j \left(D_{ikj}\right)$$

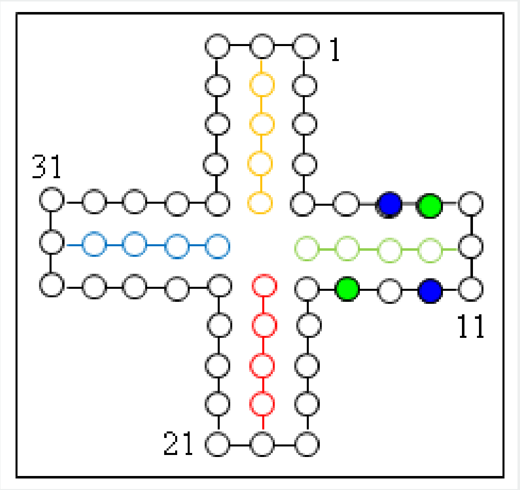
Stav	dZ	dM	pZ	pM	ddZ	ddM	pZM	pMZ	hodnota	min	$P(h)$	$P \cdot$ min	expectimin
[10,14],[8,12]	0	0	2	2	1	0	0	2	0				
[10,14],[7,13]	0	0	2	2	1	0	0	2	0	0	0,33	0	
[10,14],[9,12]	0	0	2	2	1	0	0	2	0				
[10],[7,14]	0	0	1	2	1	0	0	1	-4	-4	0,50	-2	
[14],[10,12]	0	0	1	2	0	0	0	2	-7				[10,14],[7,12]
[10,14],[7,15]	0	0	2	2	1	0	1	1	2	-7	0,17	-1,17	**-3,17**
[16],[8,12]	0	0	1	2	1	0	0	0	-3				
[8,16],[7,13]	0	0	2	2	1	0	0	2	0	-3	0,33	-1	
[8,16],[9,12]	0	0	2	2	1	0	1	0	3				
[8,16],[7,14]	0	0	2	2	1	0	0	2	0	0	0,50	0	
[8,16],[10,12]	0	0	2	2	1	0	1	1	2				[8,16],[7,12]
[8,16],[7,15]	0	0	2	2	1	0	1	1	2	2	0,17	0,33	**-0,67**

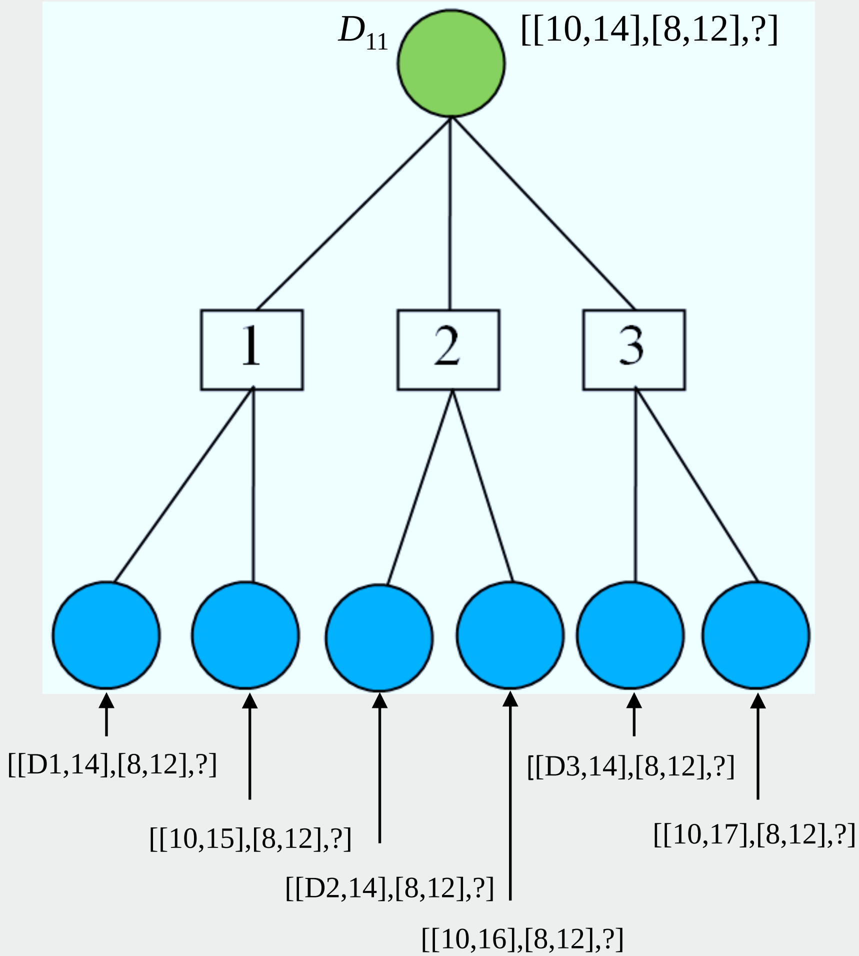
Hráč na tahu si v tomto jednoduchém příkladu opět vybere tah do stavu [[8,16],[7,12],-0.67], který má nyní jiné, ale opět lepší ohodnocení.





Při uvažování dalšího tahu se postupně ohodnocují dolní zelené stavy z předcházejícího snímku a pak se hodnoty předchozích (modrých) stavů přepočítají, atd.:





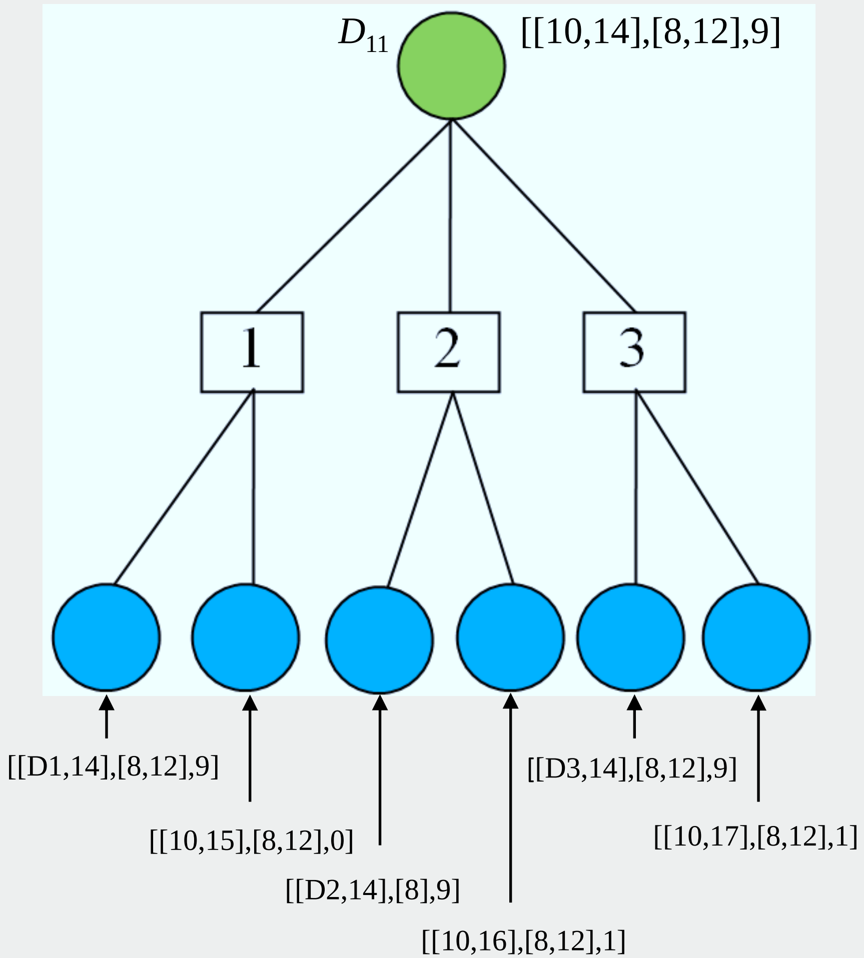
Symboly Di jsou označeny i-té pozice v domečku.

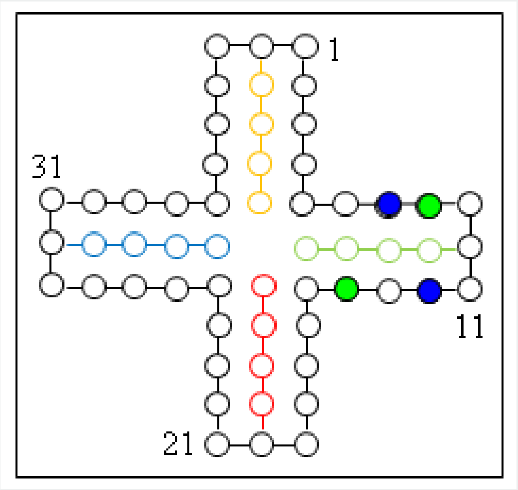
Výpočet ohodnocení stavu D_{11} ([10,14],[8,12]),

$$expectimax(D_j) = \sum_k P(h_k) * \max_i (C_{jki}) :$$

Stav	dZ	dM	pZ	pM	ddZ	ddM	pZM	pMZ	hodnota	max	$P(h)$	$P \cdot \max$	expectimax
[D1,14],[8,12]	1	0	1	2	0	0	0	1	9				
[10,15],[8,12]	0	0	2	2	1	0	0	2	0	9	0,33	3	
[D2,14],[8,12]	1	0	1	2	0	0	0	1	9				
[10,16],[8,12]	0	0	2	2	1	0	0	1	1	9	0,50	4,5	
[D3,14],[8,12]	1	0	1	2	0	0	0	1	9				[10,14],[8,12]
[10,17],[8,12]	0	0	2	2	1	0	0	1	1	9	0,17	1,5	9

Nové ohodnocení stavu D_{11} je tedy následující:



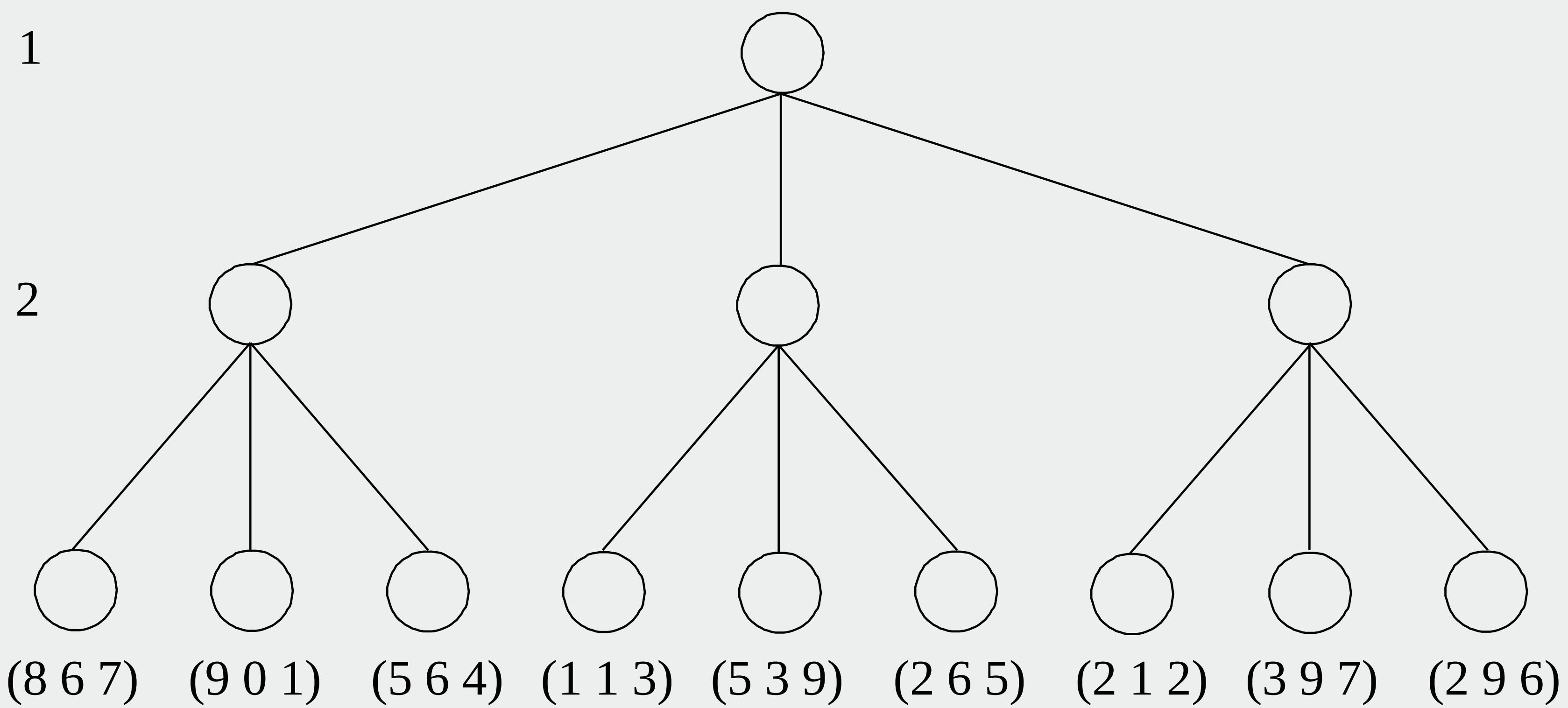


Nejznámější hrou dvou hráčů s kostkami jsou Vrhcáby/Backgammon. Jde o hru se dvěma kostkami jejíž pravidla naleznete například na https://cs.wikipedia.org/wiki/Vrhcáby

Consider now N players denoted by the indices i = 1...n, which regularly alternate (the second after the first, the third after the second to the last (nth) after the penultimate and then the first after the last, etc.). Again, assume that each player wants to win and that all players have a complete overview of the current state of the game.
We will evaluate each game state with a single evaluation function applied to each player separately, and the value of this function must be higher the more advantageous the position is for the evaluated player.
For each leaf state (node), the overall rating is then given by the list of values of the evaluation function for each player - for example, a rating of (8 3 5) means that the state has a rating of 8 for the first player, 3 for the second player and 5 for the third player.

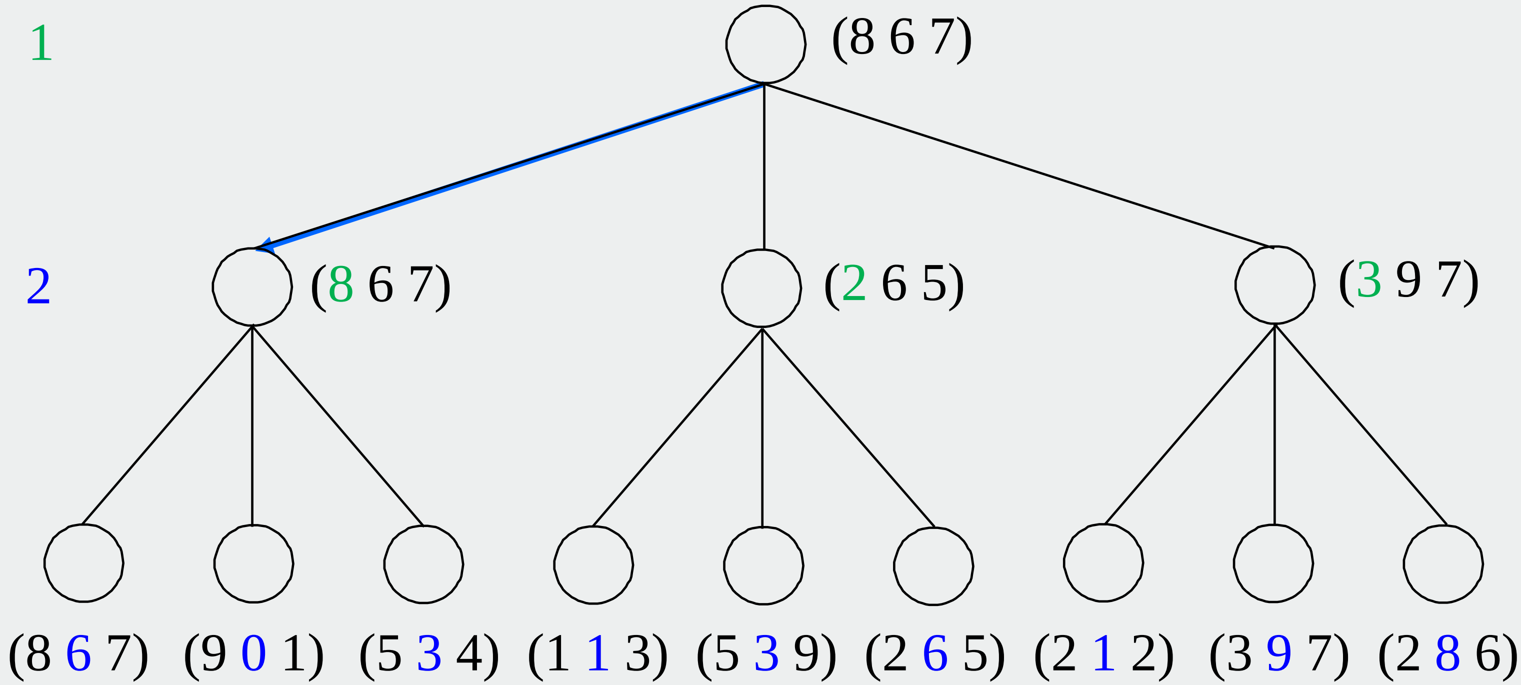
Clearly, each player chooses the state with the rating that is most advantageous to him.

A three-player example where Player 1 is considering a situation that may arise after Player 2's move:

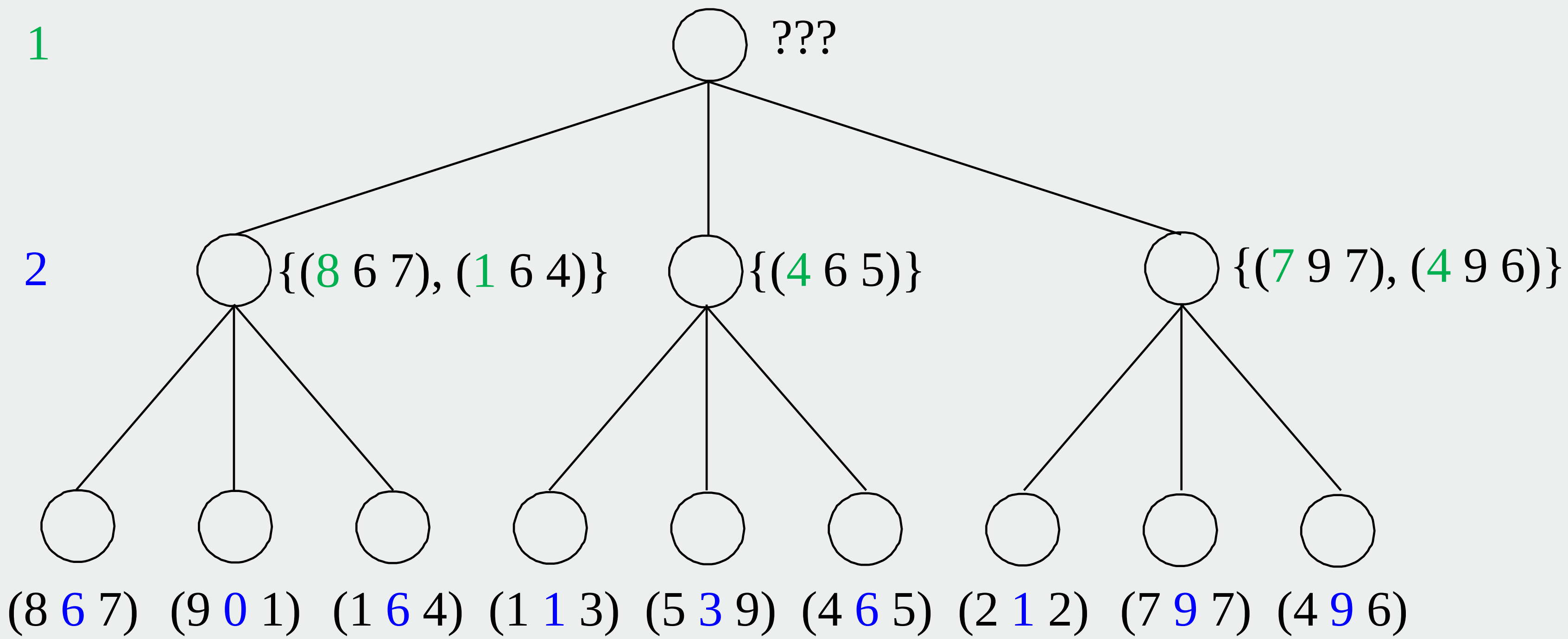


A generalization of the MiniMax procedure for n players is called Max^n:

- If node X is a leaf (the final state of the game, or a node at maximum depth), then the procedure returns the evaluation of this node $(v_1, v_2, \ldots, v_n)$.
- If player i is on the turn, then the procedure returns the valuation given by the valuation $(v_{j1}, v_{j2}, \ldots, v_{jn})$ of his immediate successor j for which $v_{ji} \geq v_{ki}$ holds, where v_{ki} are the state valuations for the player and all his other immediate successors:



The problem with the Max^n procedure is the ambiguity of the evaluation of a node (game state) when several immediate successors have the same maximum evaluation. This problem is (partially) solved by the Soft- Max^n procedure, which returns sets with the same valuations for the corresponding player, thus allowing the player on the turn to make more informed decisions:



If player 1 needs less than 5 points to win, then he will probably choose a move straight down, i.e. to the node (4 6 5). Otherwise, he can either take a risk (a move to the left), or bet on the sure thing, i.e. a move to the right.