# I007 Osnove umjetne inteligencije

## Tema: Igranje igara - pretraživanje u suparničkim okruženjima.

24. 3. 2021.


predavač: Darija Marković

asistent: Darija Marković

1 Igranje igara - pretraživanje u suparničkim okruženjima

# Igranje igara - pretraživanje u suparničkim okruženjima

- također problem pretraživanja prostora stanja, ali postoji protivnik
- u svakom stanju potrebno je donijeti optimalnu odluku o sljedećem potezu, tj. treba pronaći optimalnu strategiju
- fokusiramo se na deterministične igre s dva igrača, s potpunom informacijom i sumom nula

# Formalizacija problema

Problem pretraživanja sa sljedećim komponentama:

## Igra

1. **Početno stanje igre** $s_0 \in S$
2. **Funkcija sljedbenika** $\mathrm{succ} : S \to \wp(S)$ koja definira valjane poteze igre (prijelaze između stanja)
3. **Test na završno stanje** $\mathrm{terminal} : S \to \{\top, \bot\}$
4. **Funkcija korisnosti (dobitka)** $\mathrm{utility} : S \to \mathbb{R}$
   Npr. u šahu: $\mathrm{utility(s)} \in \{+1, 0, -1\}$

Početno stanje i funkcija sljedbenika definiraju stablo igre

# Primjer igre: križić – kružić

Imamo dva igrača = imena su

- MAX (prvi): stavlja križić (X)
- MIN (drugi): stavlja kružić (O)

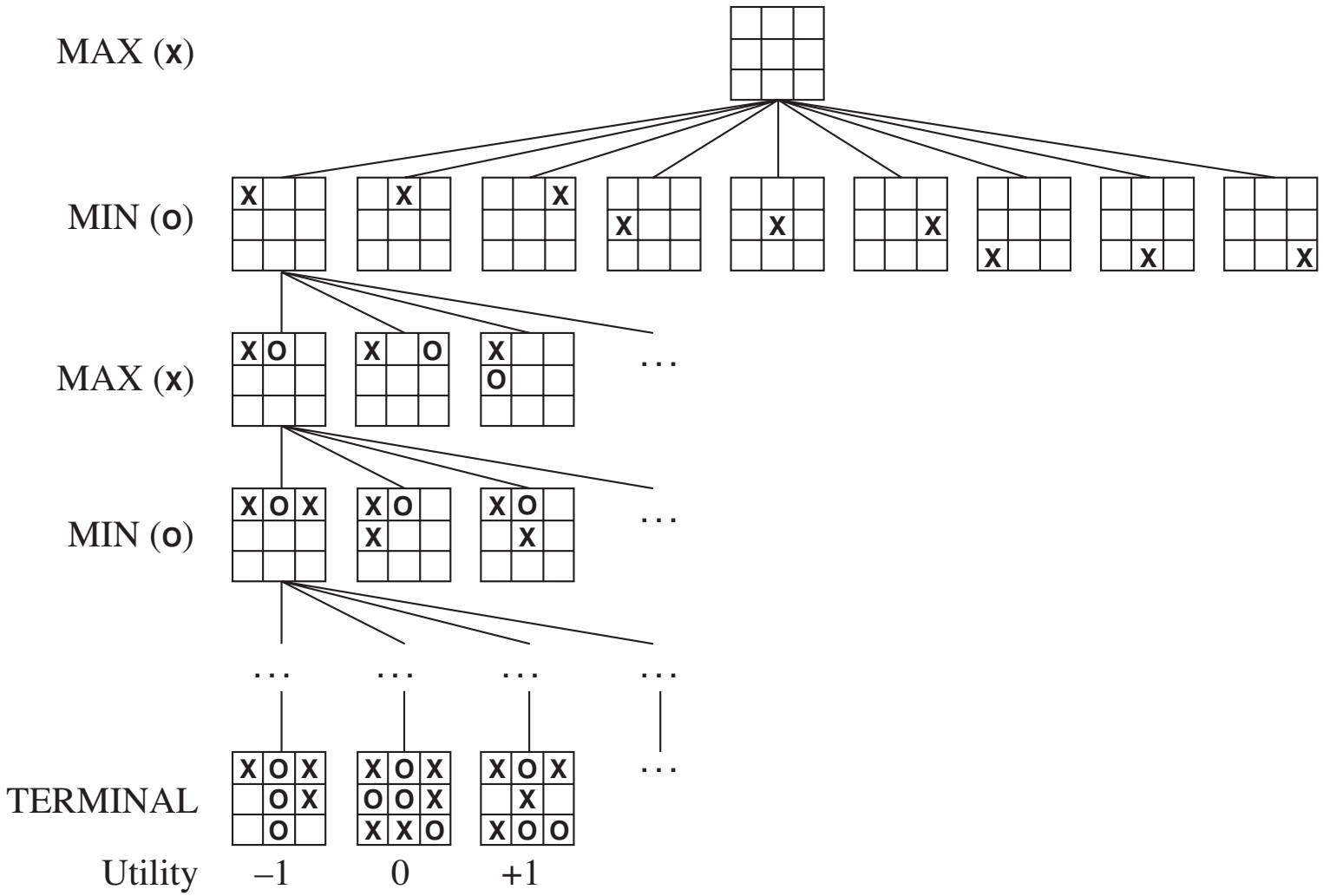
Igra se naizmjence: prvi, pa drugi, . . .

Cilj (pobjeda/poraz): 3 ista znaka u redu (vodoravno, okomito, koso)

Funkcija korisnosti (dobitka) za prvog igrača (engl. utility function):

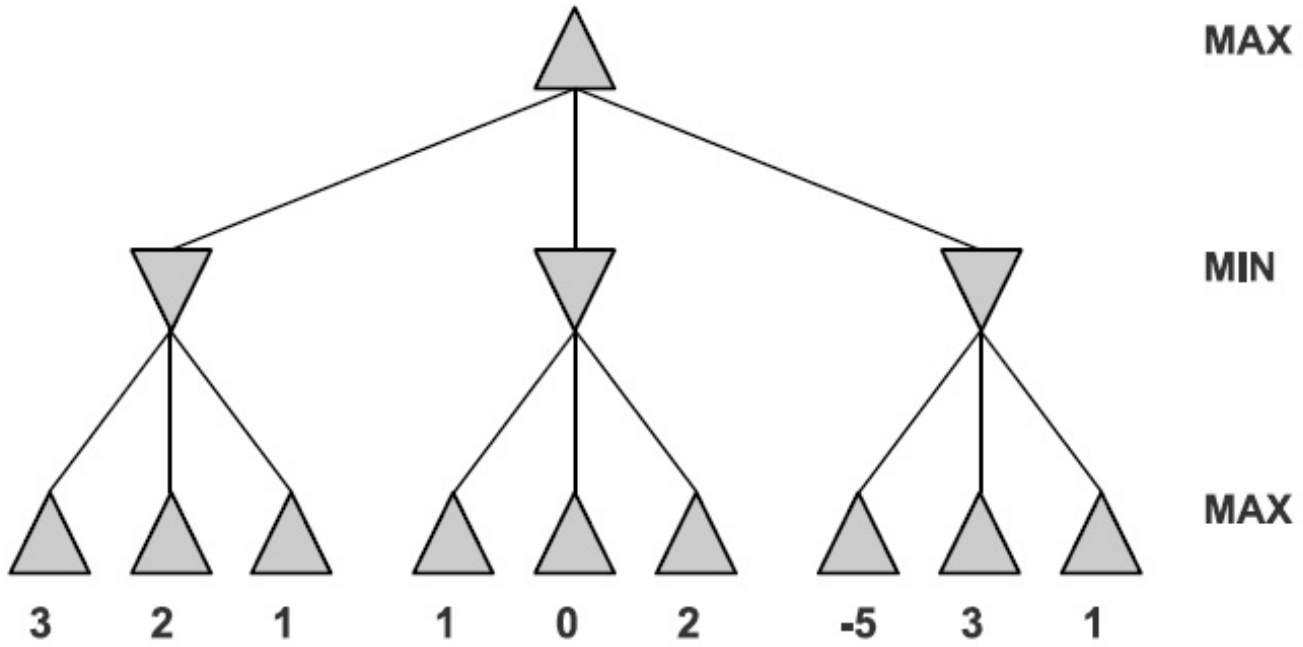
- pobjeda: dobitak = +1
- poraz: dobitak = -1
- neriješeno: dobitak = 0

# Stablo igre



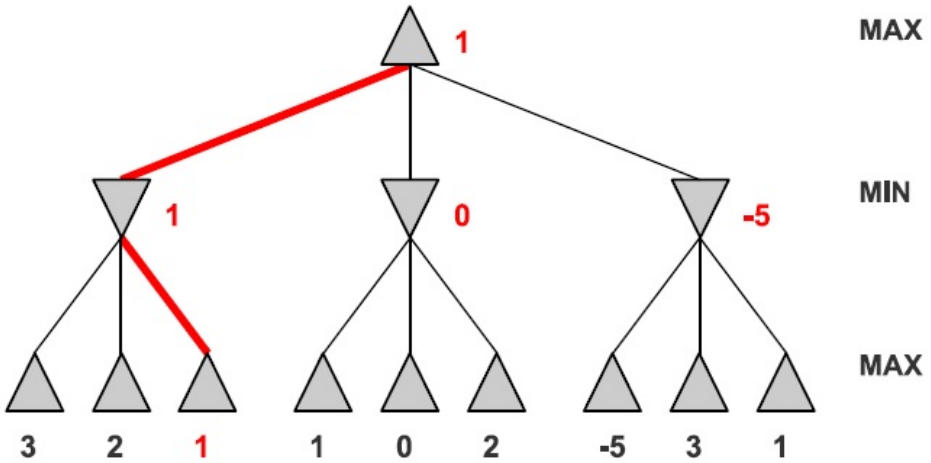
# Minimax strategija

- igrače ćemo nazvati MAX i MIN
- igrač MAX nastoji maksimizirati svoj dobitak, dok igrač MIN nastoji minimizirati dobitak igrača MAX
- igrači igraju naizmjenično: čvorovi na parnoj udaljenosti neka su MAX, a čvorovi na neparnoj udaljenosti neka su MIN



# Optimalna strategija

- Optimalna strategija igrača MAX je strategija koja mu donosi najveći dobitak, uz pretpostavku da igrač MIN koristi istu strategiju
- Svaki igrač koristi strategiju koja minimizira maksimalan gubitak




- Da bismo odredili optimalnu strategiju onog igrača koji je upravo na redu, trebamo izračunati minimax-vrijednost korijenskog čvora

# Minimax-vrijednost

- Minimax-vrijednost čvora s definirana je rekurzivno:

$$m(s) = \begin{cases} \text{utility}(s), & \text{ako } \text{terminal}(s) \\ \max_{t \in \text{succ}(s)} m(t), & \text{ako } s \text{ je MAX čvor} \\ \min_{t \in \text{succ}(s)} m(t), & \text{ako } s \text{ je MIN čvor} \end{cases}$$

- U prethodnom primjeru je:

$$m(s_0) = \max(\min(3,2,1), \min(1,0,2), \min(-5,3,1)) = 1$$

## Algoritam minimax

function $\max\text{Value}(s)$
  if $\text{terminal}(s)$ then return $\text{utility}(s)$
  $m \leftarrow -\infty$
  for $t \in \text{succ}(s)$ do
    $m \leftarrow \max(m, \text{minValue}(t))$
  return $m$

function $\min\text{Value}(s)$
  if $\text{terminal}(s)$ then return $\text{utility}(s)$
  $m \leftarrow \infty$
  for $t \in \text{succ}(s)$ do
    $m \leftarrow \min(m, \text{maxValue}(t))$
  return $m$

# Minimax strategija – napomene

- U praksi je protivnikova strategija nepoznata (i vjerojatno različita od strategije igrača MAX), pa zbog toga nije moguće savršeno predvidjeti protivnikove poteze (inače bi igra ionako bila dosadna)
- Zbog toga, kako bi napravili optimalan potez, svaki puta kada su na redu igrači moraju nanovo izračunati svoju optimalnu strategiju, krenuvši od trenutne pozicije igre u korijenu stabla
- Minimax pretražuje u dubinu, pa je njegova prostorna složenost $\mathcal{O}(m)$, gdje je $m$ dubina stabla pretraživanja
- Međutim, vremenska složnost je $\mathcal{O}(b^m)$, gdje je $b$ faktor grananja igre. To je vrlo nezgodno!

# Nesavršene odluke

- U stvarnosti nemamo vremena potpuno pretražiti stablo sve do završnih čvorova
- Moramo donositi vremenski ograničene i nesavršene odluke
- Pretraživanje treba presjeći na odredenoj dubini $d$ i napraviti procjenu vrijednosti funkcije dobitka uporabom heurističke funkcije
- Vrijednost $h(s)$ je procjena dobitka stanja $s$ za igrača MAX
  Npr. za šah: zbroj vrijednosti igračevih figura
- Heuristička funkcija često je definirana kao težinska linearna kombinacija više značajki:
  $h(s) = w_1 x_1(s) + w_2 x_2(s) + \cdots + w_n x_n(s)$

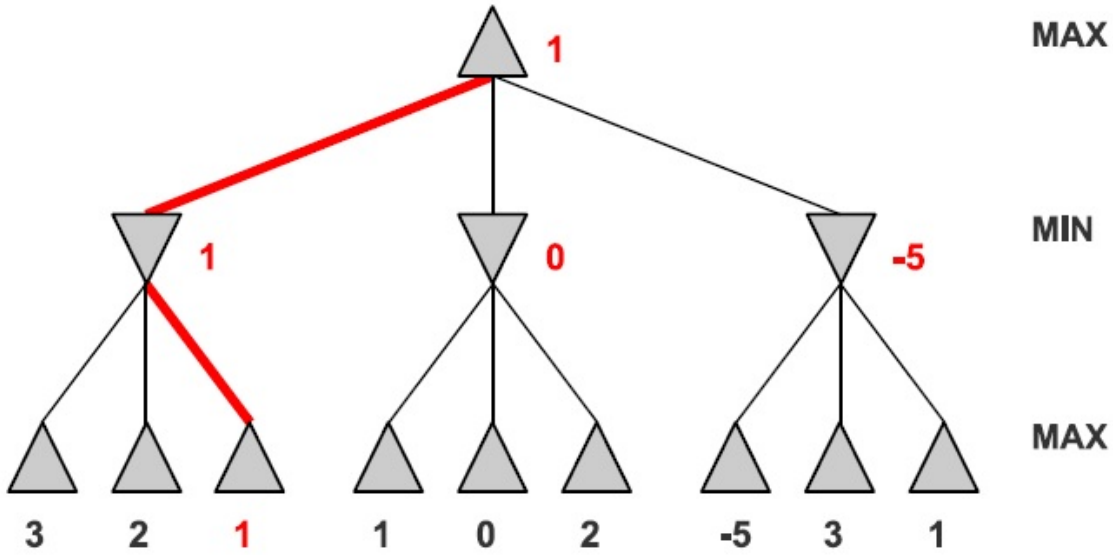
## Algoritam minimax s ograničenom dubinom

function $\max\text{Value}(s, d)$
  if $\text{terminal}(s)$ then return $\text{utility}(s)$
  if $d = 0$ then return $h(s)$
  $m \leftarrow -\infty$
  for $t \in \text{succ}(s)$ do
    $m \leftarrow \max(m, \text{minValue}(t, d-1))$
  return $m$

function $\min\text{Value}(s, d)$
  if $\text{terminal}(s)$ then return $\text{utility}(s)$
  if $d = 0$ then return $h(s)$
  $m \leftarrow \infty$
  for $t \in \text{succ}(s)$ do
    $m \leftarrow \min(m, \text{maxValue}(t, d-1))$
  return $m$

# alfa-beta podrezivanje

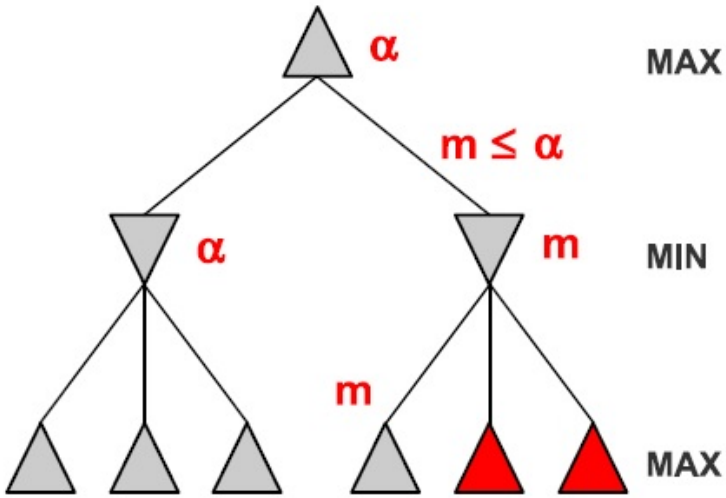
- Broj stanja igre raste eksponencijalno s brojem poteza
- Primjenom podrezivanja taj broj možemo međutim prepoloviti

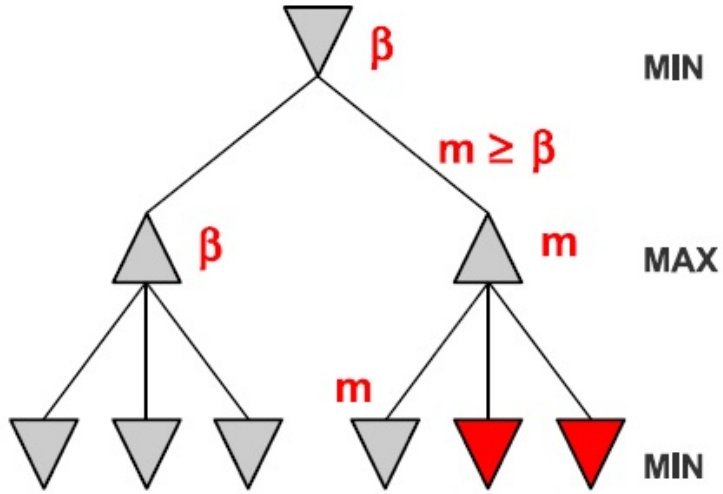



$$m(s_0) = \max(\min(3,2,1), \min(1,X,X), \min(-5,X,X)) = 1$$

# alfa-beta podrezivanje

- Podrezujemo kad god se za neki čvor ustanovi da potezi ni u kojem slučaju ne mogu biti povoljniji od nekog već istraženog poteza
- Ako su to potezi MAX: alfa-podrezivanje
- Ako su to potezi MIN: beta-podrezivanje




$\alpha$ – najveća nađena MAX-vrijednost

$\beta$ – najmanja nađena MIN-vrijednost

## Algoritam minimax s alfa-beta podrezivanje

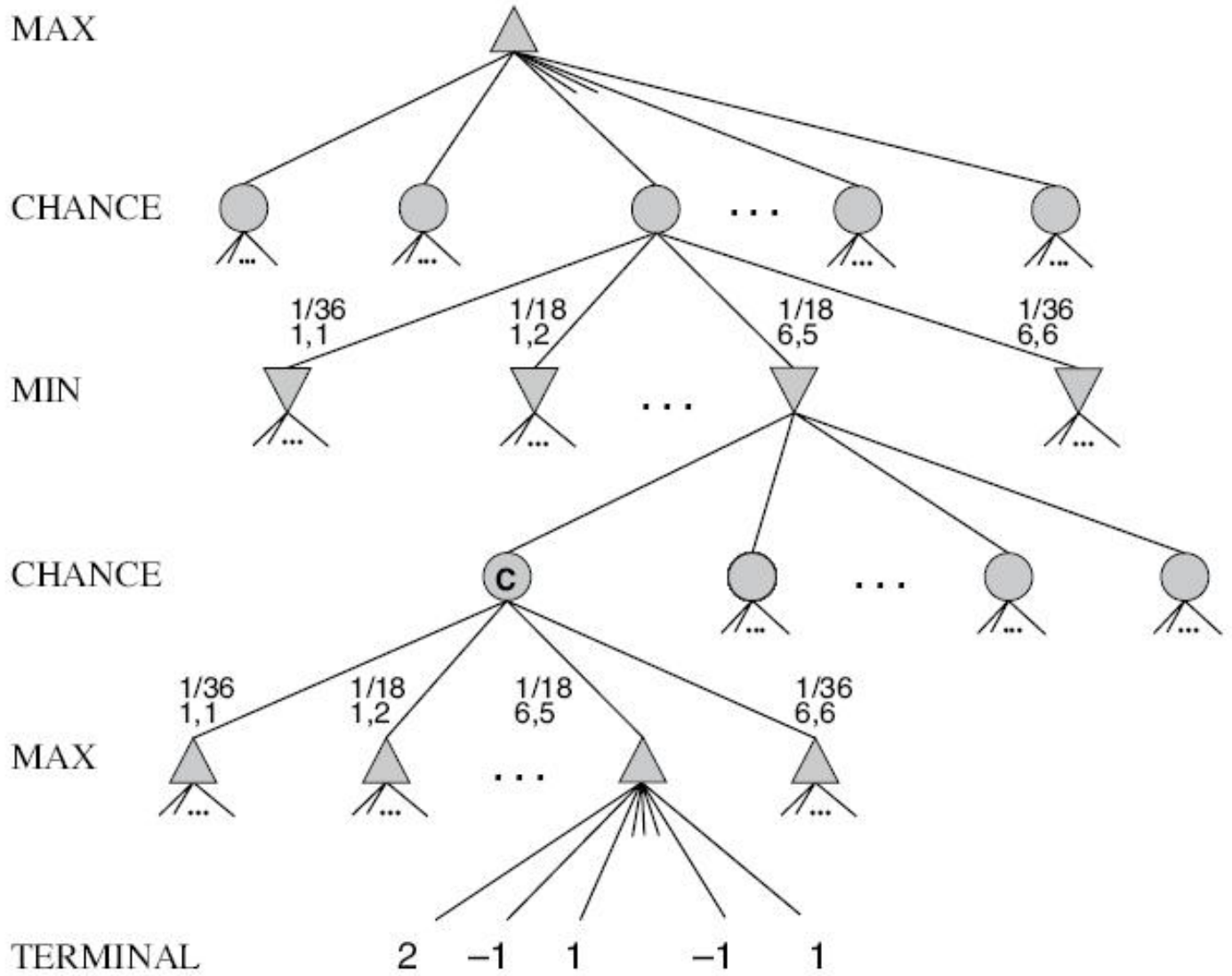
function $\max\text{Value}(s, \alpha, \beta)$
  if $\text{terminal}(s)$ then return $\text{utility}(s)$
  $m \leftarrow \alpha$
  for $t \in \text{succ}(s)$ do
    $m \leftarrow \max(m, \text{minValue}(t, m, \beta))$
    if $m \geq \beta$ then return $\beta$
  return $m$

function $\min\text{Value}(s, \alpha, \beta)$
  if $\text{terminal}(s)$ then return $\text{utility}(s)$
  $m \leftarrow \beta$
  for $t \in \text{succ}(s)$ do
    $m \leftarrow \min(m, \text{maxValue}(t, \alpha, m))$
    if $m \leq \alpha$ then return $\alpha$
  return $m$

# Nedeterminističke (stohastičke) igre

- Rezultati akcija nisu deterministički (bacanje kocke, izvlačenje karte)
- Osim MIN i MAX čvorova postoje i CHANCE čvorovi
- Za konstrukciju stabla igre:
  - slučajnost se tretira kao novi igrač = CHANCE
  - oznaka = kružni čvorovi, između MAX i MIN čvorova
- Grananje u CHANCE čvoru
  - predstavlja sve moguće ishode akcije,
  - svaka grana je označena ishodom i vjerojatnošću tog ishoda

# Primjer stabla igre za poziciju u backgammonu



# Izbor poteza

- Kako odlučujemo? Pozicije nemaju precizno definirane minimax vrijednosti! Umjesto toga, jedino razumno što možemo napraviti u CHANCE čvoru:
  - izračunati očekivanu vrijednost pozicije = srednja vrijednost po svim mogućim ishodima u tom čvoru.
- To radimo za svaki CHANCE čvor posebno.
- U svim ostalim čvorovima postupamo kao i ranije
  - u terminalnim čvorovima — vrijednost = dobitak
  - u MAX i MIN čvorovima (za koje se zna ishod bacanja tik ispred) radimo isto što i prije — optimiziramo očekivani dobitak (max/min)
- Ovo je generalizacija Minimax algoritma za detereminističke igre na nedeterminističke igre sa slučajnim čvorovima = Expectiminimax.

# Expectiminimax algoritam

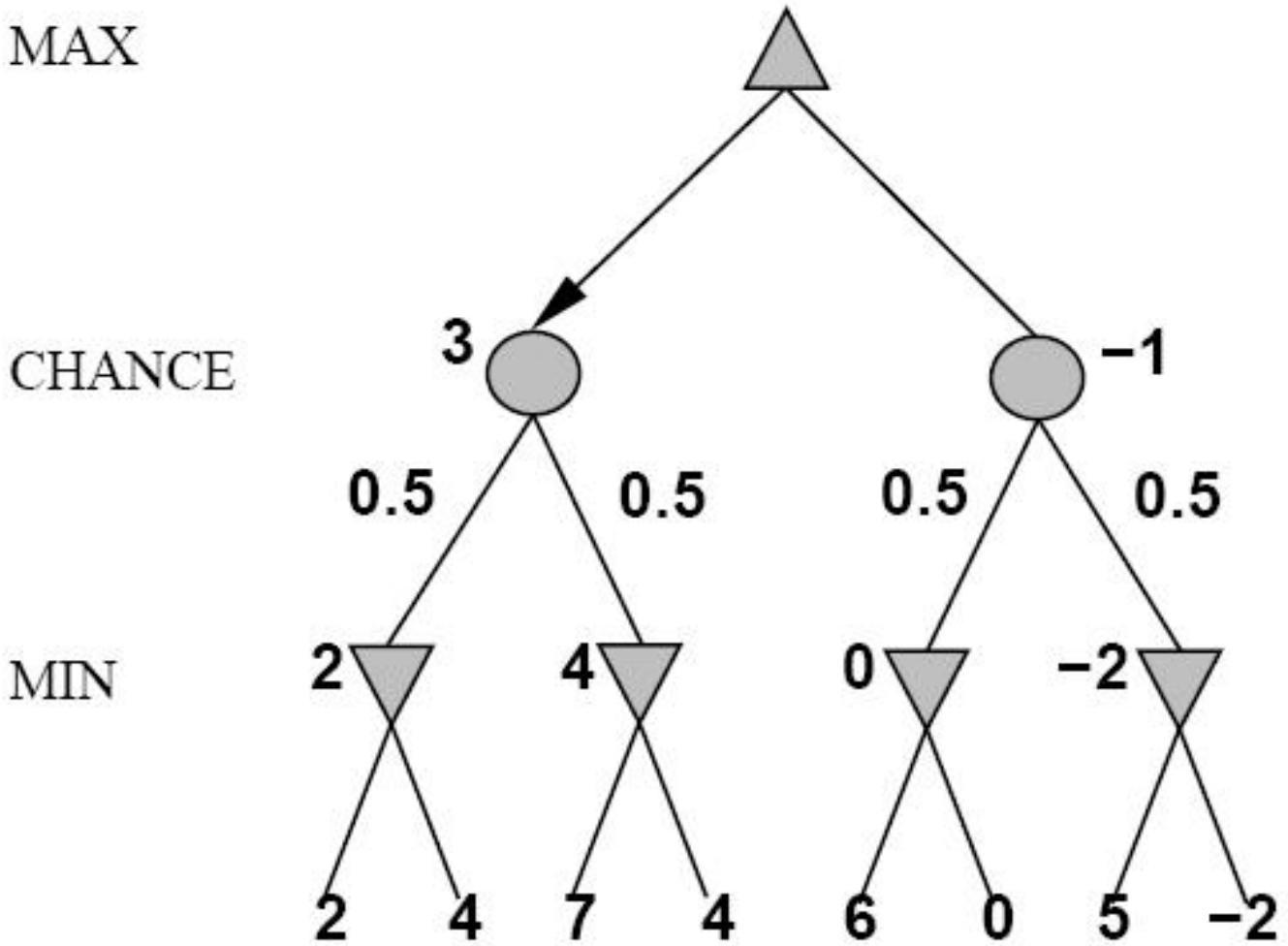
- Expectiminimax (očekivana minimax) vrijednost čvora $s$ definirana je rekurzivno:

$$Em(s) = \begin{cases} \text{utility}(s), & \text{ako } \text{terminal}(s) \\ \max_{t \in \text{succ}(s)} Em(t), & \text{ako } s \text{ je MAX čvor} \\ \min_{t \in \text{succ}(s)} Em(t), & \text{ako } s \text{ je MIN čvor} \\ \text{avr}_{t \in \text{succ}(s)} Em(t), & \text{ako } s \text{ je CHANCE čvor} \end{cases}$$

- Pri tome je $\text{avr}$ srednja vrijednost

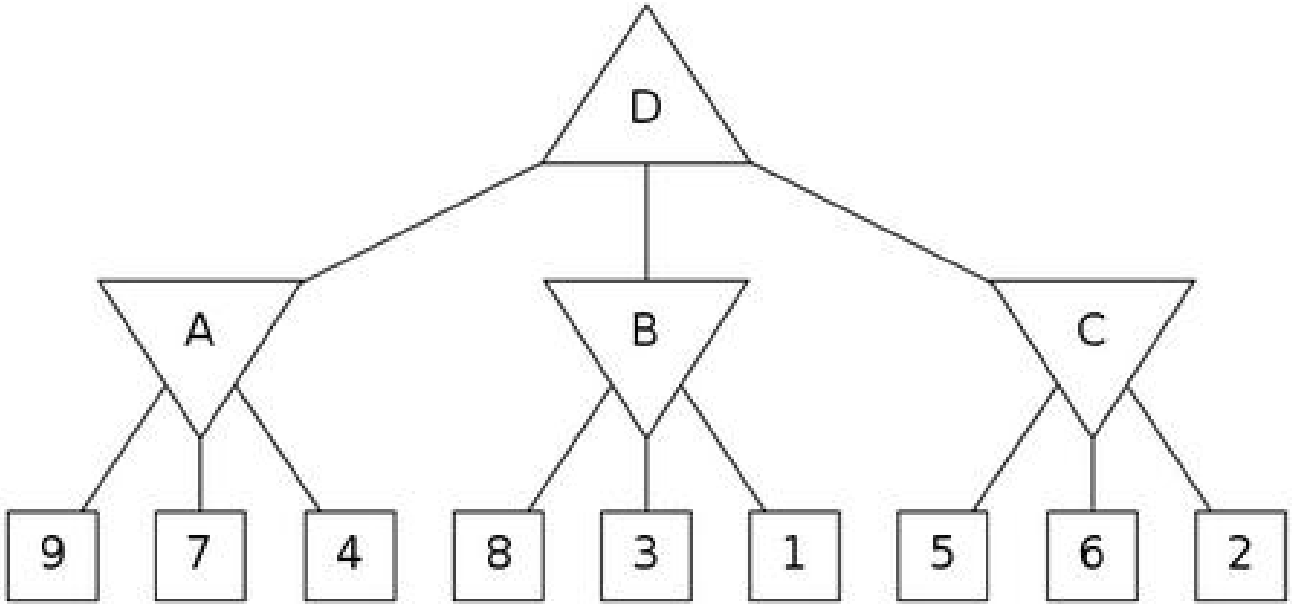
$$\sum_{t \in \text{succ}(s)} P(t) Em(t)$$

# Primjer bacanje novčića



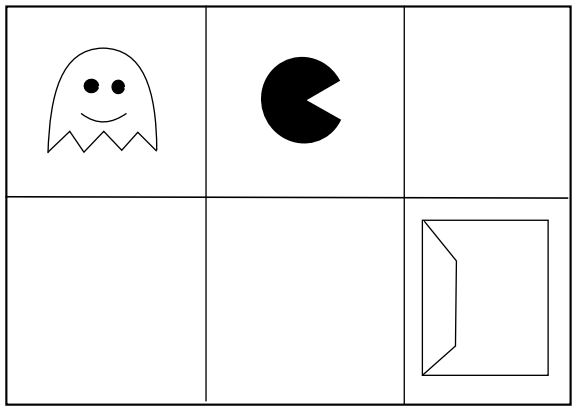
**Zadatak 1.**

Koristeći Minimax algoritam odredite vrijednost u čvorovima $A$, $B$, $C$ i $D$, uz pretpostavku da je prvi na potezu igrač MAX (označen s trokutima okrenutim prema gore), a nakon toga igrač MIN (označen s trokutima okrenutim prema dolje).


D
A
B
C
9
7
4
8
3
1
5
6
2

## Zadatak 2.

Pacman treba izaći iz labirinta u kojem je duh. U svakom koraku može se pomaknuti ili Pacman ili duh na jedan od susjednih kvadrata (unutar labirinta). Pacman je prvi na potezu i želi izaći iz labirinta što je prije moguće. Ukoliko se Pacman i duh nađu unutar istog kvadrata, igra završava i Pacmanova korist iznosi $-1$. Ako Pacman dođe na izlazni kvadrat (s vratima) igra završava i njegova korist iznosi $1$. Duh se ponaša suparnički i nastoji minimizirati Pacmanovu korist.



(a) Odredite minimax-vrijednost stanja na slici.

(b) Ukoliko provodite minimax algoritam s ograničenom dubinom iz prikazanog stanja, za dubine $d = 1, 2, 3$ koristeći heurističku funkciju koja je $0$ na svim ne-terminalnim stanjima, odredite minimax-vrijednost stanja na slici.

(c) Pretpostavite da je labirint dimenzije $M \times N$ i sadrži $K$ duhova te da provodite minimax algoritam s ograničenom dubinom $d$. Koliko mnogo stanja će se proširiti (asimptotski) u ovisnosti o $d$ koristeći standardni minimax algoritam?