Системи, основани на знания – 2020/2021 учебна година

Тема 5:
Моделиране на игри

Класическа постановка на задачата:

Разглеждат се т. нар. интелектуални *игри с пълна информация*, които се играят от двама играчи и върху хода на които не оказват влияние случайни фактори. Двамата играчи играят последователно и всеки от тях има пълна информация за хода на играта.

Най-често се решава задачата за намиране на най-добър първи ход на играча, който трябва да направи текущия ход.

ДС при тези задачи е дърво на възможните позиции в резултат на възможните ходове на двамата играчи. Общият брой възли в ДС е от порядъка на b^d (b - коефициент на разклонение на игровото дърво, d – височина (дълбочина) на игровото дърво).

Пример. В шахмата е установено, че b има средна стойност около 35, а всеки играч играе около 50 хода, т.е. d има средна стойност около 100.

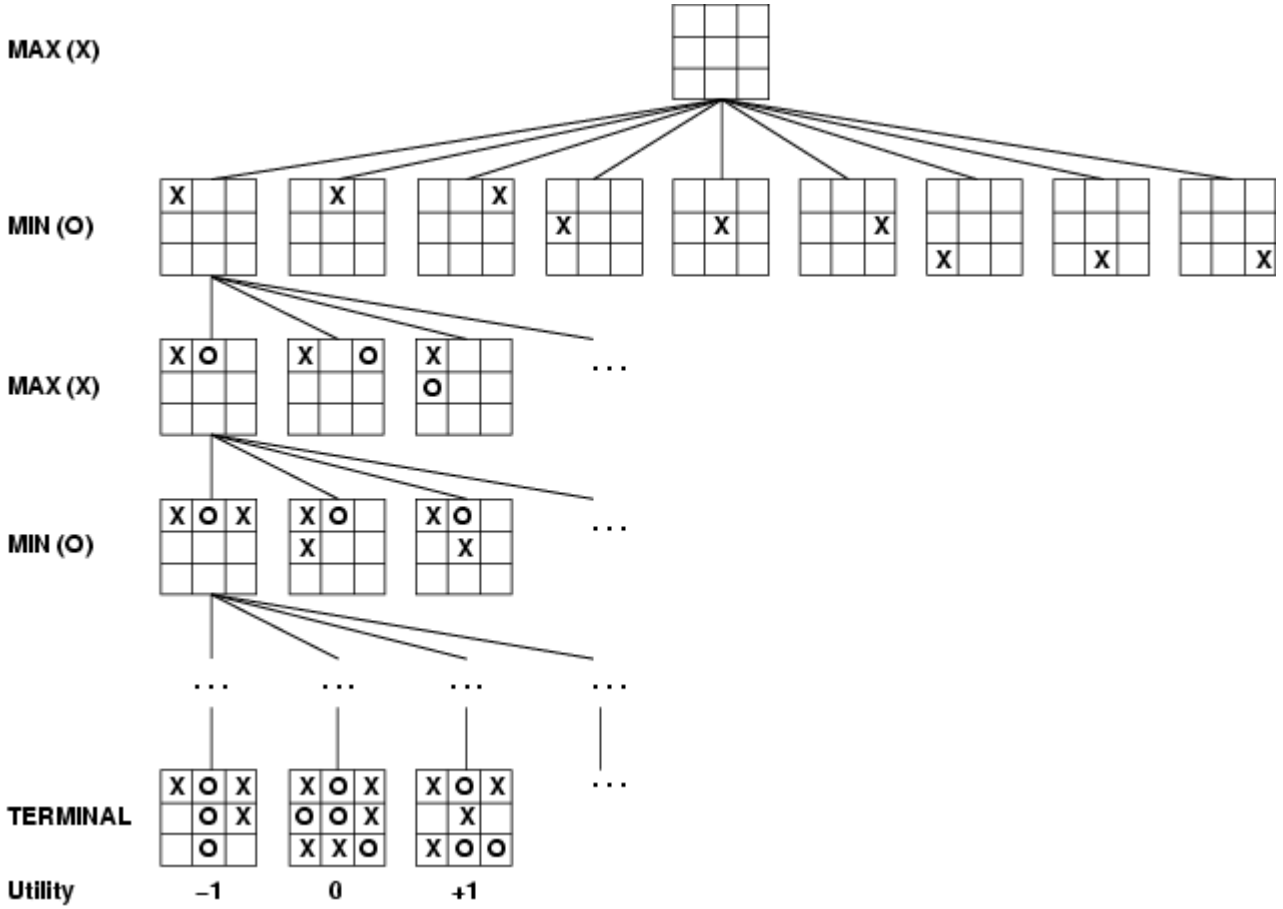
Минимаксна процедура

Обща характеристика

Метод за намиране на най-добрия ход на първия играч при предположение, че другият играч също играе оптимално. Изисква построяване на цялото ДС и намиране на оценките на листата (наричат се *статични оценки*).

Предполага се, че двамата играчи имат противоположни интереси, изразени в това, че единият търси стратегия, която води до получаване на позиция с максимална оценка (максимизиращ играч, MAX), а другият търси стратегия, която води до получаване на позиция с минимална оценка (минимизиращ играч, MIN).

Пример

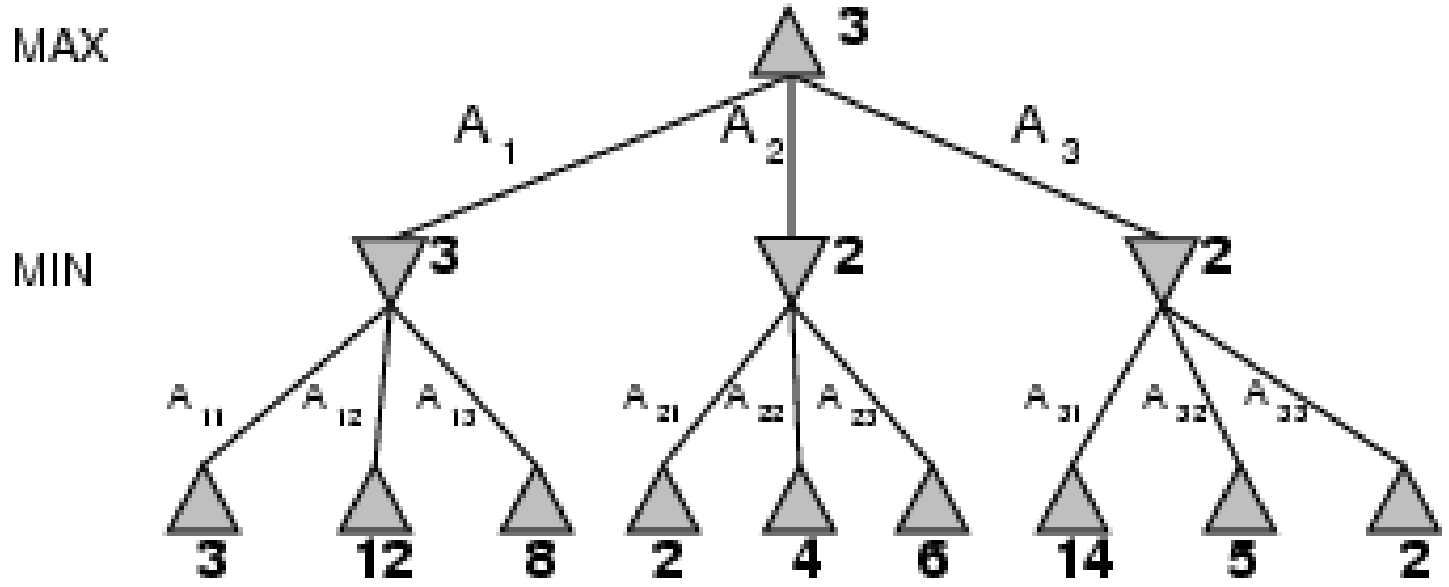


Минимаксната процедура е метод за получаване на оценките (наричат се *придобити* или *породени оценки*) на възлите от по-горните нива на ДС, които позволяват на първия играч да избере най-добрия си ход.

Получаване на оценките на възлите от ДС

Оценките се разпространяват отдолу нагоре, като всеки от възлите, съответни на ход на максимизиращия играч, получава оценка, равна на максималната от оценките на преките му наследници, а всеки от възлите, съответни на ход на минимизиращия играч, получава оценка, равна на минималната от оценките на преките му наследници.

Пример


MAX
MIN
3
A 1
A 2
A 3
3
2
2
A 11
A 12
A 13
A 21
A 22
A 23
A 31
A 32
A 33
3
12
8
2
4
6
14
5
2

```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
```

Оценка на минимаксната процедура

Изисква построяване на цялото ДС, което може да се окаже твърде голямо в общия случай на реална игра. Точен, но крайно неефективен и често нереализуем на практика метод.

Алфа-бета процедура

Идея на метода

Преодоляват се проблемите при минимаксната процедура, породени от обстоятелството, че там процесът на генерирането на ДС е напълно отделен от процеса на оценяване на позициите, което води до силна неефективност.

Тук листата се оценяват веднага след генерирането им и при първа възможност се пресмятат и съответните придобити (породени) оценки на възлите от по-горните нива или поне се намират подходящи горни граници на оценките на възлите от минимизиращите нива (съответни на ходове на минимизиращия играч) и/или долни граници на оценките на възлите от максимизиращите нива (съответни на ходове на максимизиращия играч). Така броят на операциите за намиране на същия резултат, както при минимаксната процедура, може да се намали значително.

Алфа-бета процедурата изисква генериране на ДС в дълбочина, при което се преценява безполезността от генерирането и оценяването на някои клонове, неоказващи влияние върху резултата.

Дефиниции

Алфа-стойност на даден максимизиращ възел се нарича текущо установената долна граница на породената му оценка.

Бета-стойност на даден минимизиращ възел се нарича текущо установената горна граница на породената му оценка.

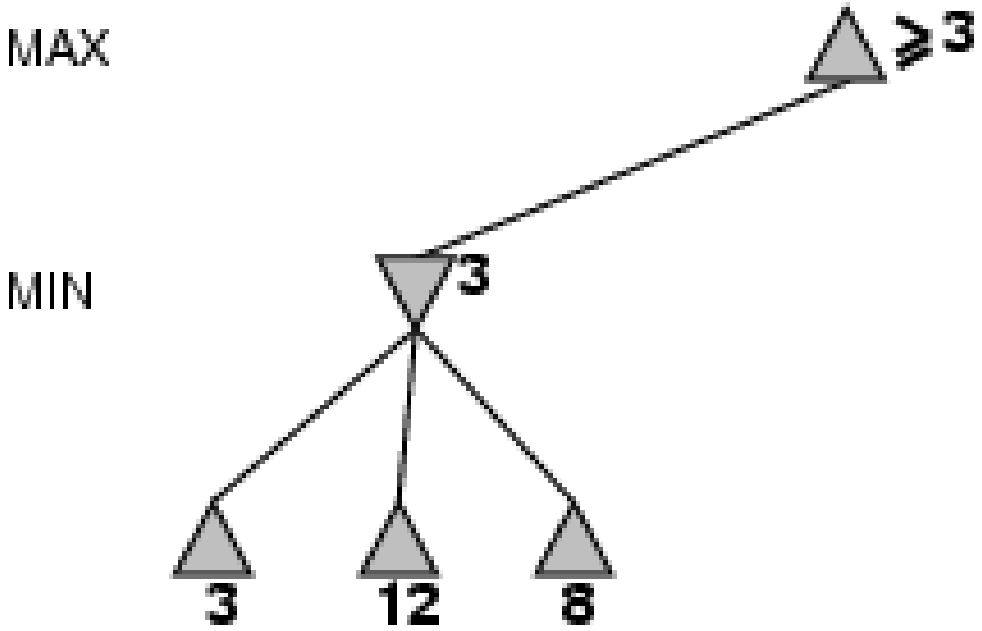
Първоначално определените алфа- и бета- стойности на възлите могат да се уточняват в процеса на работа, като алфа-стойностите могат само да растат, а бета-стойностите – само да намаляват.

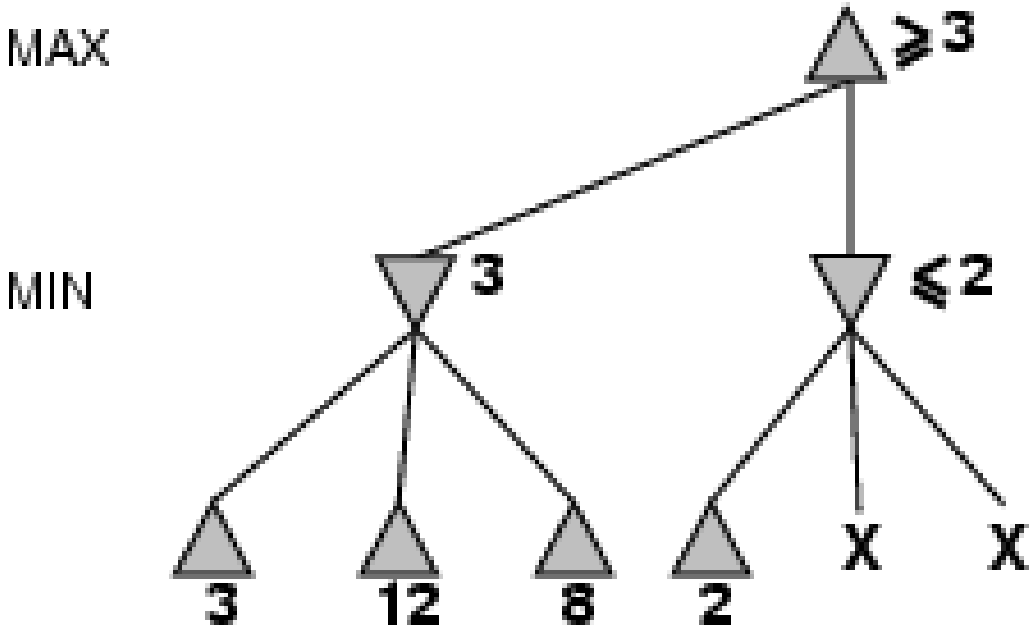
Описание на метода

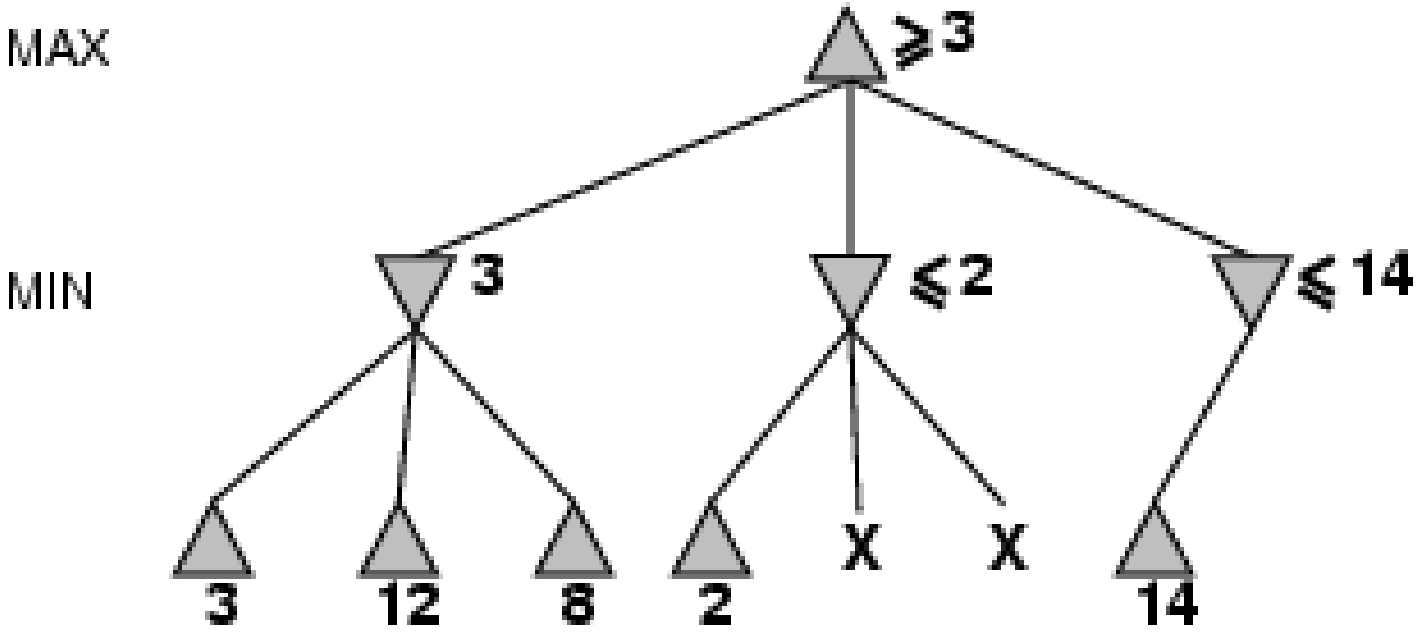
Правила за прекратяване на генерирането и търсенето:

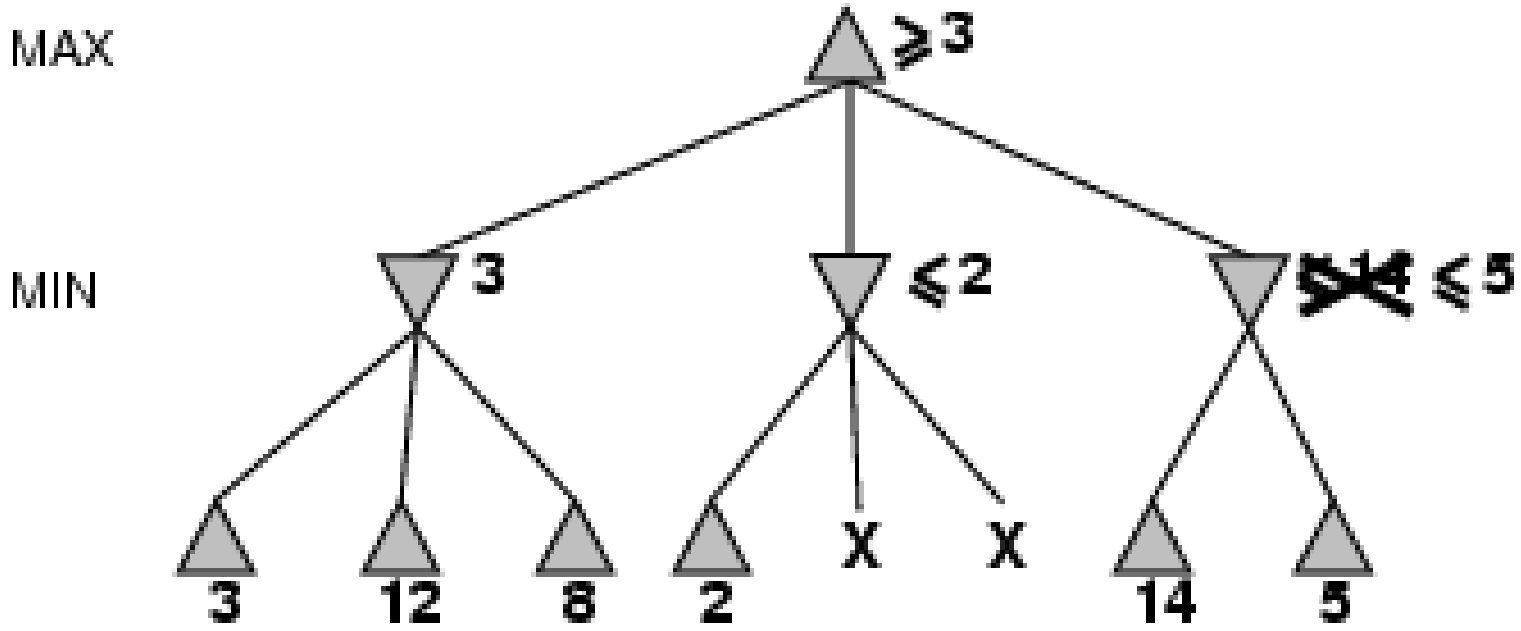
- *Алфа-отсичане.* Не е необходимо да се извършва генериране и търсене върху всяко поддърво, което произлиза от минимизиращ възел, бета-стойността на който е по-малка или равна на алфа-стойността на съответния максимизиращ родител.
- *Бета-отсичане.* Не е необходимо да се извършва генериране и търсене върху всяко поддърво, което произлиза от максимизиращ възел, алфа-стойността на който е по-голяма или равна на бета-стойността на съответния минимизиращ родител.

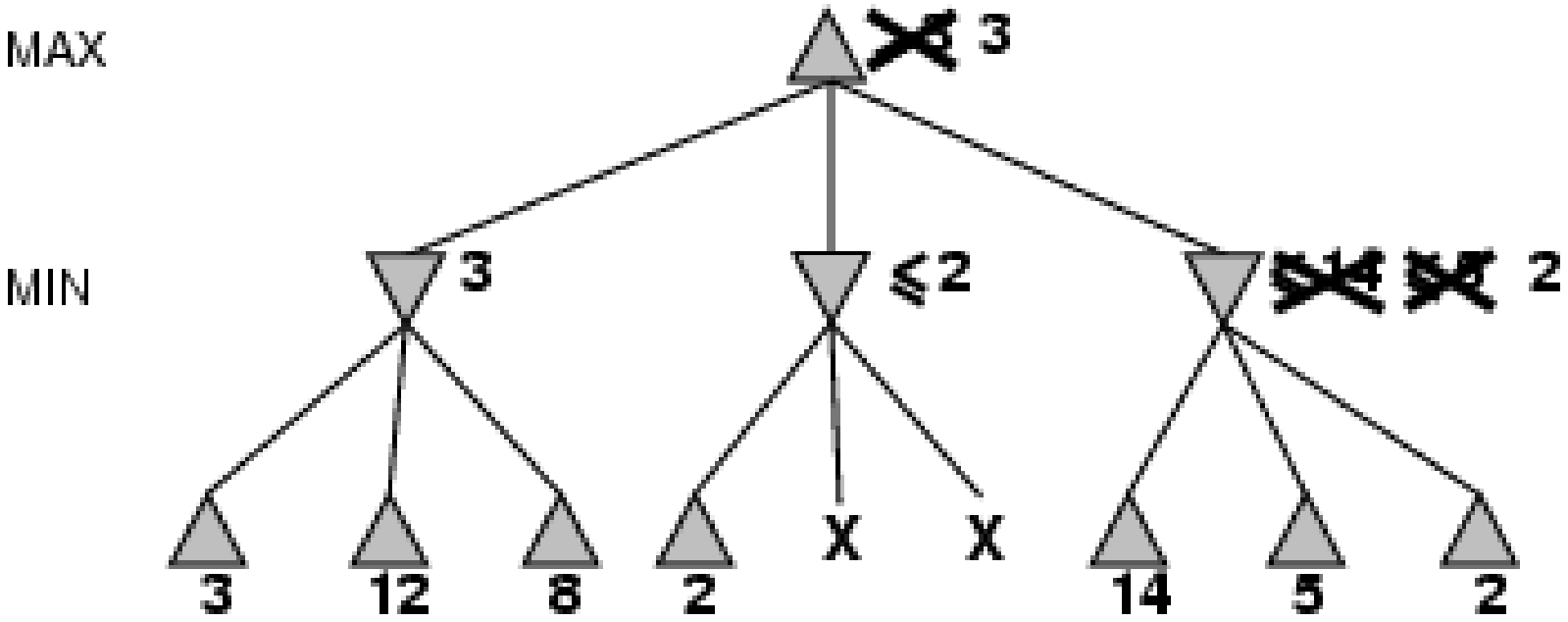
Пример




MAX
MIN
≥ 3
3
≤ 2
3
12
8
2
X
X


MAX
MIN
≥ 3
3
≤ 2
≤ 14
3
12
8
2
X
X
14


MAX
MIN
≥ 3
3
≤ 2
≤ 5
3
12
8
2
X
X
14
5


MAX
MIN
3
3
≤2
≤2
3
12
8
2
X
X
14
5
2

Забележка. В много литературни източници понятията алфа-стойност и бета-стойност се дефинират не локално (по отношение на даден възел от съответния тип), а „глобално“ (по отношение на текущо изследвания път в игровото дърво). Под *алфа-стойност* се разбира установената до момента най-добра стойност, която може да достигне максимизиращият играч по текущо изследвания път. Аналогично под *бета-стойност* се разбира установената до момента най-добра стойност, която може да достигне минимизиращият играч по текущо изследвания път. И при този подход към дефиницията на понятията алфа-стойност и бета-стойност правилата за алфа-отсичането и бета-отсичането могат да бъдат формулирани по начин, аналогичен на разгледания по-горе.

function ALPHA-BETA-SEARCH(*state*) **returns** *an action*
  inputs: *state*, current state in game

  $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
  return the *action* in SUCCESSORS(*state*) with value v

---

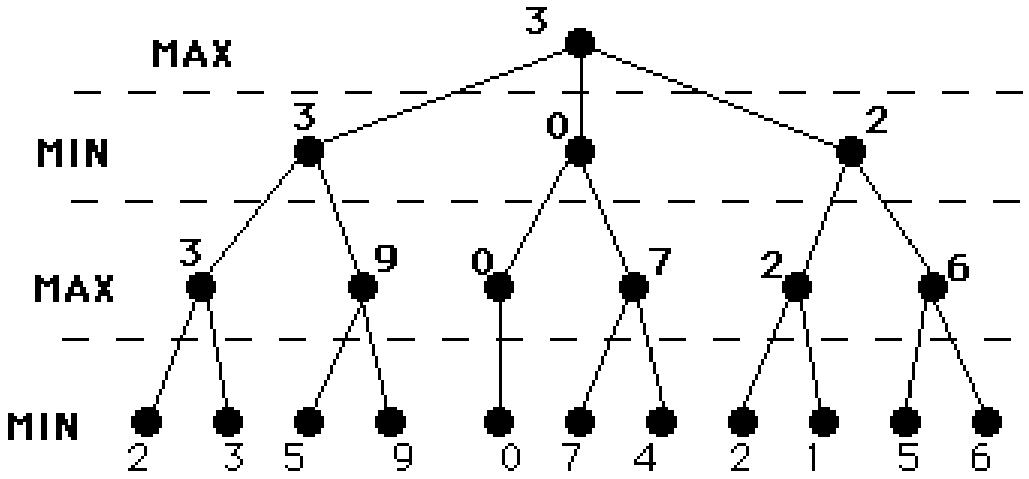
function MAX-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  inputs: *state*, current state in game
    α, the value of the best alternative for MAX along the path to *state*
    β, the value of the best alternative for MIN along the path to *state*

  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s, α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

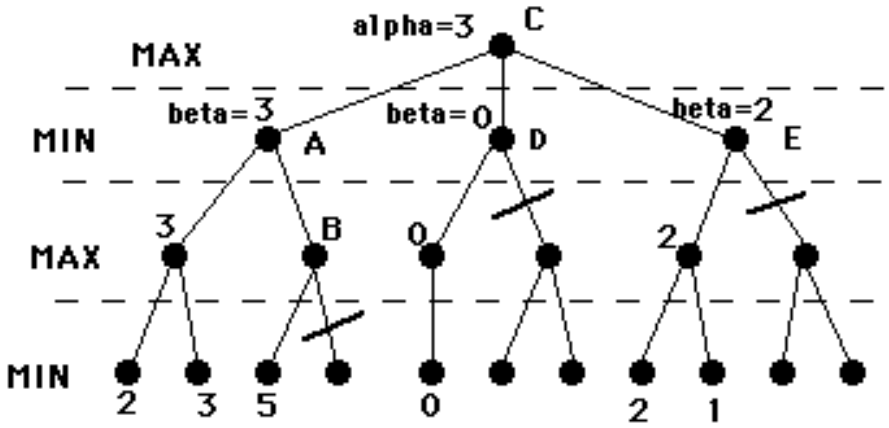
```
function MIN-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for a, s in SUCCESSORS(state) do
    v ← MIN(v, MAX-VALUE(s, α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

Следващи (по-сложни) примери

Пример 1



Minimax of a hypothetical search space. Leaf nodes show heuristic values.



1. Start at C. Descend to full-ply depth and assign the heuristic to a state and all siblings (MIN 2, 3). Back up these values to their parent node (MAX 3).

2. Offer this value to the grandparent (A), as its beta value. So, **A has beta=3.** A will be no larger than 3

3. Descend to A's other grandchildren. Terminate the search of their parent if any grandchildren is >= A's beta. **Node B is beta-pruned**, as shown, because its value must be at least 5.

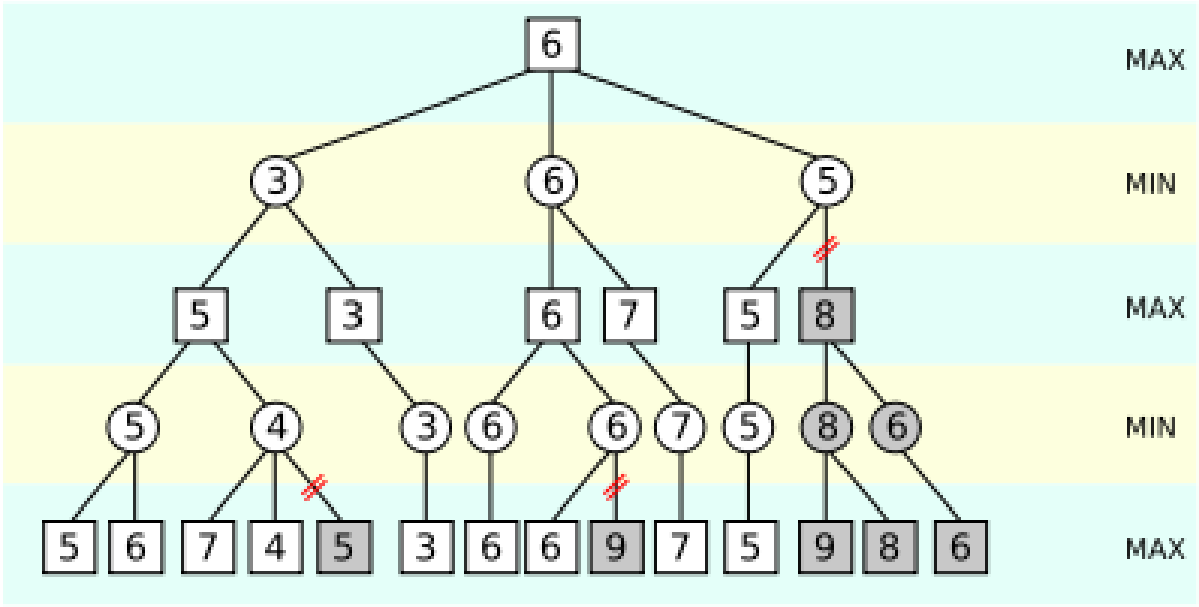
4. Once A's value is known, offer it to its parent (C) as its alpha value. So **C has alpha=3**. C will be no smaller than 3.

5. Repeat this process, descending to C's great grandchildren (0) in a depth-first fashion. **D is alpha-pruned**, because no matter what happens on its right branch, it cannot be greater than 0.

6. Repeating on E, **E is alpha-pruned** because its beta value (2) is less than its parent's alpha value (3). So no matter what happens on its right branch, E cannot have a value greater than 2.

7. Therefore **C is 3.**

Пример 2



Оценка на алфа-бета процедурата

При използване на алфа-бета процедурата се получава напълно точен резултат.

Броят на генерираните и оценени възли при алфа-бета процедурата е най-малко от порядъка на $b^{d/2}$. Тази стойност се получава в идеалния случай, когато оценките на листата са наредени максимално добре. Следователно алфа-бета процедурата намалява скоростта на развитие на комбинаторния взрив, но не го предотвратява.

Практически реализации на минимаксната процедура

- Отсичане на игровото дърво от дадено ниво надолу и намиране на евристични оценки на получените терминални възли (които в общия случай не са листа в ДС).
- Идеалната евристика би трябвало да отразява вероятността, че съответният играч ще победи в дадената (оценяваната) позиция. При добре познатите игри се използват опитът и интуицията на силните играчи. Евристиките могат да бъдат усъвършенствани експериментално.

- *Пример* за елементарна евристика при шахмата: разлика на тегловните суми от броя на белите и черните фигури (при което например 1 пешка е равна на 1/3 кон или офицер, 1/5 топ или 1/9 дама). Разположението на фигурите може да бъде отчетено чрез специално подбрани коефициенти.

- Определяне на нивото на отсичане (предполага се, че евристичните оценки на възлите от по-долните нива се намират по-лесно и са по-точни):
  - Предварително задаване на горна граница на дълбочината на генерираната част от ДС;
  - Използване на итеративно търсене по нива при предварително зададено максимално време за работа;
  - Търсене на затихване: генериране на дървото до достигане на „спокойна“ позиция (позиция, която не предполага рязка промяна на оценката в непосредствено бъдеще).