Основы программирования на языке Java

Решение игр. Алгоритм minimax. Альфа-бета отсечение.

http://courses.cs.ttu.ee/pages/ITI0011

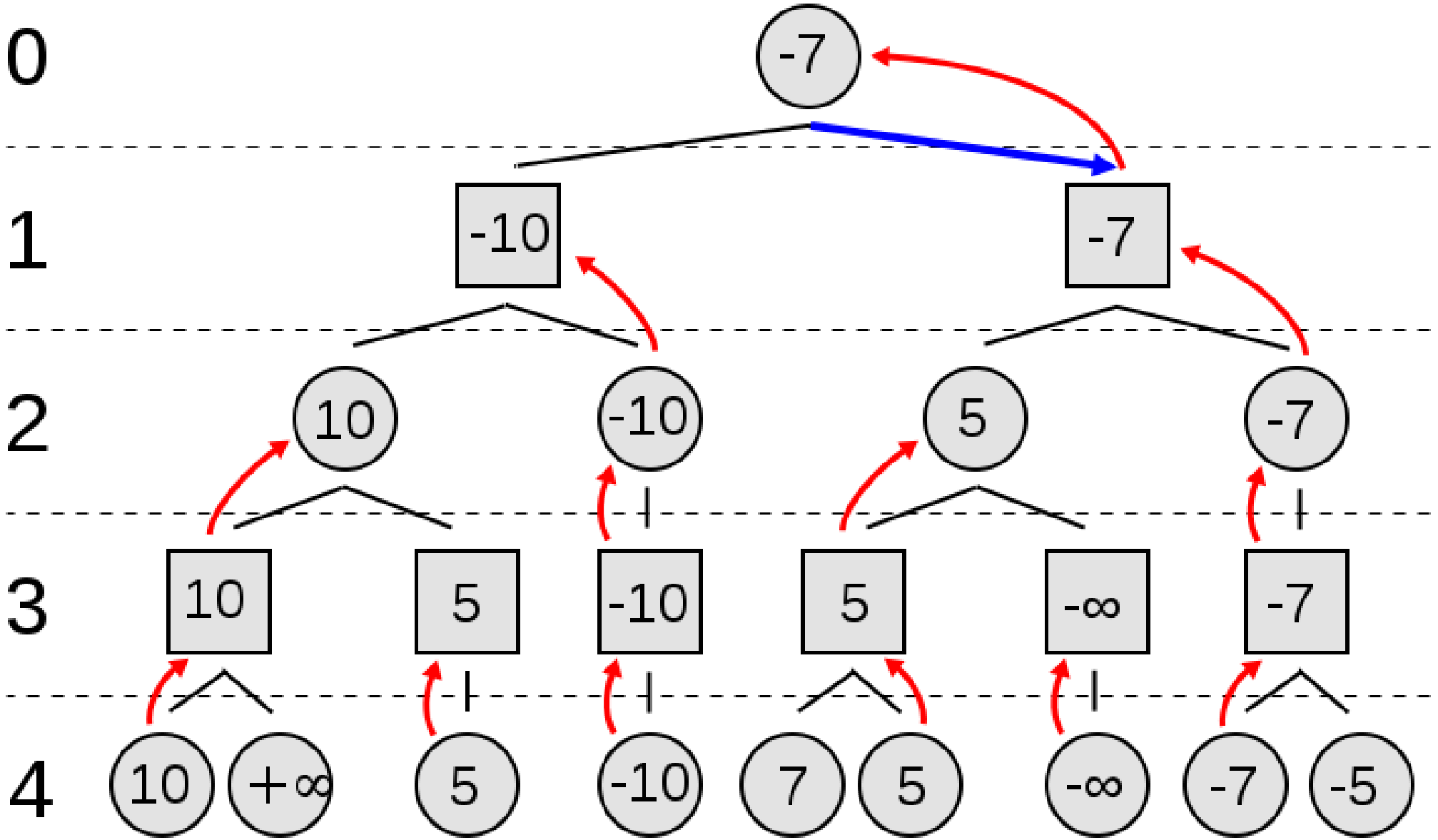
Minimax

- Рекурсивный алгоритм решения игр.
- Расчитывает оптимальную стратегию игры для заданного игрока.
- Предлагает оптимальный следующий ход в заданном *состоянии игры*
- С каждым состоянием связано значение – результат функции оценки состояния (эвристика).
- Игрок выполняет самый выгодный ход (ход с наибольшим значением)
- Просматривает несколько ходов вперед (глубина поиска).
- Возвращает оценку состояния, которого можно достичь после *K* ходов.
- Игрок и его оппонент (максимизирующий и минимизирующий игроки).
- Максимизация минимального выигрыша.
- Минимизация максимальных потерь.

Minimax (contd.)

- Два соперничающих игрока, которые ходят поочередно.
- Три параметра:
  - Текущее состояние игры (игровое поле).
  - Текущая глубина поиска (кол-во просматриваемых ходов).
  - Текущий игрок.
- Рекурсия останавливается в терминальных состояниях игры.
- Терминальное состояние – состояние, в котором больше нельзя сделать легитимный ход (выигрыш, проигрыш, ничья, ограничение по глубине поиска).
- Достигнув терминального состояния производится эвристическая оценка этого состояния и возвращается результат.
- Результат промежуточных состояний рассчитывается на основе результатов производных состояний.
- Результат анализируемого состояния и есть результат анализа.

Minimax (step by step)

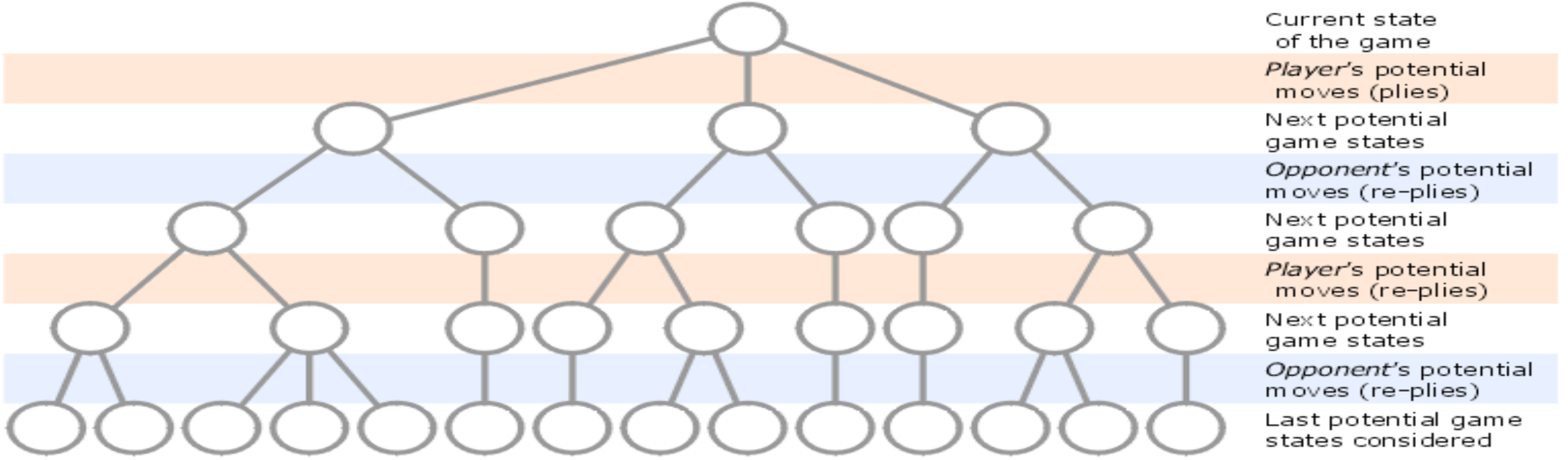


Minimax (псевдокод)

```
function minimax(state, depth, maximizingPlayer) {
    if depth = 0 or state is a terminal state
        return score(state)
    if maximizingPlayer
        foreach derived:        values[] += minimax(derived, depth-1, FALSE)
        return max { values[] }
    else
        foreach derived:        values[] += minimax(derived, depth-1, TRUE)
        return min { values[] }
}
```

Initial call: minimax(initial_state, depth, TRUE)

Minimax demo (live)



Альфа-бета отсечение

- Оптимизация алгоритма minimax.
- Сокращает пространство перебора (иерархию состояний).
- Прекращает дальнейшее исследование *слабо доминируемых стратегий.*
- Некоторые из ветвей дерева могут быть исключены из поиска, послк того как хотя бы одна из ветвей рассмотрена полностью.
- Отсечения происходят на каждом уровне вложенности кроме последнего.
- Возвращает то же самое значение, что вернул бы minimax, только не спускается в ветви дерева, которые не повлияют на окончательный результат (не улучшат его).
- В итоге – пространство перебора сокращается, алгоритм работает быстрее.
- Можно увеличить глубину поиска сохраняя время работы алгоритма.

Альфа-бета отсечение (contd.)

- Увеличивает глубину поиска minimax в чуть более чем 1.5 раза.
- При условии оптимального (или околооптимального) расположения ходов (наивыгоднейшие ходы располагаются самыми первыми в массиве возможных ходов).
- При наихудшем расположении ходов может понизить эффективность работы по сравнению с эффективностью minimax.
- Хорошо продуманная функция сортировки последовательности возможных ходов (эвристика) – экспоненциально сократит пространство перебора.
- Эвристика – должна приводить к наиболее ранним альфа-бета отсечениям (killer heuristics, refutation tables, …).
- Расстановка ходов в начале игры более «дешевая» в ресурсах (немного возможных состояний).
- На практике, расстановка ходов в текущем состоянии основывается на расстановке ходов в предыдущем состоянии (с меньшим пространством возможных состояний).

Альфа-бета отсечение (contd.)

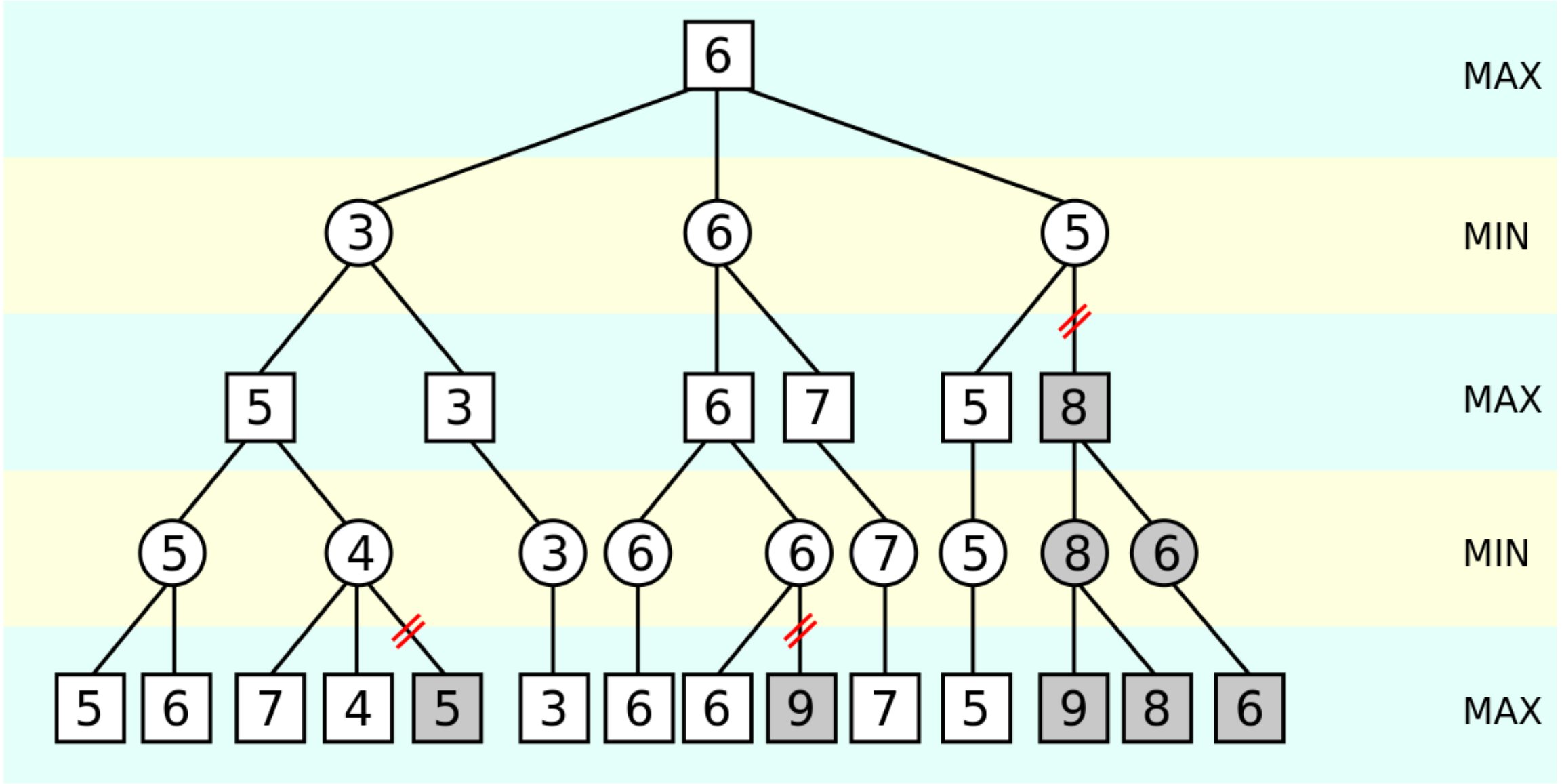
- Текущее состояние характеризуется двумя параметрами – α и β.
- Фактор ветвления (branching factor).
- α – максимальный результат, который гарантирован для максимизирующего игрока. Изначально -∞ (самый низкий возможный счет).
- β – минимальный результат, который гарантирован для минимизирующего игрока. Изначально +∞ (самый низкий возможный счет).
- При выполнении условия $\beta \leq \alpha$ текущую ветвь не улучшит результата.
- Текущее состояние можно исключить из пространства поиска и сразу же переходить к следующему.

Альфа-бета отсечение (псевдокод)

```
function alphabeta(state, depth, α, β, maximizingPlayer){
 if depth = 0 or state is a terminal state
      return score(state)
   if maximizingPlayer
      foreach derived:
         α  = max { α, alphabeta(derived, depth-1, α, β, FALSE) }
         if β ≤ α     break  /* β-cutoff */
      return α
   else
      foreach derived:
         β = min { β , alphabeta(derived, depth-1, α, β, TRUE) }
         if β ≤ α     break  /* α-cutoff */
      return β
}
```

Initial call: alphabeta(initial_state,depth, -∞, +∞, TRUE)

Альфа-бета отсечение (step by step)



Альфа-бета отсечение demo (live)