
INSTITUT TEKNOLOGI BANDUNG
1920

Modul 4: Adversarial Search

Adversarial Search

Masayu Leylia Khodra
(masayu@informatika.org)

KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)

Adversarial Search

Non-Adversarial Search

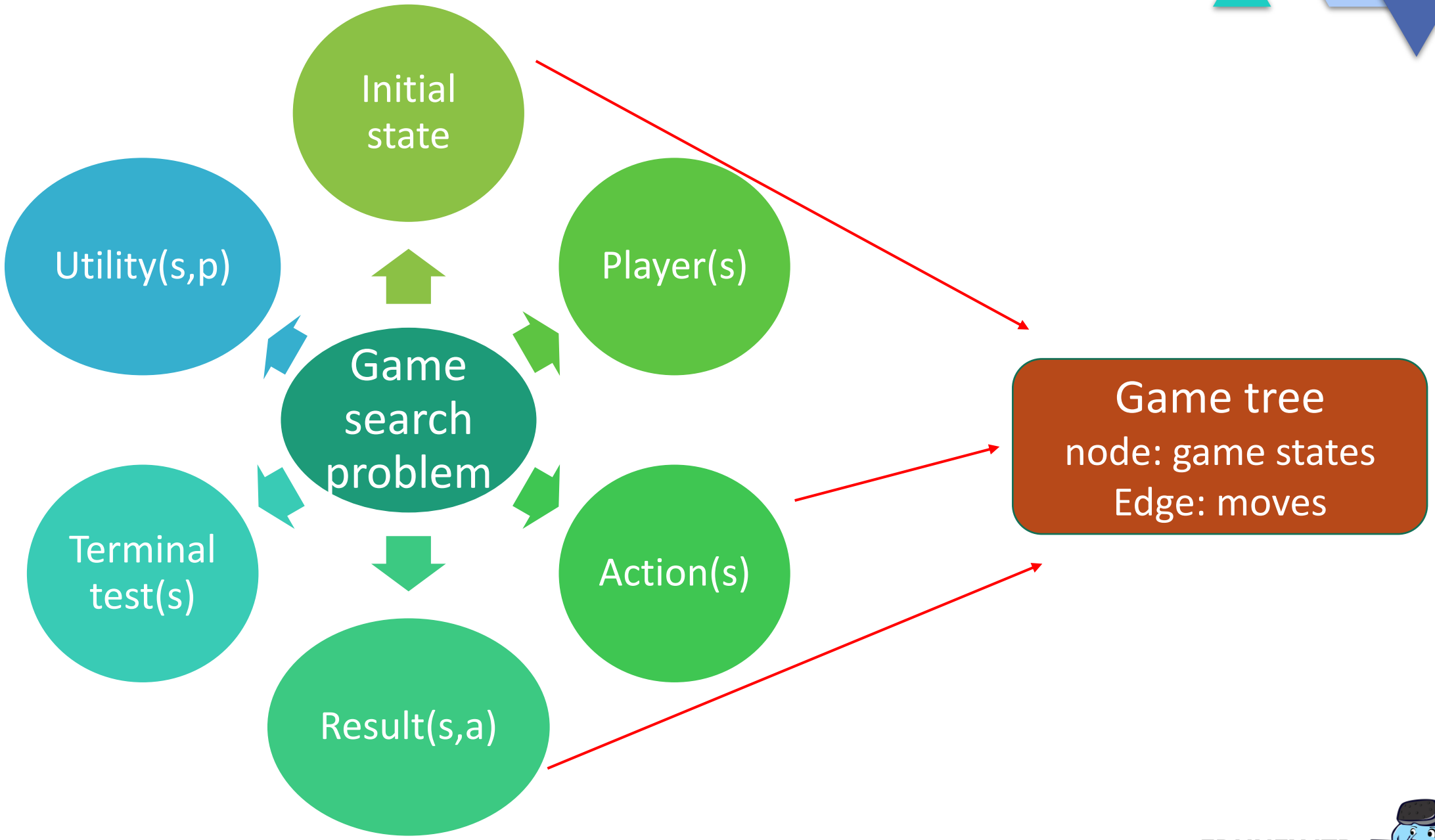
Approximate solution: strategy

Multiagent: competitive environment (games)

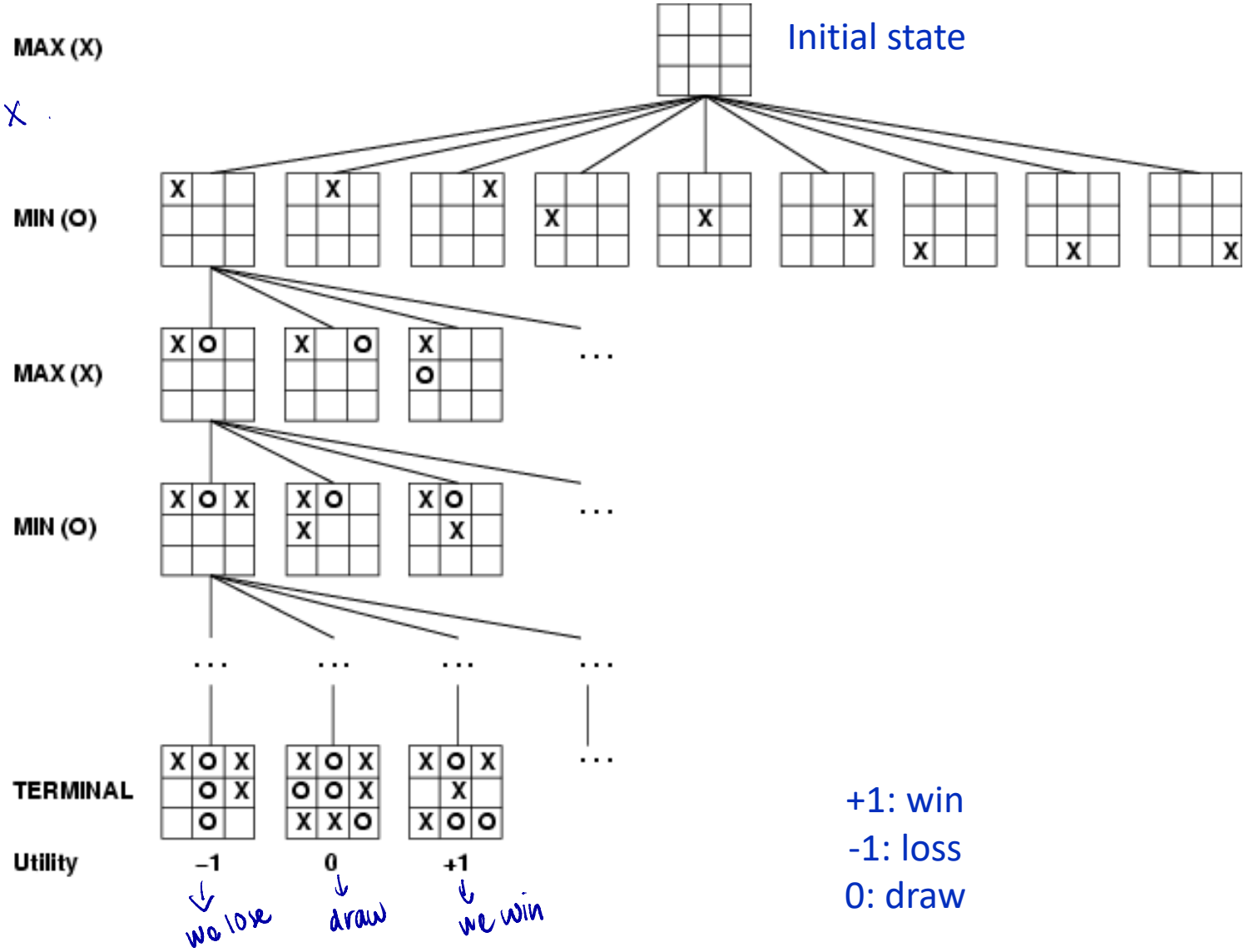
Optimize solution: path to goal or solution state

Deterministic and fully observable

Turn-taking


Initial state
Utility(s,p)
Player(s)
Game search problem
Terminal test(s)
Action(s)
Result(s,a)
Game tree
node: game states
Edge: moves

Game tree (2-players, tic-tac-toe)



node: game states
Edge: moves

+1: win
-1: loss
0: draw

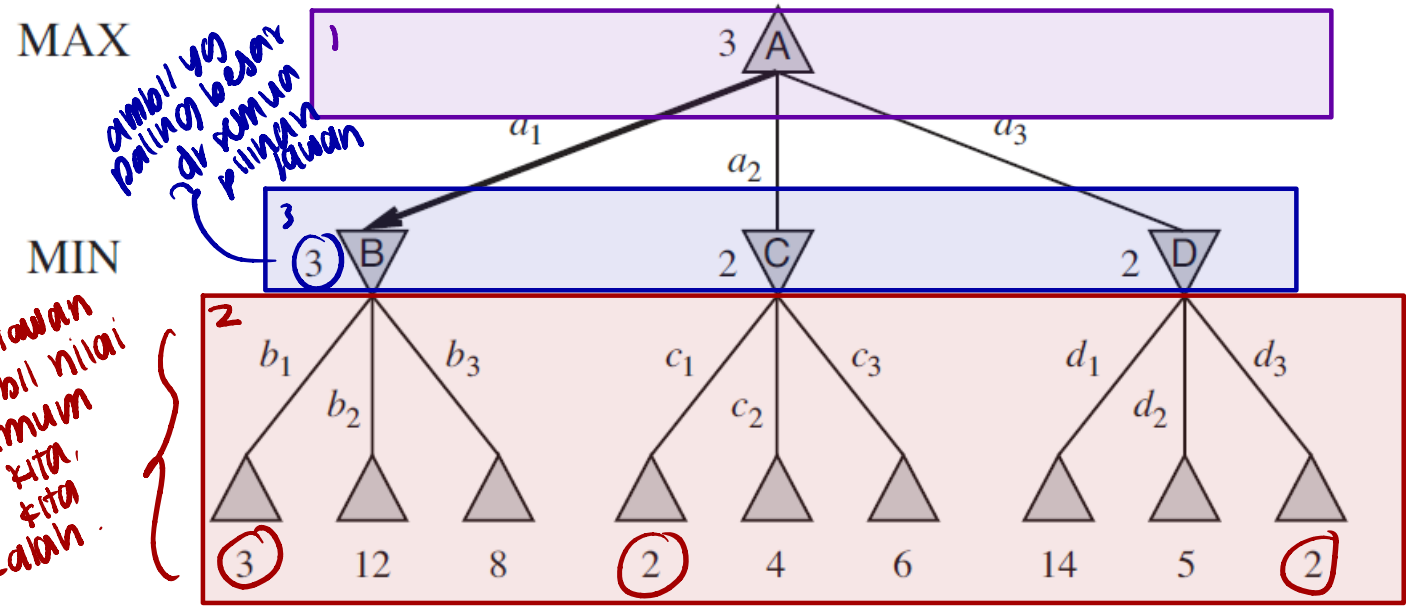
Optimal Decision with Minimax Algorithm

function MINIMAX-DECISION($state$) **returns** *an action*
  return $\arg\max_{a \in \text{ACTIONS}(s)}$ MIN-VALUE(RESULT($state, a$))

function MIN-VALUE($state$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow \infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s, a)))
  return v

function MAX-VALUE($state$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow -\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

Minimax: 2-ply Game Tree

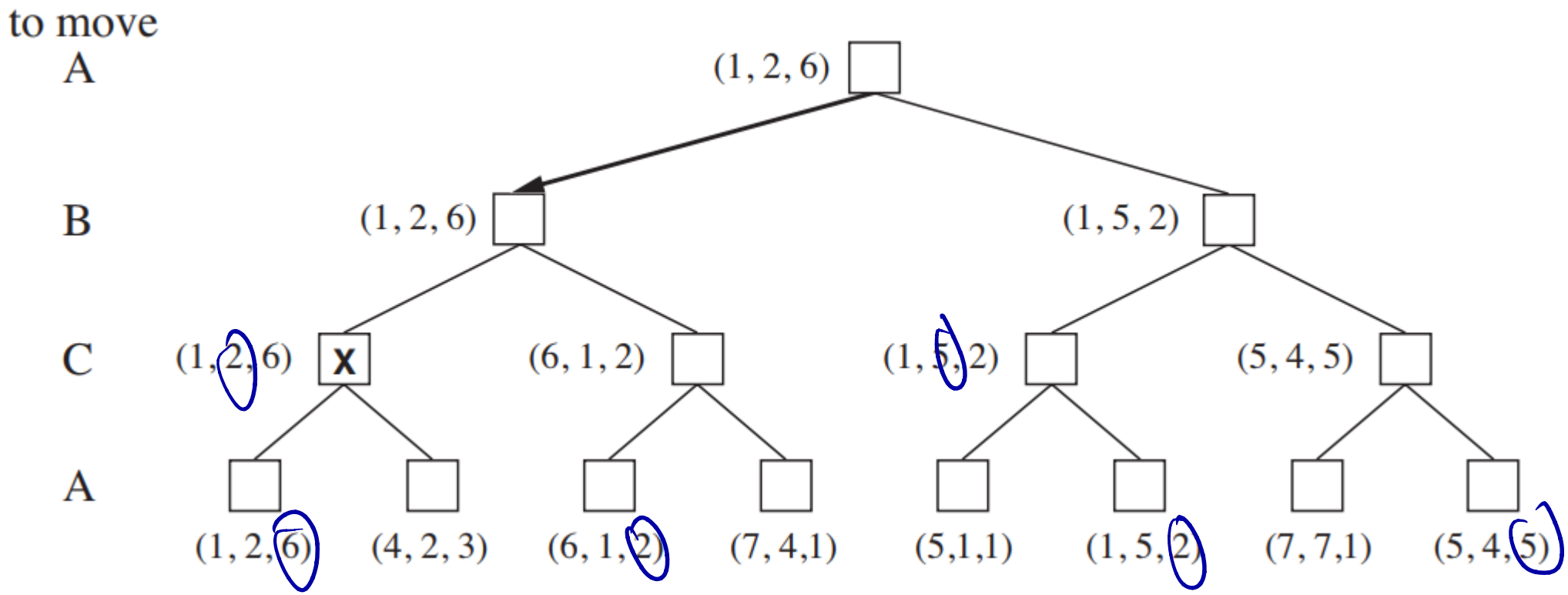


Minimax(B)=min(
Minimax(result(B,b1)),
Minimax(result(B,b2)),
Minimax(result(B,b3)))
=min(3,12,8)=3

Minimax(A)=max(Minimax(B),
Minimax(C),Minimax(D))
=max(3,2,2)=3

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \end{cases}$$

Minimax 3-ply in Multiplayer Games



Minimax Properties

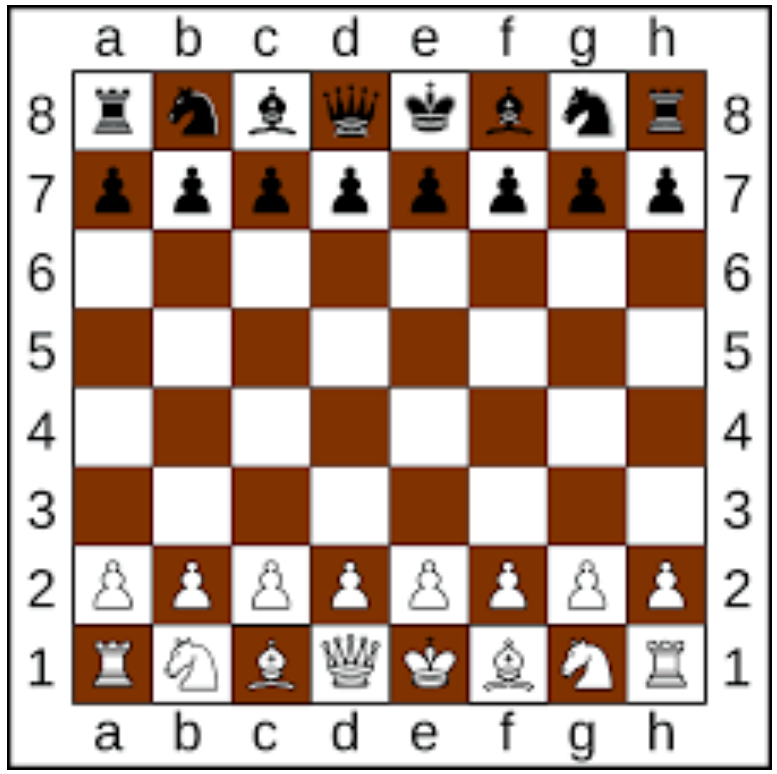
Complete?
Yes (if tree is finite)

Optimal?
Yes (against an optimal opponent)

Time complexity?
$O(b^m)$

Space complexity?
O(bm) (depth-first exploration)

Minimax for Chess

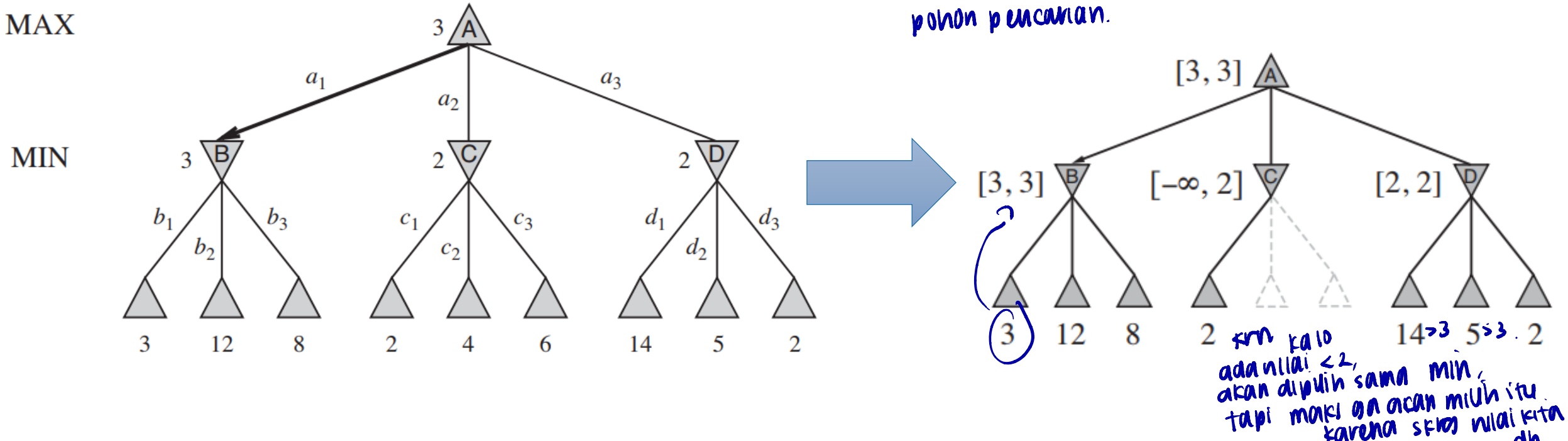


https://commons.wikimedia.org/wiki/File:AAA_SVG_Chessboard_and_chess_pieces_02.svg

- Branching factor: 35 (avg)
- Games often 50 moves for each player → m=100
- Game states is exponential in the depth of game tree.
- Exact solution is completely infeasible

αβSearch: Minimax with Pruning

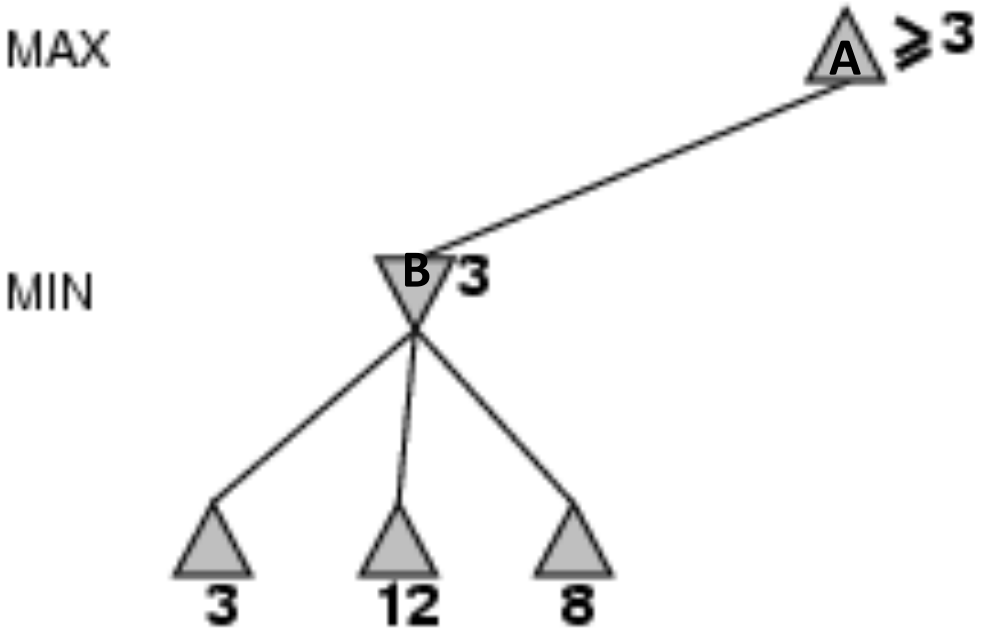
tidak mengubah hasil akhir, hanya memangkas pohon pencarian.



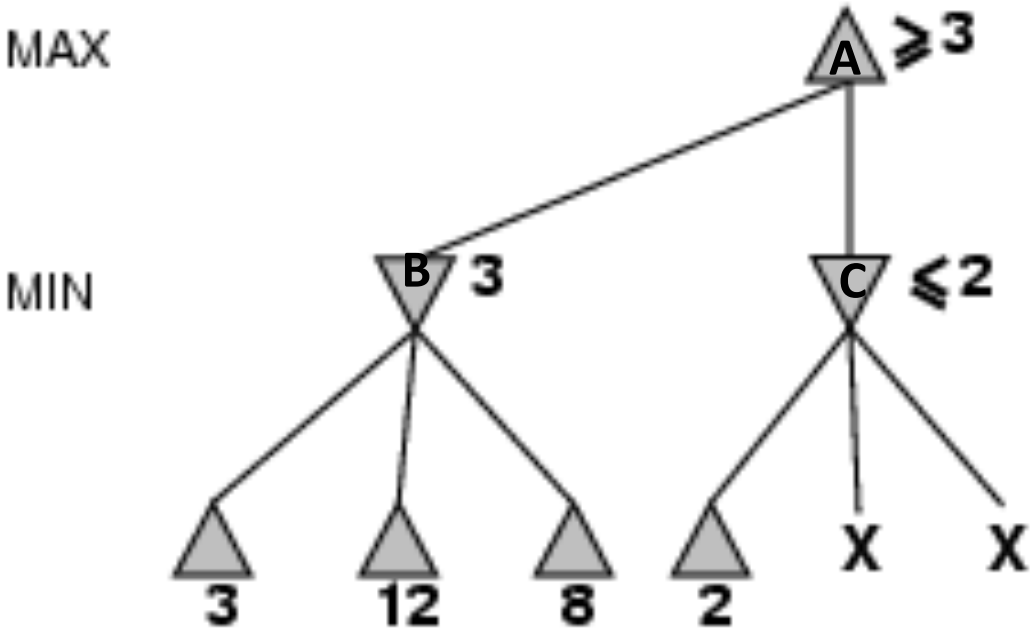
kalo ada nilai < 2, akan dipilih sama min, tapi maks gak akan milih itu karena sdh ada nilai kita 3.

Pruning does not affect final result. It returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision.

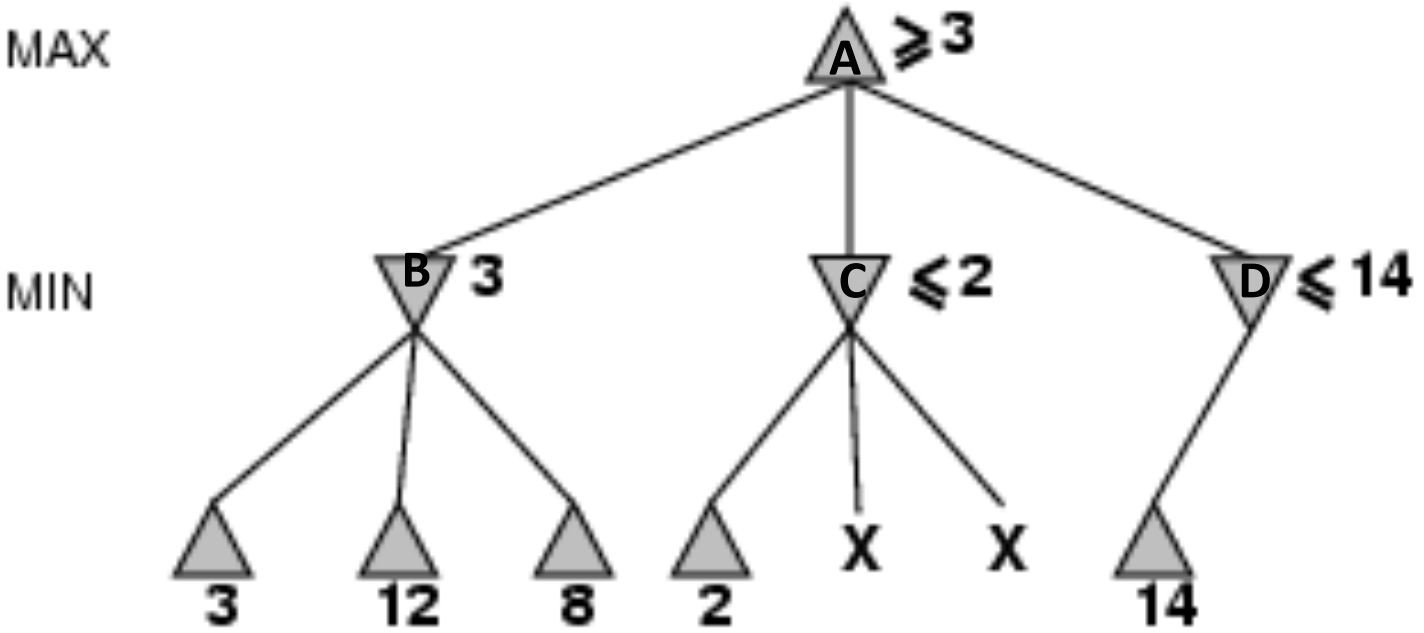
α-β pruning example



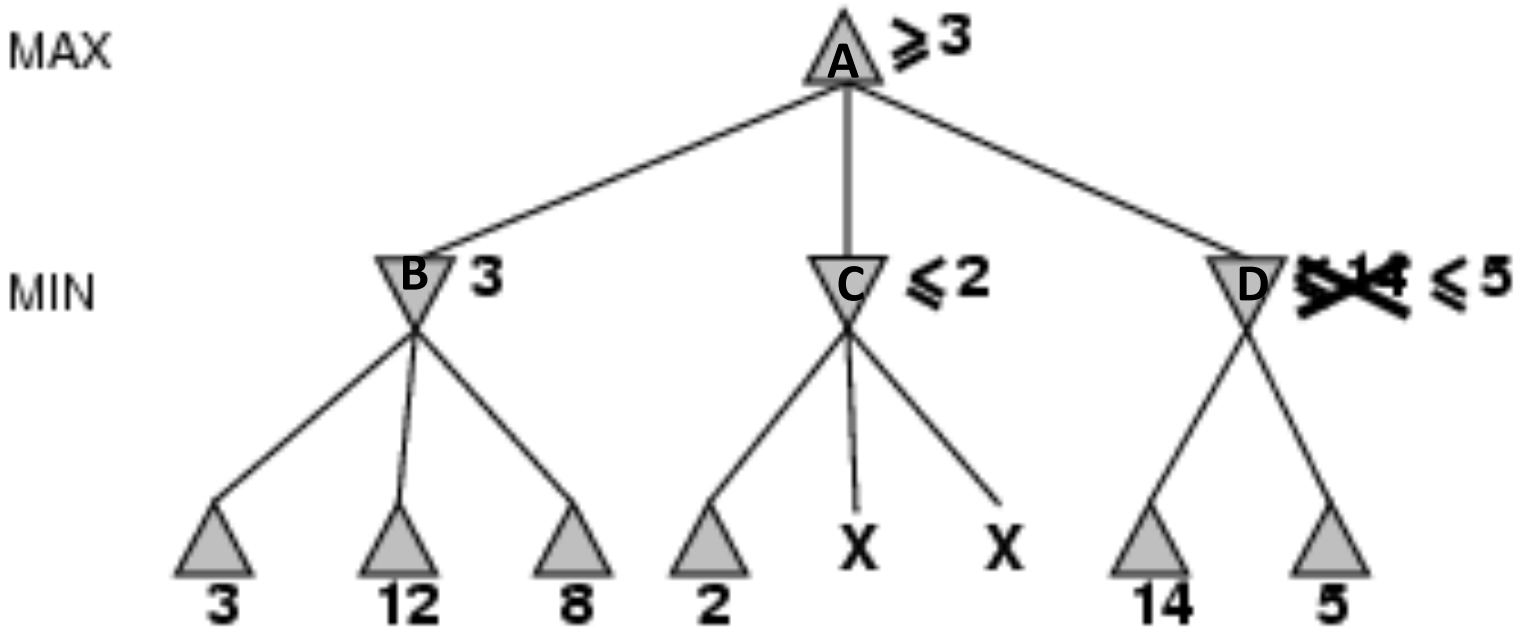
α-β pruning example



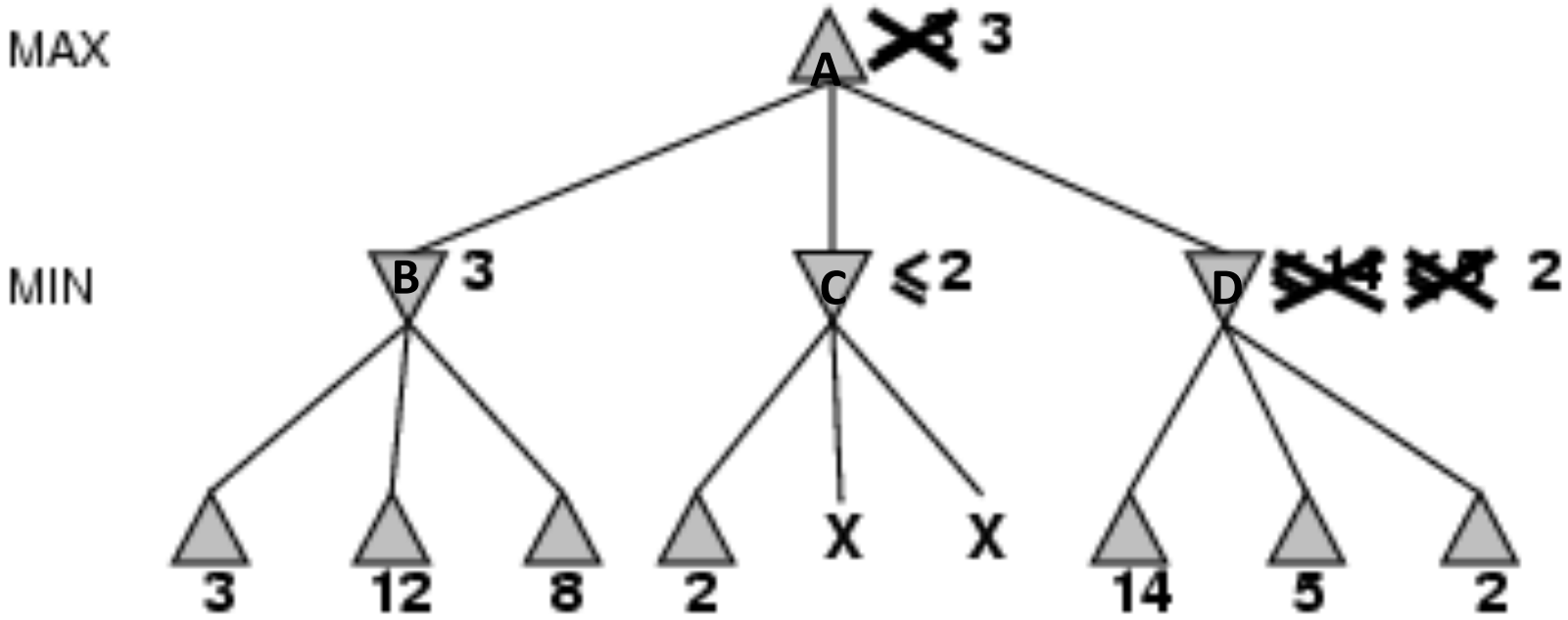
α-β pruning example



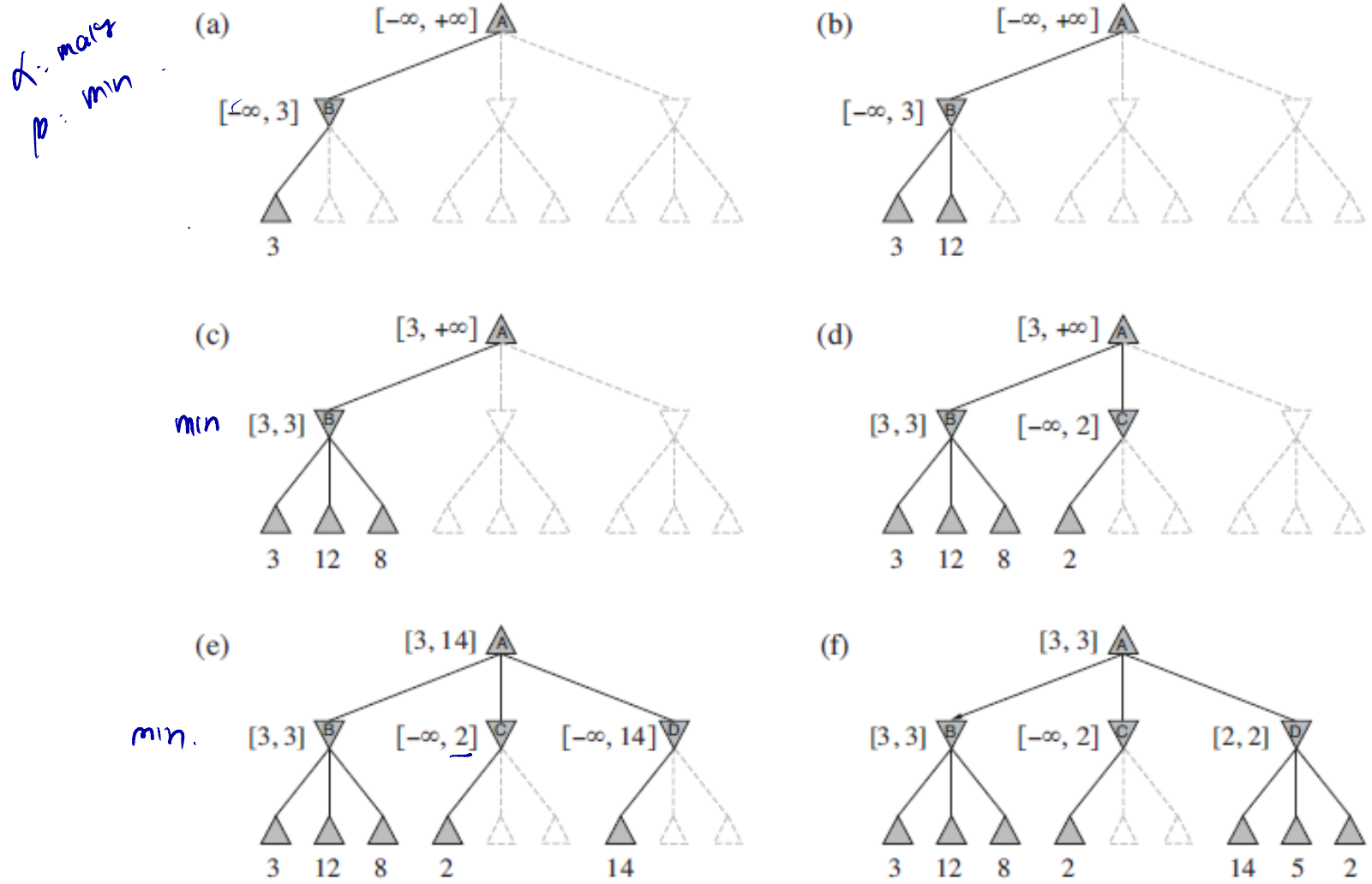
α-β pruning example



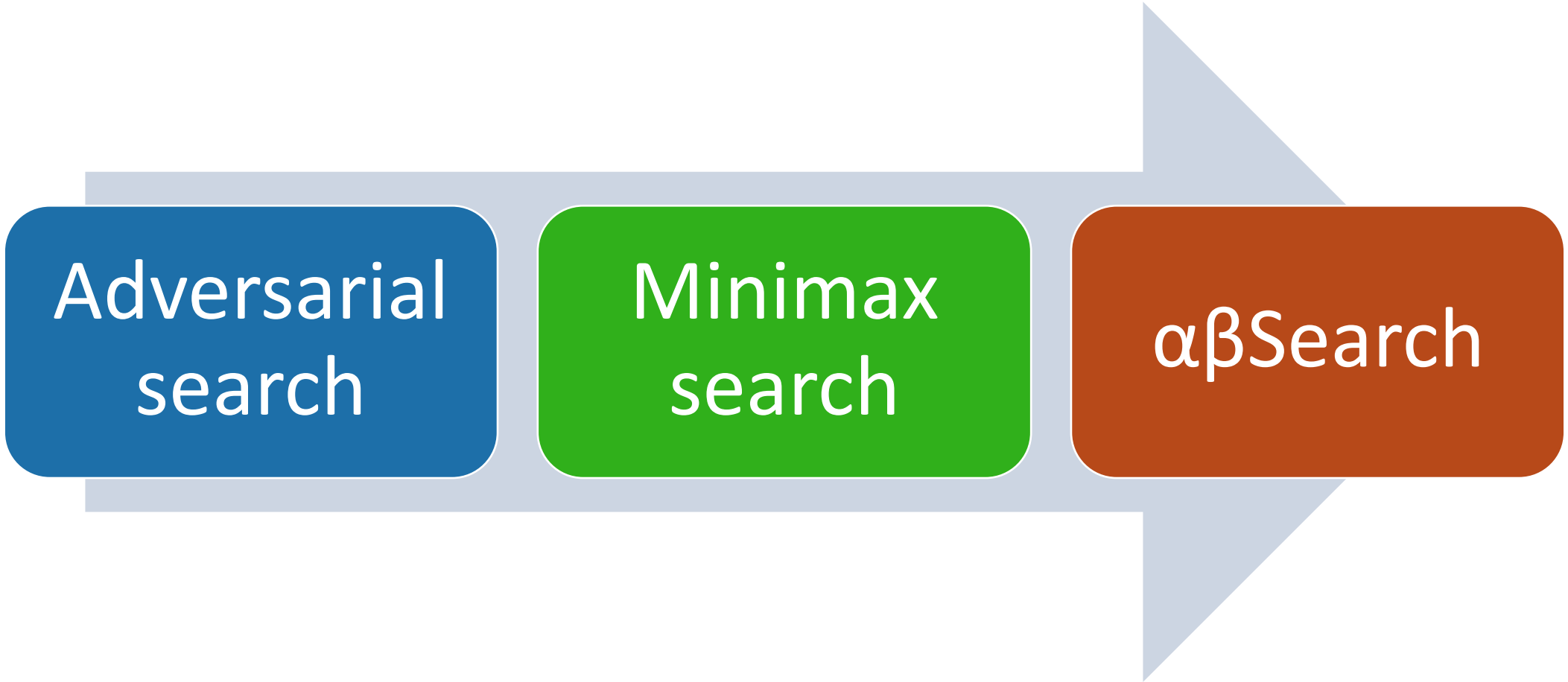
α-β pruning example



Algorithm illustration with α-β value



Summary



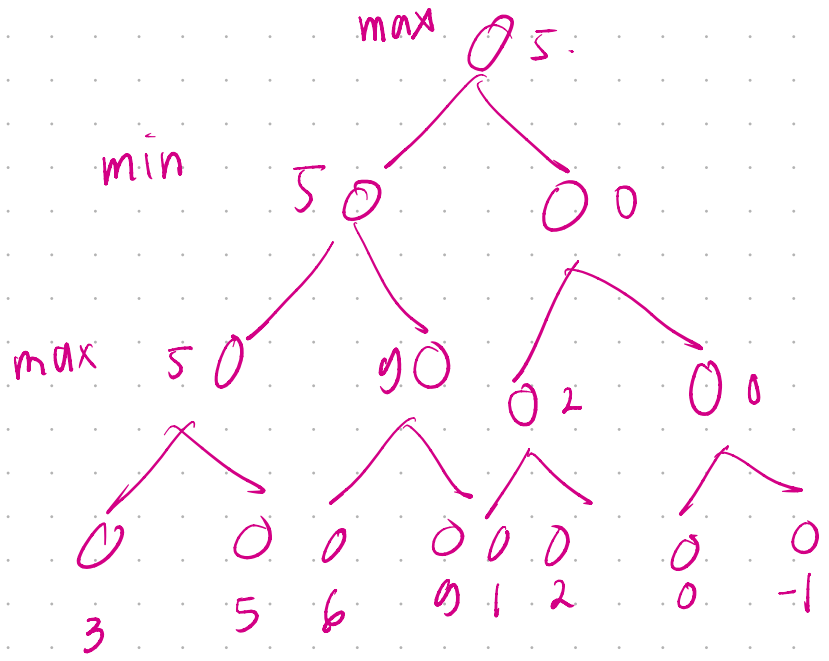

INSTITUT TEKNOLOGI BANDUNG
1920

Soal Adversarial Search

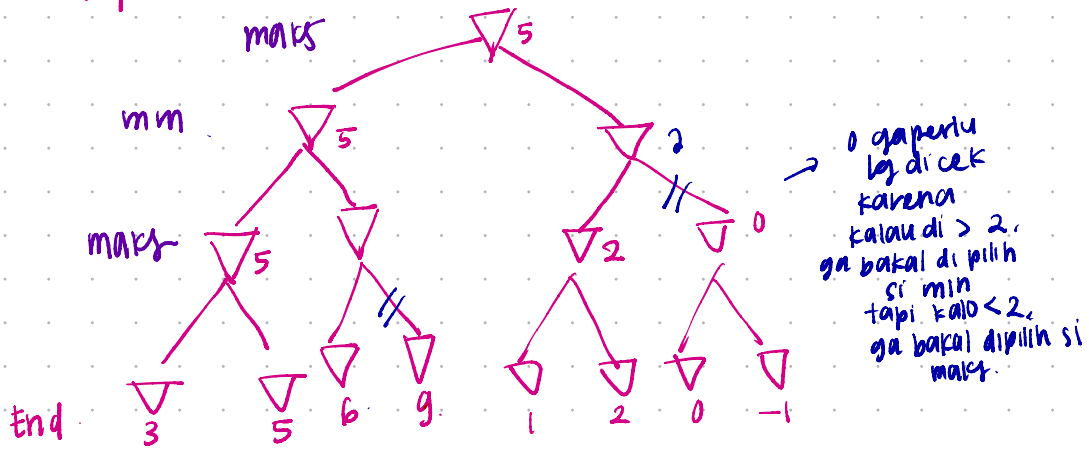
Terdapat suatu permainan berupa *3-ply game* sebagai berikut. Langkah pertama MAX yang bergerak dan memiliki dua kemungkinan langkah yaitu L atau R. Langkah berikutnya dilakukan MIN dengan dua kemungkinan langkah juga yaitu L atau R. Langkah terakhir dilakukan oleh MAX dengan dua kemungkinan langkah yaitu L atau R. Banyaknya kemungkinan urutan langkah adalah 8. Nilai pay-offs untuk MAX pada setiap kemungkinan urutan langkah adalah sebagai berikut: LLL = 3; LLR = 5; LRL = 6; LRR = 9; RLL = 1; RLR = 2; RRL = 0; dan RRR = -1. MAX berusaha memaksimalkan nilai payoffs nya, sedangkan MIN berusaha meminimalkan nilai payoffs MAX.

a) Gambarkan pohon *3-ply game* tersebut lengkap dengan nilai payoffs dari MAX pada setiap simpul pohon, dengan asumsi semua pemain memilih aksi secara rasional.

b) Terapkan *alpha-beta pruning* saat melakukan pencarian, dan ilustrasikan dengan gambar pohon untuk **setiap *pruning*** yang mungkin dilakukan pada cabang pohon dan jelaskan alasannya. Aksi L dieksplorasi terlebih dahulu sebelum aksi R dalam tahapan pencarian.


max
5
min
5
0
max
5
9
2
1
3
5
6
9
1
2
0
-1

alpha beta-pruning



→ 0 gaperlu lg dicek karena kalau di > 2, ga bakal dipilih si min tapi kalo < 2, ga bakal dipilih si maks.

kalo misal dia > 6 ga bakal dipilih min.

< 6, ga bakal dipilih maks.

α mengubah nilai maks.