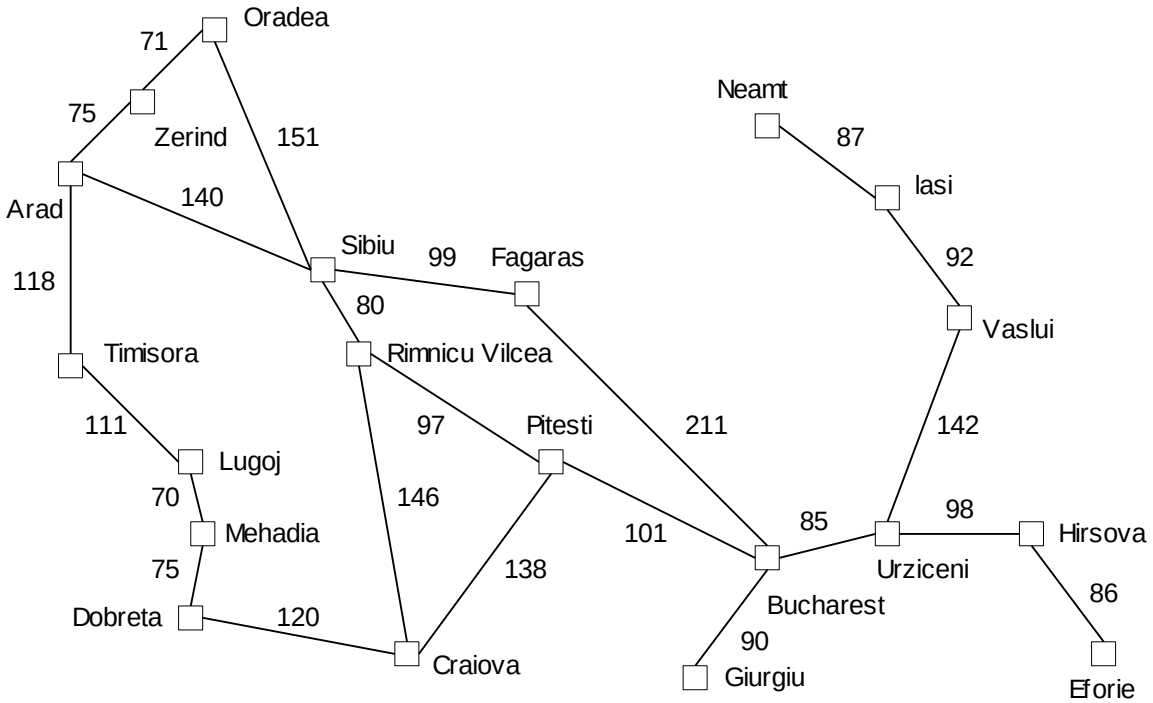
Metode Pelacakan/Searching

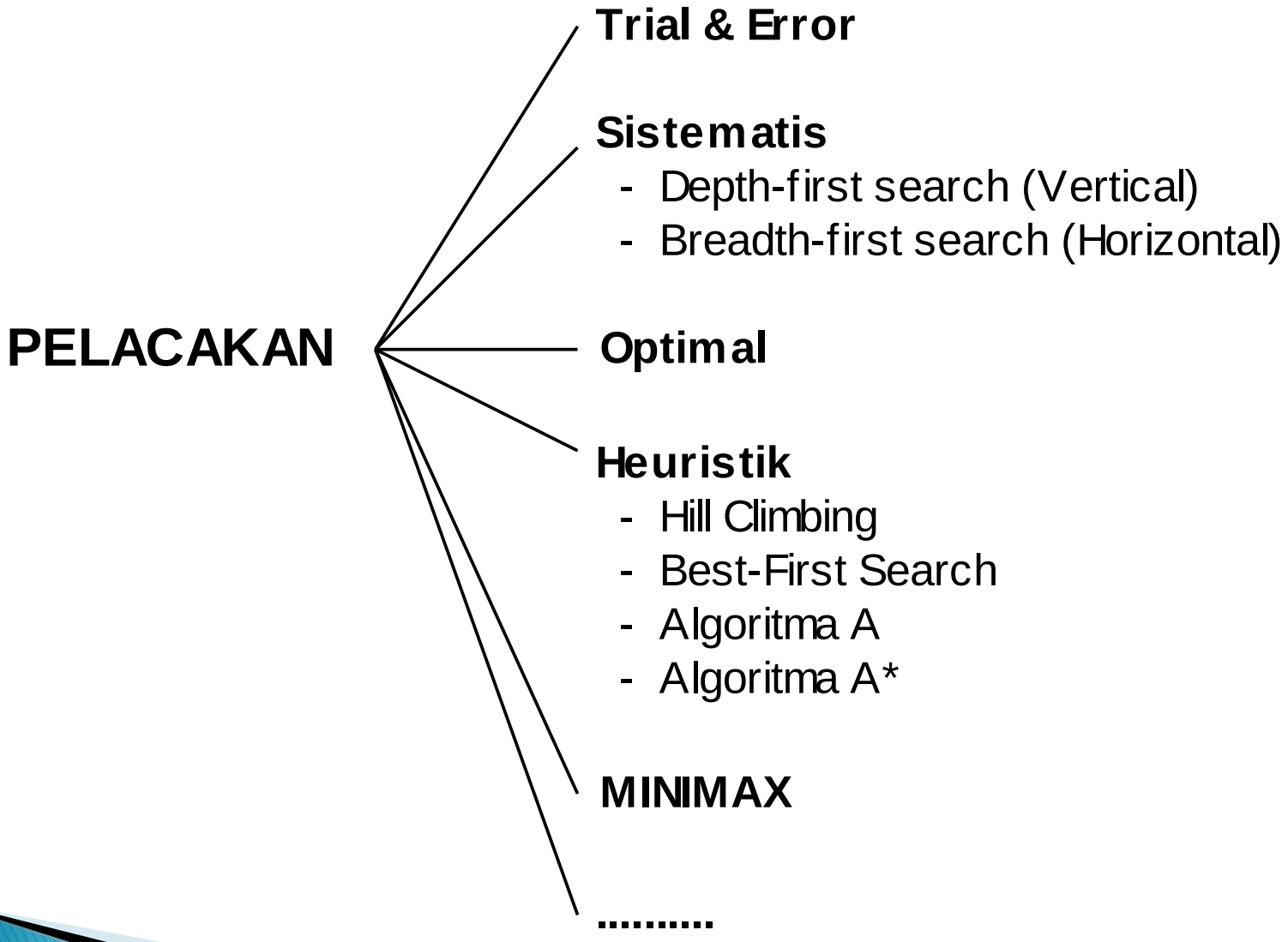
Kasus Pelacakan untuk Pemilihan rute terpendek



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	226
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Romania with step cost in km

Bagaimana Representasi Graph (start : Arad => tujuan:Bucharest)???


PELACAKAN
Trial & Error
Sistematis
- Depth-first search (Vertical)
- Breadth-first search (Horizontal)
Optimal
Heuristik
- Hill Climbing
- Best-First Search
- Algoritma A
- Algoritma A*
MINIMAX
.....

Yang perlu dipertimbangkan :

A. Arah pelacakan
B. Topologi proses pelacakan
C. Representasi simpul (Matrik, String, Array)
D. Pemilihan kaidah (rule) yang dapat diterapkan
E. Penggunaan fungsi heuristik

A. Arah Pelacakan

- Forward
  - dimulai dari initial state
  - matching dengan pola kiri kaidah $\rightarrow$ pola kanan membentuk simpul anak

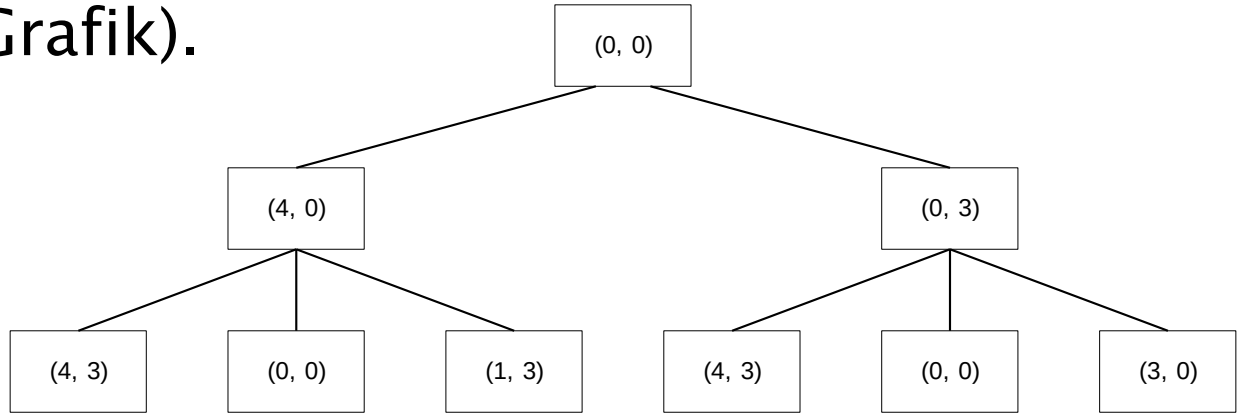
- Backward
  - dimulai dari goal state
  - matching dengan pola kiri kaidah $\rightarrow$ pola kiri membentuk simpul induk

- $\sum$ initial state > $\sum$ goal state $\rightarrow$ backward dan sebaliknya
- Faktor percabangan (jumlah rata-rata simpul yang dapat dicapai secara langsung dari sebuah simpul). Arah pelacakan menuju faktor percabangan yang kecil
- Proses penalaran

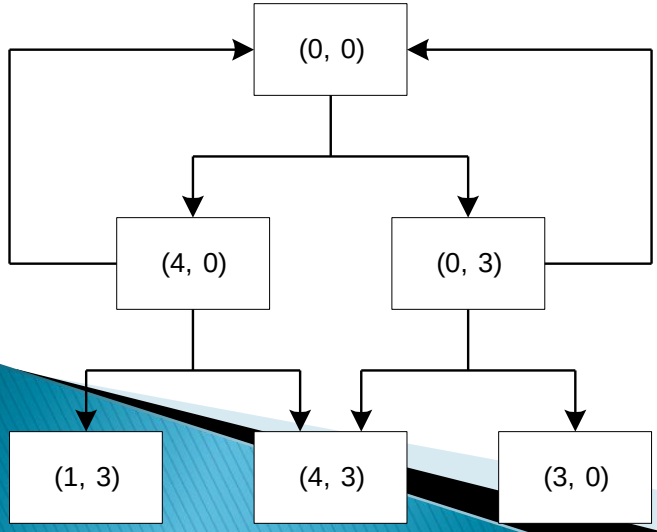
B. Topologi
Kasus 4-3 wadah

- Graph Tree (pohon),
- Graph Bebas (Grafik).

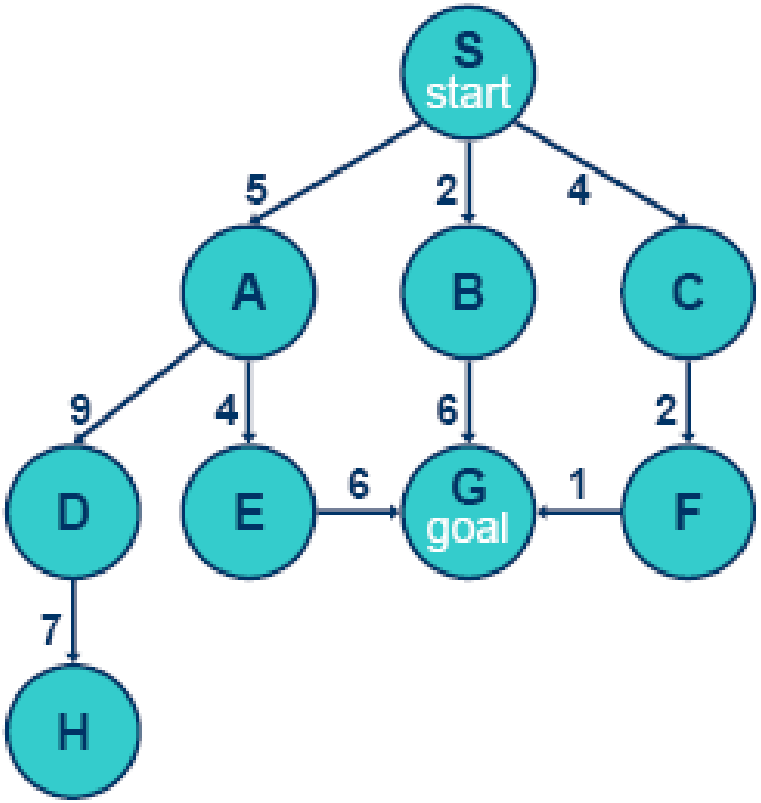
Graph Tree



Graph Bebas



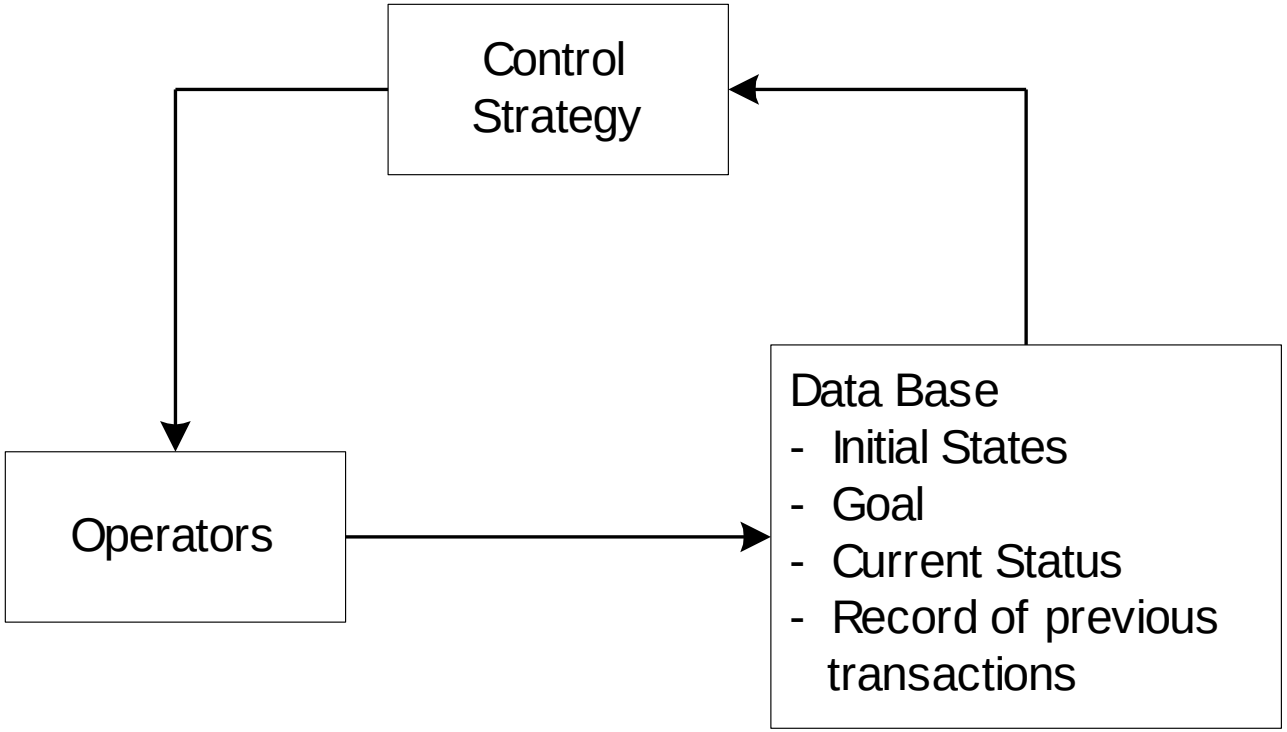
State Space



The size of a problem is usually described in terms of the number of possible states:

Tic-Tac-Toe: 3^9 states
Checkers: 10^{40} states
Rubik's Cube: 10^{19} states
Chess: 10^{120} states

Basic search process



Pelacakan(0): Trial & Error

TRIAL & ERROR

Metode paling sederhana

Prosedur Pelacakan

1. Ambil state sebagai keadaam awal
2. While state ≠ keadaan sasaran do
3. Begin
4. Pilih operator yang dapat diterapkan pada state, dan diset sebagai operator
5. State : = operator (state)
6. End

Penjelasan:

- Operator = fungsi untuk penentuan state berikutnya.
- Pada langkah 4, operator dipilih secara acak
- Pada langkah 5, operator yang dipilih diterapkan pada state membentuk state baru
- Stokastik, tidak menjamin dicapainya keadaan sasaran
- Tidak memperlihatkan karakteristik 'intelegensi'

Pelacakan(1): Depth-First Search

Pelacakan Depth-First Search

- Representasi: Diagram pohon atau grafik
- Simpul yang lebih dalam diperiksa terlebih dahulu → Penelusuran simpul-simpul pada suatu cabang sampai kedalaman yang ditentukan.

Prosedur

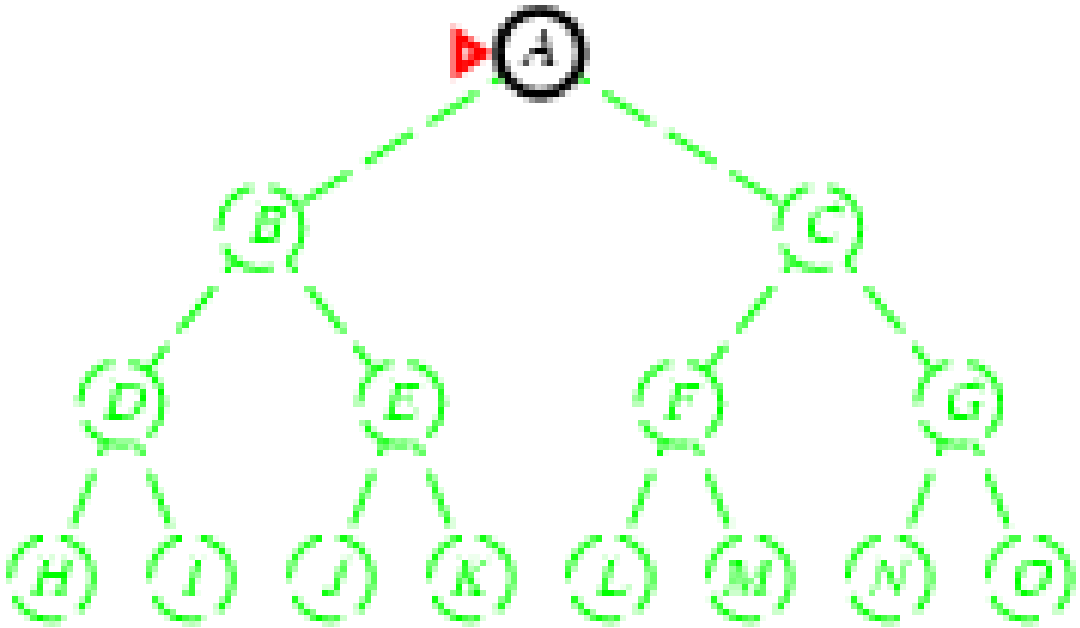
1. Berikan simpul awal pada daftar open
2. Loop: if open = kosong then exit (fail)
3. n : = first (open)
4. if goal (n) then exit (success)
5. Remove (n, open)
6. Ekspansikan n, berikan semua simpul anak pada kepala open dan bubuhkan pointer dari simpul anak ke n
7. Kembali ke Loop

Penjelasan:

- Pada langkah 3, elemen pertama daftar open diambil
- Ekspansi simpul n = pembangkitan simpul-simpul anak dari suatu simpul n

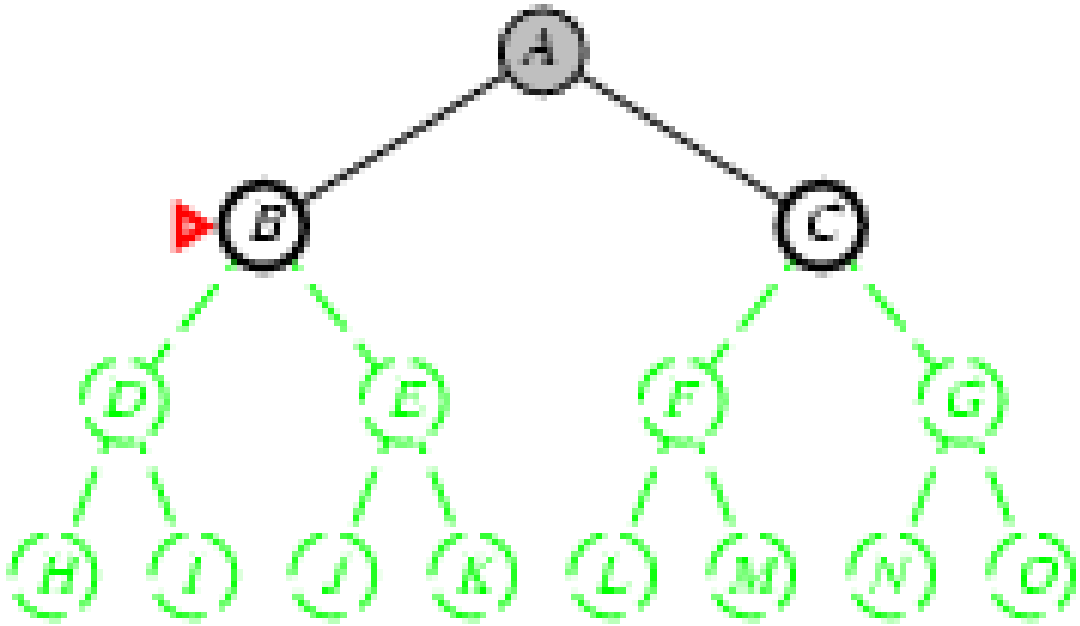
Simulasi Depth-First Search

Daftar Open: [A]



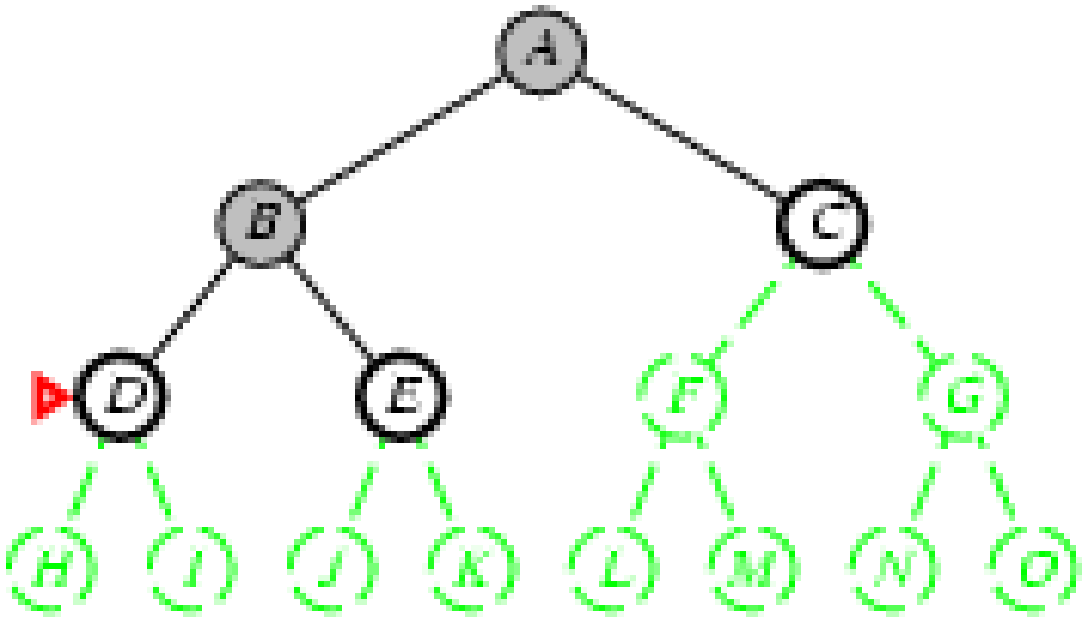
Simulasi Depth-First Search

Daftar Open: [BC]



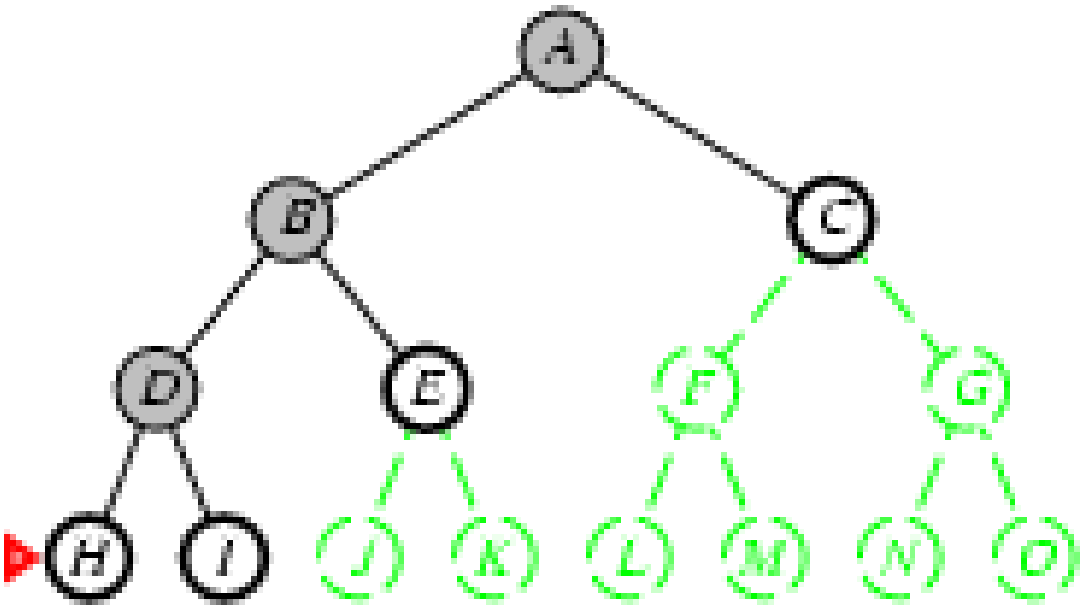
Simulasi Depth-First Search

Daftar Open: [DEC]



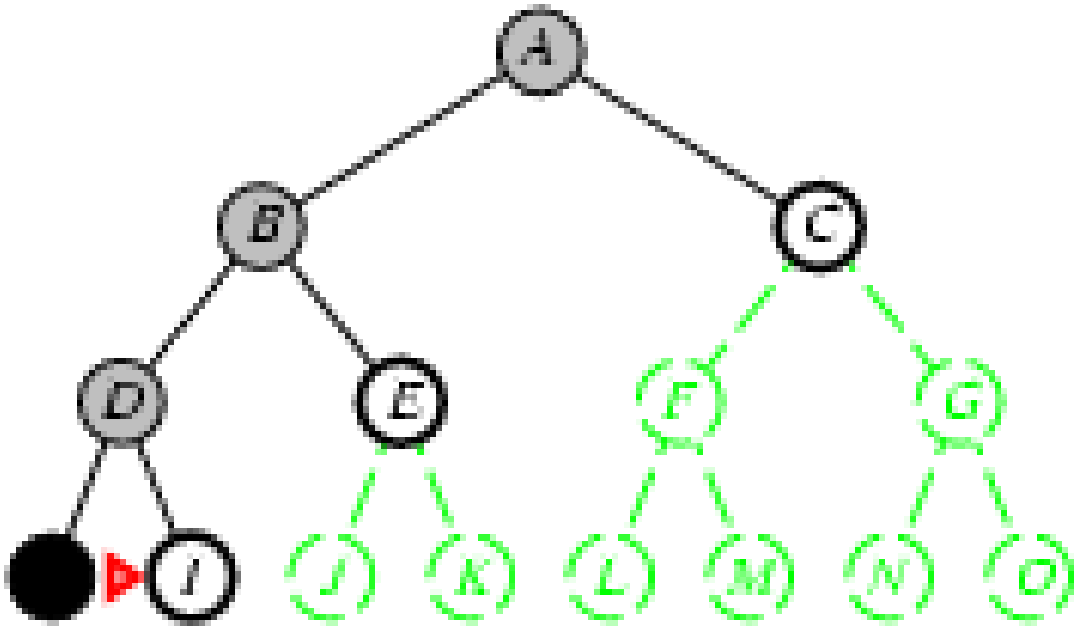
Simulasi Depth-First Search

Daftar Open: [HIEC]



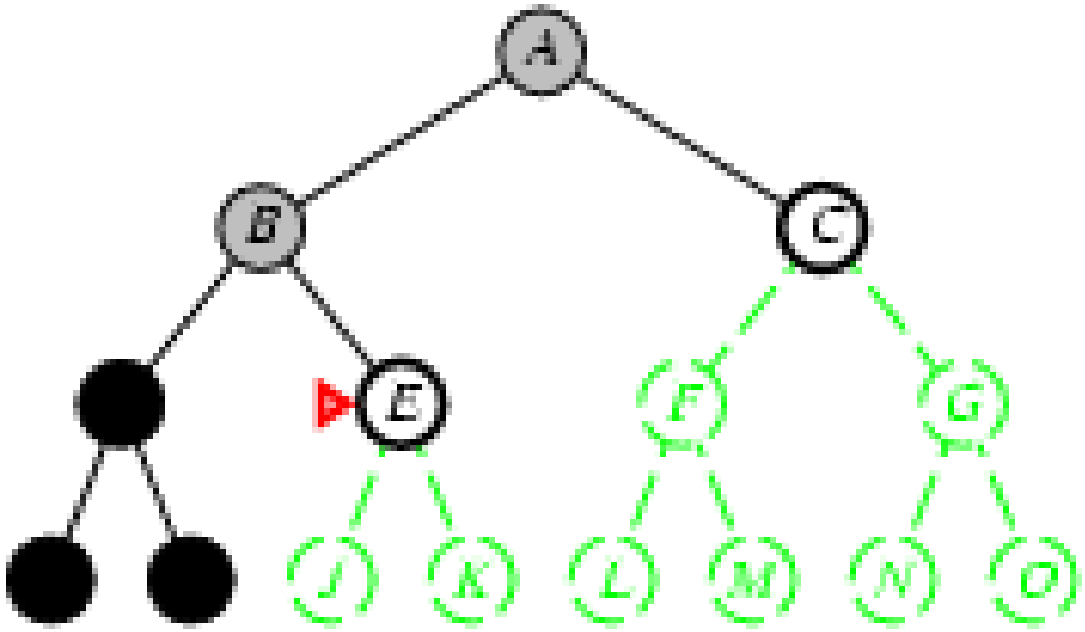
Simulasi Depth-First Search

Daftar Open: [IEC]



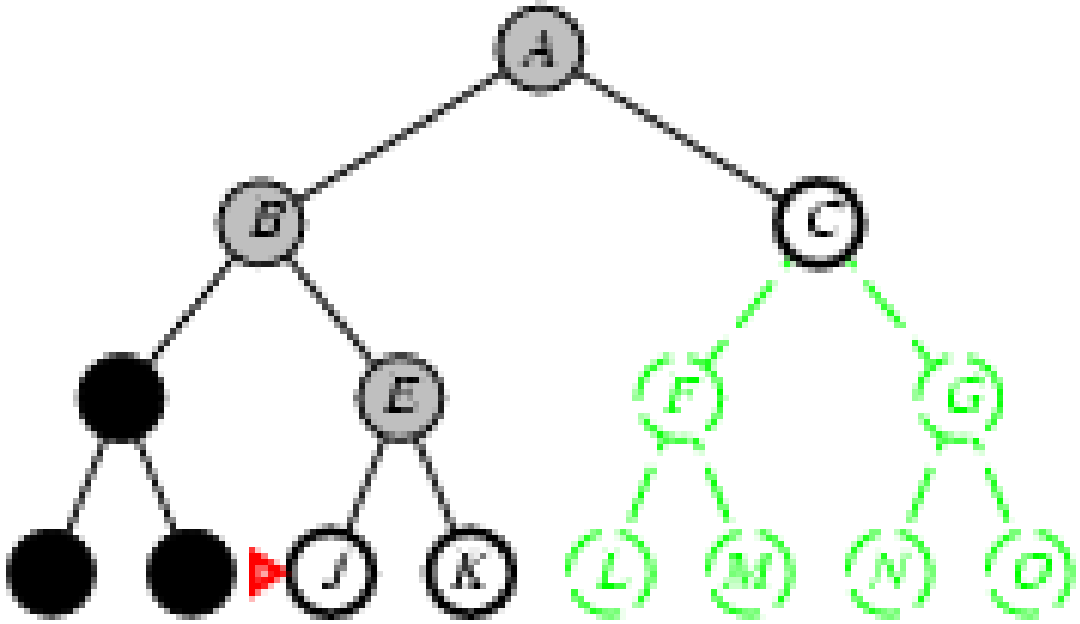
Simulasi Depth-First Search

Daftar Open: [EC]



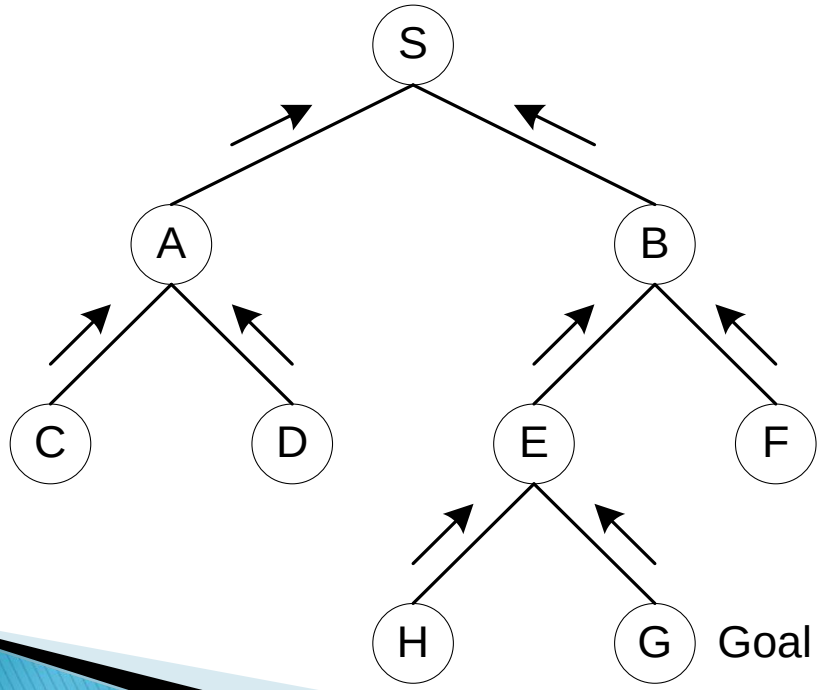
Simulasi Depth-First Search

Daftar Open: [JKC]



Key Point DFS

- Vertikal (Eksploitasi dulu anak – anaknya)
- Daftar Open
- Pointer bapak-anak → pencatatan goal (PR ? ☺)



Urutan pelacakan:
S, A, C, D, B, E, H, G

Pelacakan(2): Breadth-First Search

Pelacakan Breadth-First Search

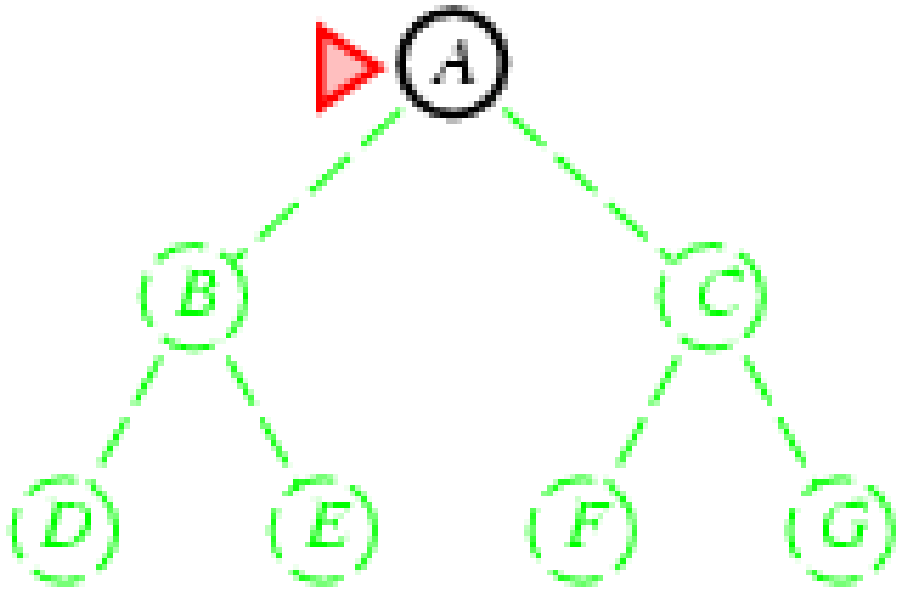
- Bersifat horizontal
- Evaluasi dilakukan terhadap simpul-simpul pada suatu level sebelum dilanjutkan pada level berikutnya

Prosedur

1. Berikan simpul awal pada open
2. Loop: if open = kosong then exit (fail)
3. n : = first (open)
4. if goal (n) then exit (success)
5. Remove (n, open)
6. Add (n, closed)
7. Ekspansikan n. Berikan pada ekonr open semua simpul anak yang belum muncul pada open atau closed dan bubuhkan pointer ke n.
8. Kembali ke Loop

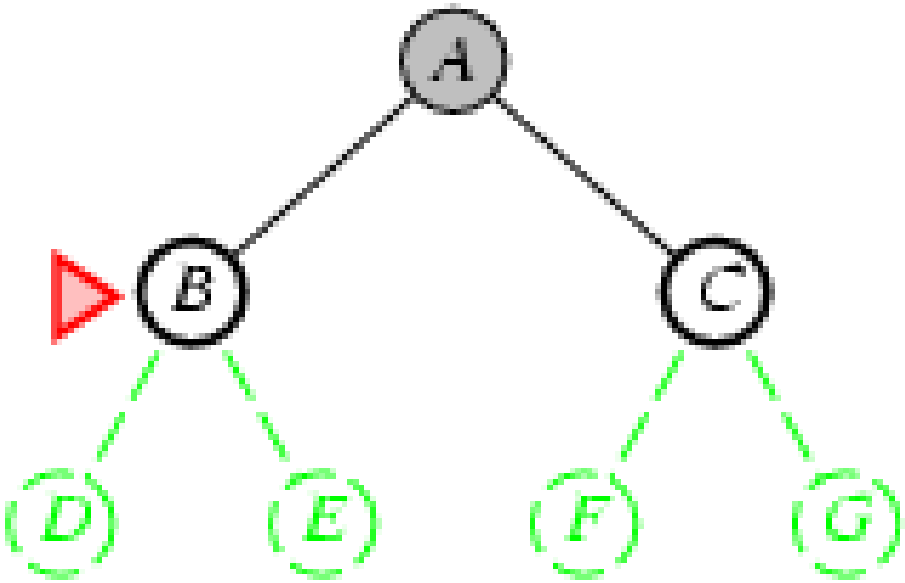
Simulasi Breadth-First Search

Daftar Open:[A]
Daftar Closed:[]



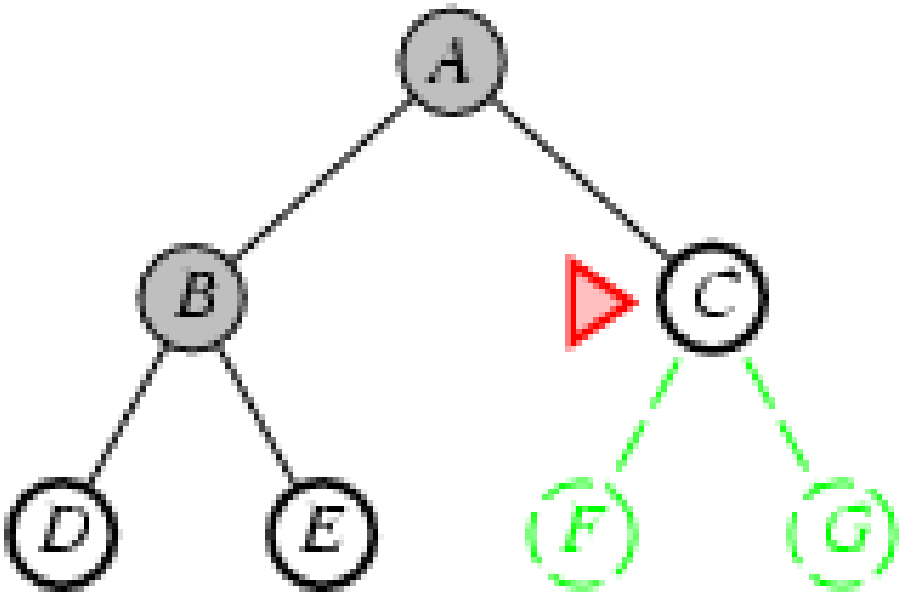
Simulasi Breadth-First Search

Daftar Open:[BC]
Daftar Closed:[A]



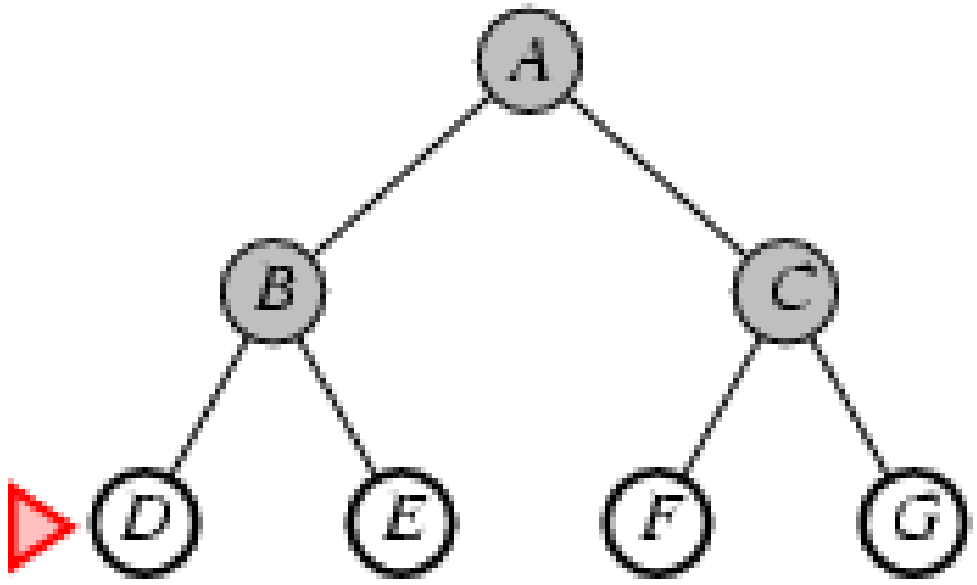
Simulasi Breadth-First Search

Daftar Open:[CDE]
Daftar Closed:[AB]



Simulasi Breadth-First Search

Daftar Open:[DEFG]
Daftar Closed:[ABC]



Key point

- Horizontal (Eksplorasi dulu sepupu-sepupunya),
- Daftar open dan closed.
- Pointer untuk trace back.

Breadth-First Search

Depth	Nodes	Time	Memory
0	1	1 millisecond	100 bytes
2	111	.1 seconds	11 kilobytes
4	11,111	11 seconds	1 megabyte
6	10^6	18 minutes	111 megabytes
8	10^8	31 hours	11 gigabytes
10	10^{10}	128 days	1 terabyte
12	10^{12}	35 years	111 terabytes
14	10^{14}	3500 years	11,111 terabytes

- Kebutuhan waktu dan memori BFS
- Branching factor b=10; (2) 1000 nodes/second; (3) 100 bytes/node

Pelacakan(3): dengan solusi optimal

- Mencari jejak dengan cost minimum,
- Cost diberikan oleh user.

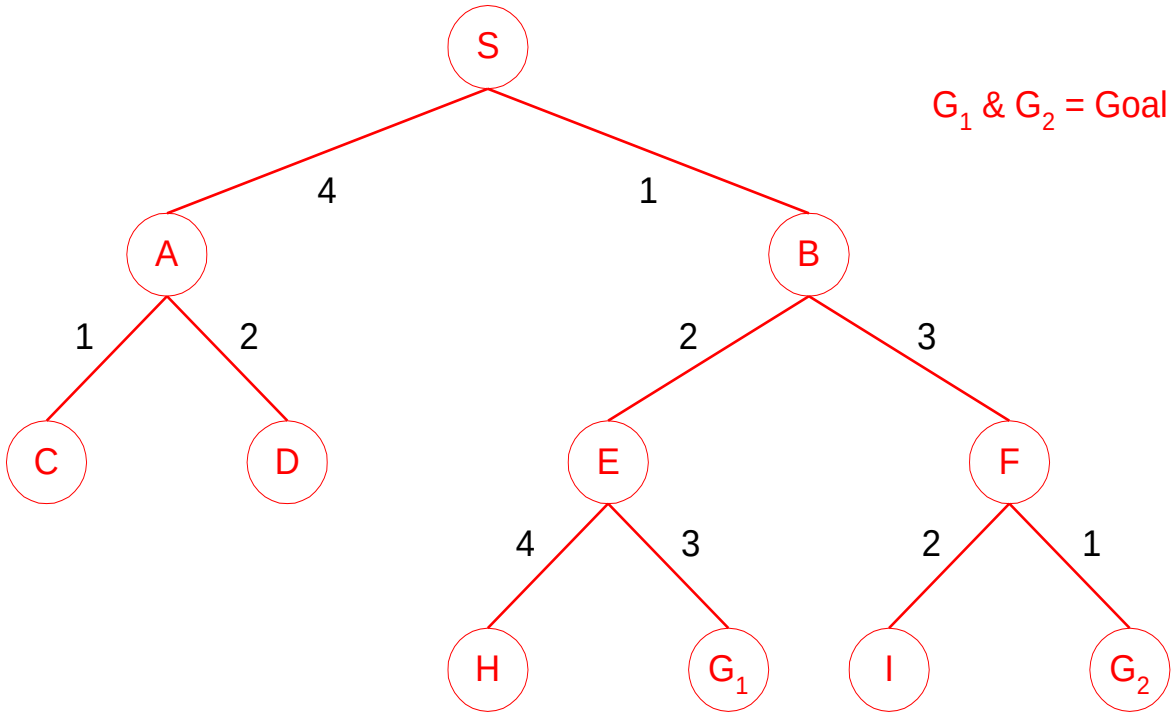
Prosedur Pelacakan

1. Berikan simpul awal S pada open, g'(s) = 0
2. Loop : if open = kosong then exit (fail)
3. n : = first (open)
4. if goal (n) then exit (success)
5. Remove (n, open)
6. Ekspansikan n, hitung $g'(n_i)$ untuk semua simpul anak n_i dan bubuhkan pointer dari n_i ke n. Berikan semua simpul anak pada open dan urutkan mulai biaya terendahnya
7. Kembali ke Loop

Penjelasan

- Pada langkah 6, jika fungsi biaya dari simpul n ke n_i didefinisikan sebagai $C(n, n_i)$, maka fungsi biaya dari simpul n_i adalah : $g'(n_i) = g'(n) + C(n, n_i)$

Pelacakan untuk memperoleh Solusi Optimal



Daftar Open : (S(0)) → (B(1)A(4)) → (E(3)A(4)F(4)) → (A(4)F(4)G_1(6)H(7)) → (F(4)C(5)G_1(6)D(6)H(7)) → (G2(5)C(5)G_1(6)D(6)I(6)H(7))

Pelacakan dengan informasi *Heuristik*

- Heuristik adalah kriteria, metode, atau prinsip-prinsip untuk menentukan pilihan dari sejumlah alternatif mencapai sasaran dengan efektif
- Mekanisme Backtracking = kembali ke state sebelumnya jika suatu solusi gagal diperoleh
- Heuristik dipergunakan untuk mempersempit ruang pelacakan.

Pelacakan(4): Hill Climbing

Hill Climbing

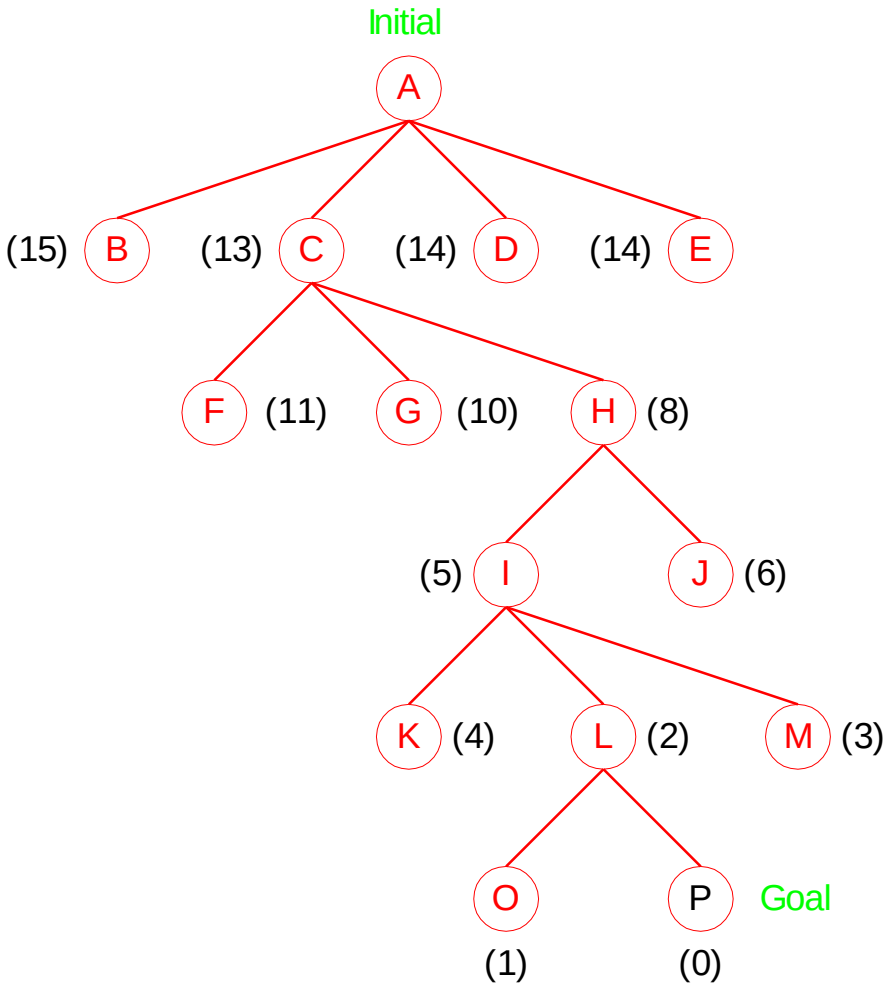
- Memilih simpul-simpul suatu cabang yang diperkirakan lebih dekat terhadap sasaran (nilai heuristik terkecil)
- Mirip "depth-first search"

Prosedur:

1. Ambil *n* sebagai simpul awal
2. Loop: if goal(*n*) then exit(success)
3. Ekspansikan n, hitung $\hat{h}(n_i)$ untuk semua simpul n_i dan ambil simpul dengan nilai heuristik terkecil next *n*
4. If $\hat{h}(n) < \hat{h}$(next *n*) then exit (fail)
5. *n* := next *n*
6. Kembali ke Loop

Hill climbing tidak dapat diterapkan pada persoalan yang memiliki puncak-puncak local.

Simulasi Hill Climbing



Pelacakan(5): Best-First Search

- Gabungan Breadth-first & Depth-first
- Pada setiap langkah dipilih simpul yang diperkirakan lebih dekat terhadap sasaran dari semua simpul yang dibangkitkan (tetapi belum diekspansikan)

Pseudo code untuk representasi Tree :

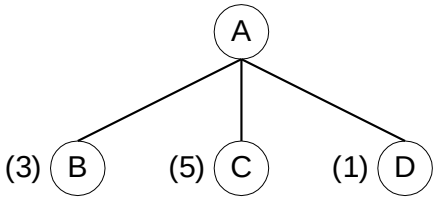
1. Berikan simpul awal n pada daftar open
2. If open = kosong then exit(fail)
3. N:= first(open)
4. Loop : if goal(n) then exit(success)
5. Remove (n,open)
6. Ekspansi n, masukan semua simpul anak dari n (ni) yang belum muncul pada open ke dalam daftar open dan bubuhkan pointer dari ni ke n, berikan h(ni) untuk setiap simpul pada ni. Ambil simpul yang memiliki nilai h(ni) yang terkecil dari daftar open sebagai next(n).
7. Kembali ke Loop

Catatan h(n) = nilai dari informasi heuristik untuk simpul n.

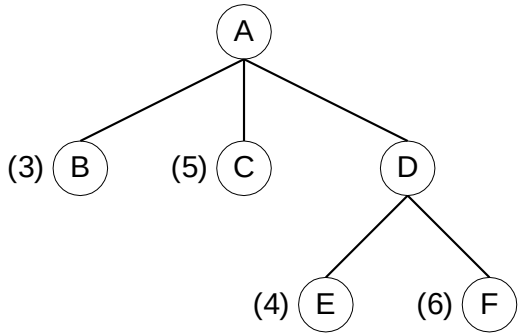
Simulasi Best-First Search

Langkah 1

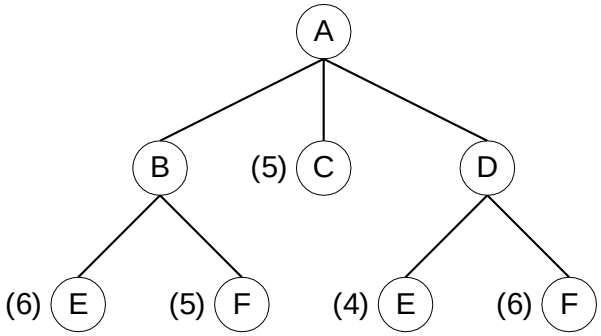
Langkah 2



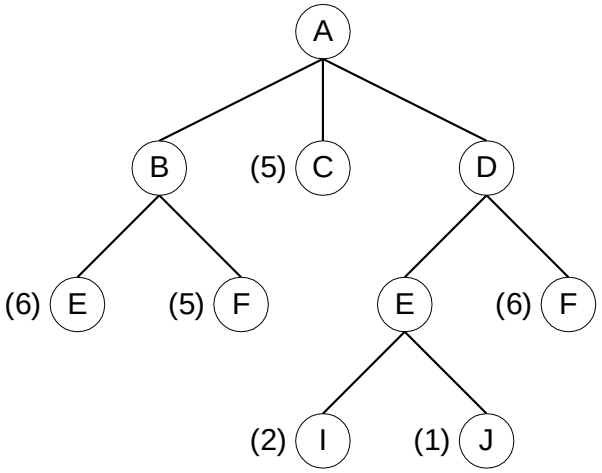
Langkah 3



Langkah 4



Langkah 5



Pelacakan(6): Algorithm A*/A

- Untuk tree

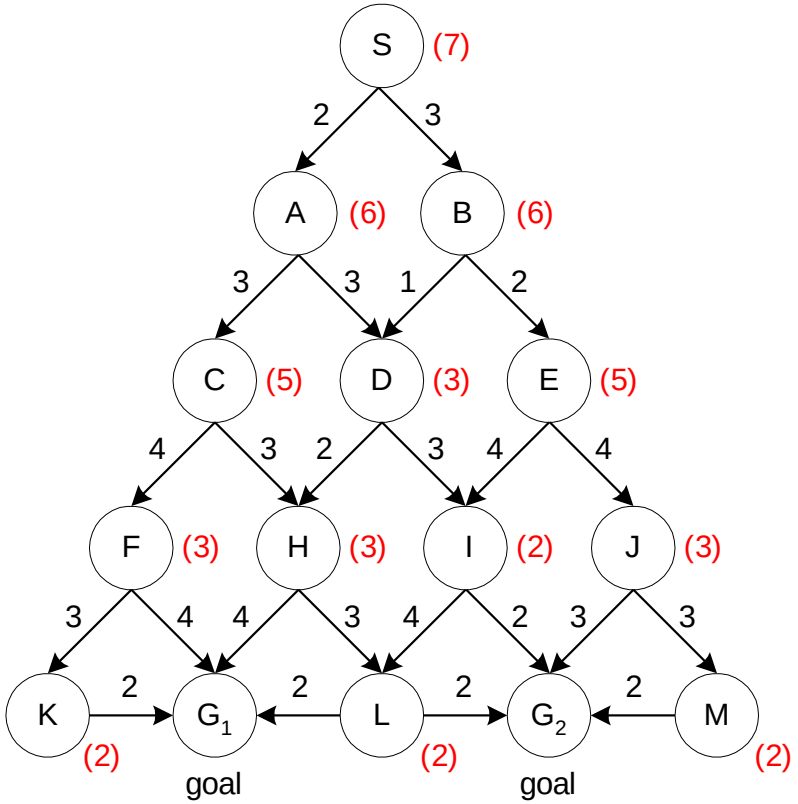
  1. Berikan simpul awal n pada daftar open
  2. if open = kosong then exit(fail).
  3. n:=first(open).
  4. Loop: if goal(n) then exit(success).
  5. Remove (n, open).
  6. Ekspansi n, Masukan simpul anak dari n (ni) ke dalam daftar open dan hitung f(ni) untuk tiap ni. Bubuhkan pointer dari ni ke n. Ambil simpul yang memiliki nilai f(ni) yang terkecil dari daftar open sebagai next(n).
  7. Kembali Loop

Catatan: f(n) = g(n) + h(n), dimana,

f(n) merupakan fungsi evaluasi dari simpul n,

g(n) merupakan kumulatif cost dari inisial awal ke simpul n(current state).

h(n) merupakan informasi heuristik dari simpul n



A-algorithm

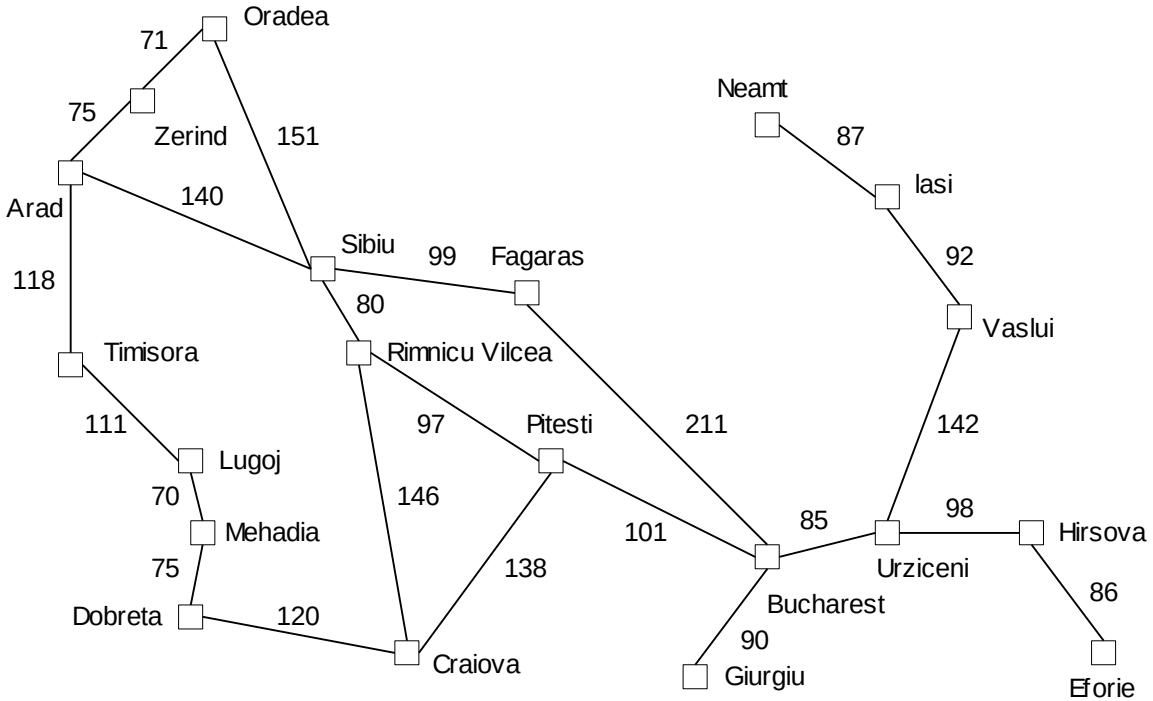
- Find optimum solution when the cost to the goal can be inferred
- Use knowledge about the problem

Open-list

$(S(7)) \rightarrow (A(8)B(9)) \rightarrow (D(8)B(9)C(10)) \rightarrow (B(9)C(10)H(10)I(10)) \rightarrow$
$(D(7)C(10)E(10)H(10)I(10) \rightarrow (H(9)I(9)C(10)E(10)) \rightarrow$
$(I(9)G_1(10)C(10)E(10)L(11)) \rightarrow (G_2(9)G_1(10)C(10)E(10)L(11))$

Solution : $S \rightarrow B \rightarrow \ldots \rightarrow I \rightarrow G_2$

Contoh Kasus



Romania with step cost in km

Heuristik

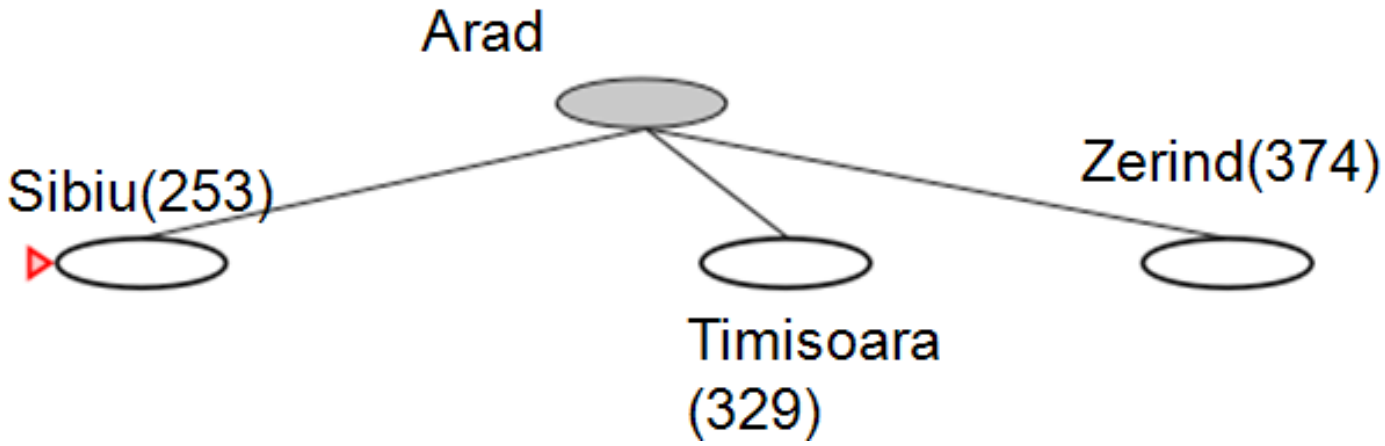
Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	226
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Tentukan rute dari arad ke Bucharest ???
Dgn Greedy search/BFS dan Alg. A*

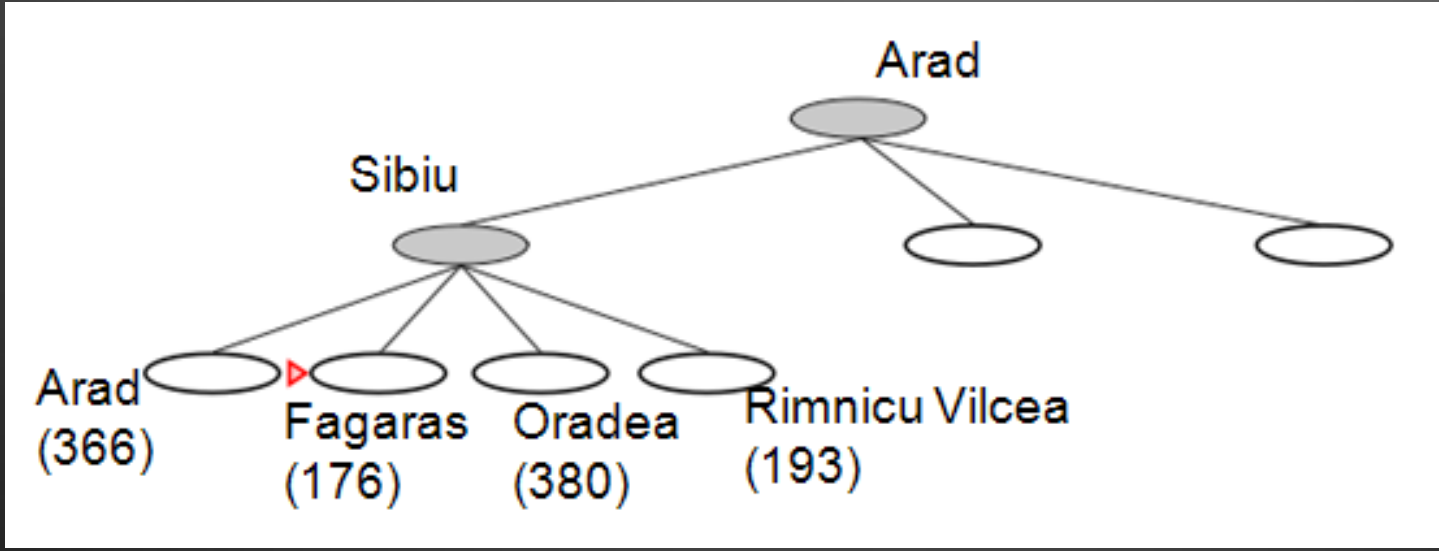
Greedy Search (Best First Search)

Arad (366)

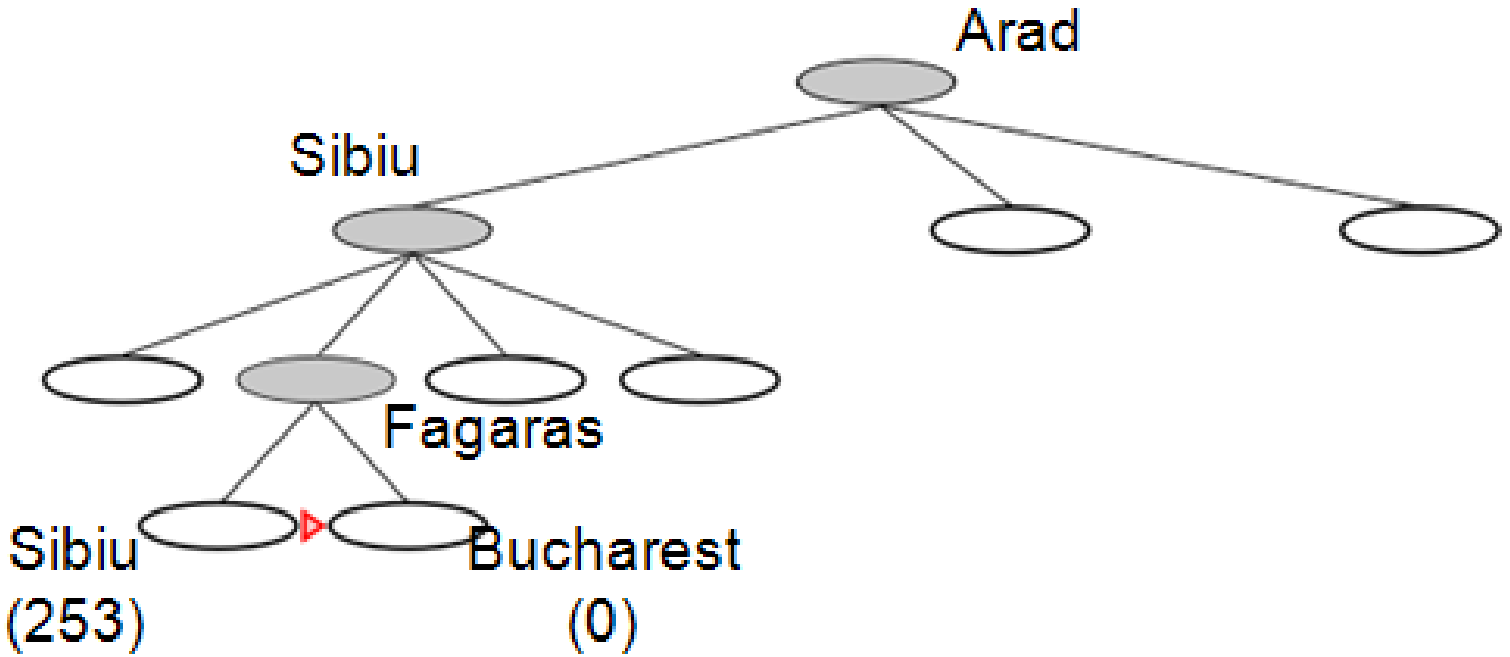
Greedy Search (Best First Search)



Greedy Search (Best First Search)



Greedy Search (Best First Search)



Goal tercapai:
Arad → Sibiu → Fagaras → Bucharest

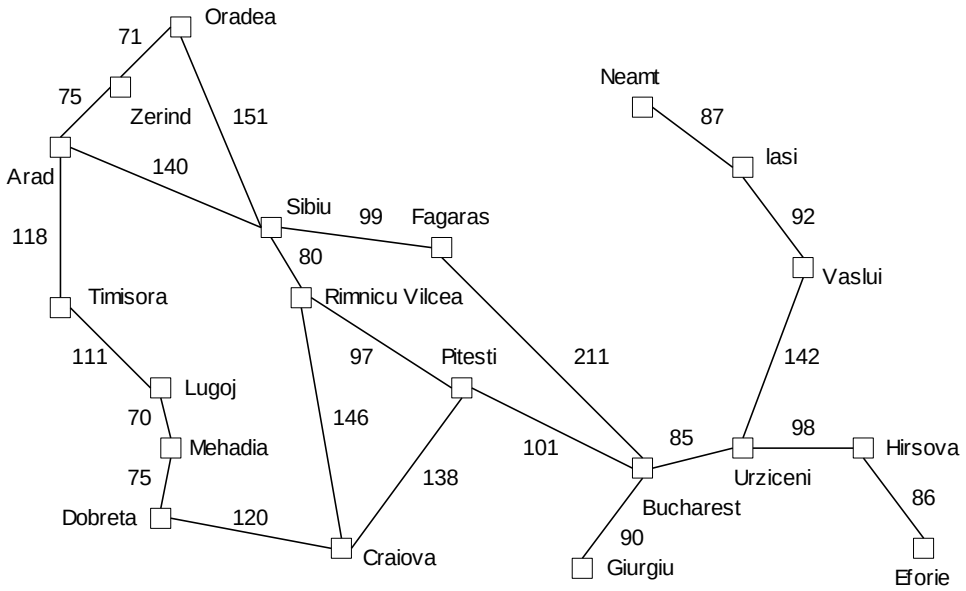
Contoh kasus dengan A* search

Evaluation function $f(n)=g(n) + h(n)$

$g(n)$ the cost (so far) to reach the node.

$h(n)$ estimated cost to get from the node to the goal.

$f(n)$ estimated total cost of path through n to goal.



Romania with step cost in km

Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	226
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

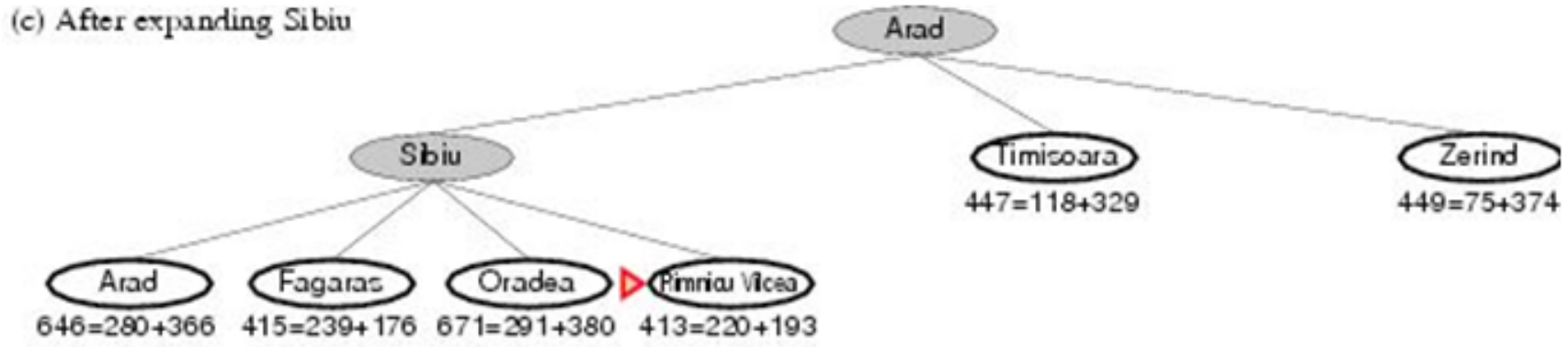
A* Search

(a) The initial state

A* Search

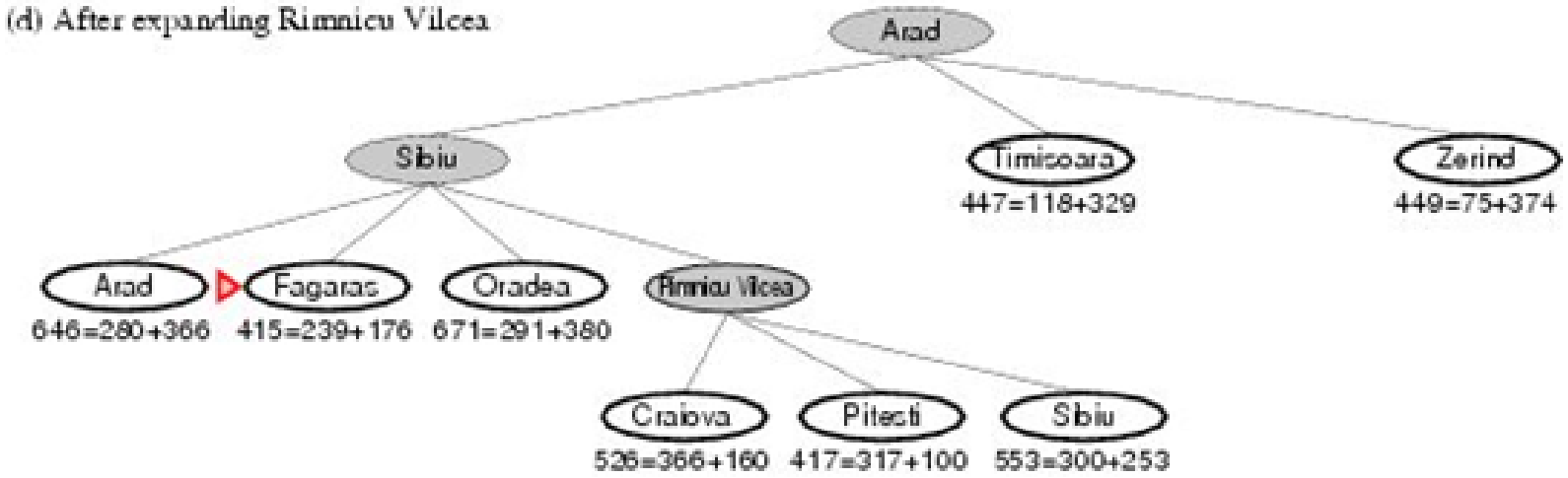
- Expand Arrad and determine *f(n)* for each node
  - **f(Sibiu)=c(Arad,Sibiu)+h(Sibiu)=140+253=393**
  - **f(Timisoara)=c(Arad,Timisoara)+h(Timisoara)=118+329=447**
  - **f(Zerind)=c(Arad,Zerind)+h(Zerind)=75+374=449**
- Best choice is Sibiu

A* Search



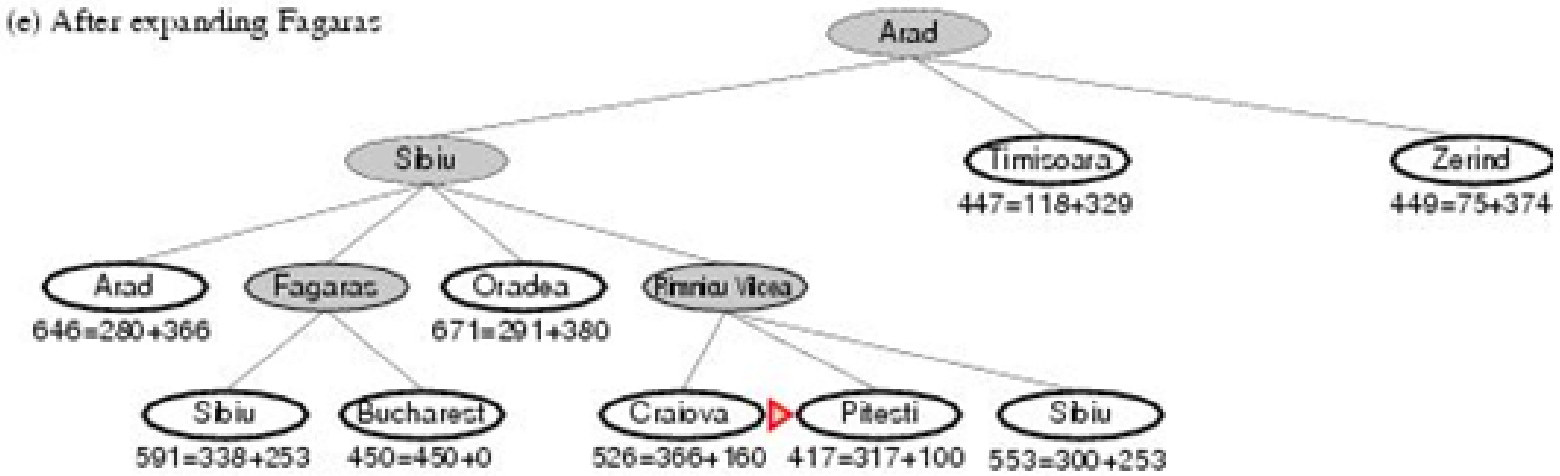
- Expand Sibiu and determine *f(n)* for each node
  - **f(Arad)=c(Sibiu,Arad)+h(Arad)=280+366=646**
  - **f(Fagaras)=c(Sibiu,Fagaras)+h(Fagaras)=239+179=415**
  - **f(Oradea)=c(Sibiu,Oradea)+h(Oradea)=291+380=671**
  - **f(Rimnicu Vilcea)=c(Sibiu,Rimnicu Vilcea)+ h(Rimnicu Vilcea)=220+192=413**
- Best choice is Rimnicu Vilcea

A* Search



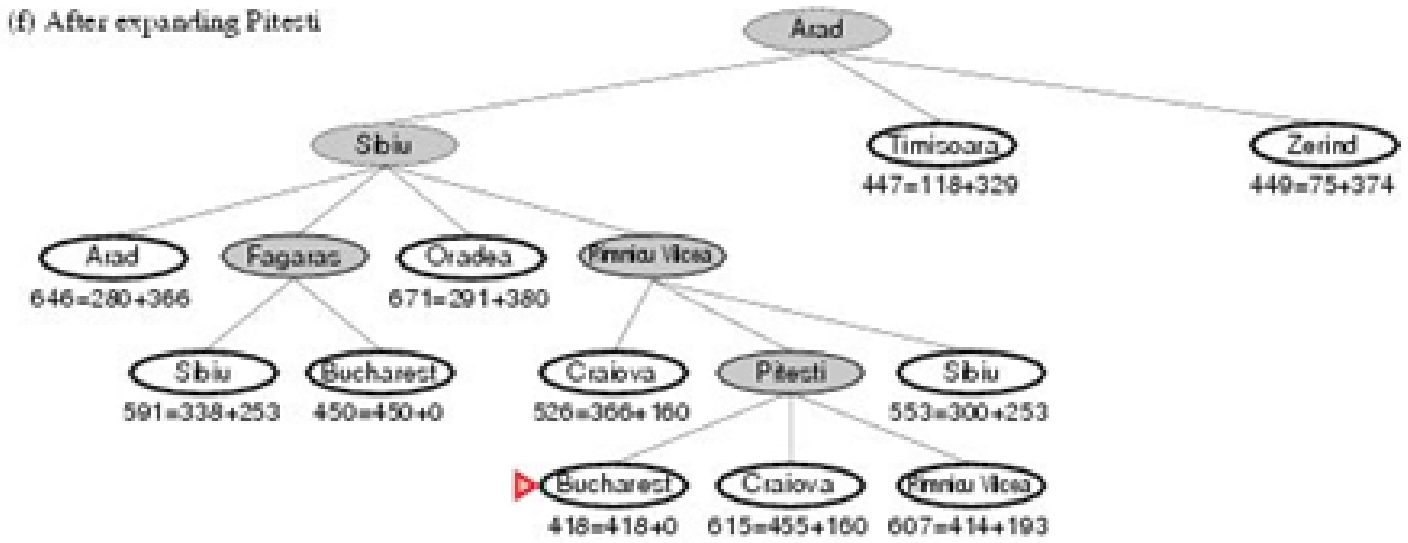
- Expand Rimnicu Vilcea and determine *f(n)* for each node
  - **f(Craiova)=c(Rimnicu Vilcea, Craiova)+h(Craiova)=360+160=526**
  - **f(Pitesti)=c(Rimnicu Vilcea, Pitesti)+h(Pitesti)=317+100=417**
  - **f(Sibiu)=c(Rimnicu Vilcea,Sibiu)+h(Sibiu)=300+253=553**
- Best choice is Fagaras

A* Search



- Expand Fagaras and determine *f(n)* for each node
  - **f(Sibiu)=c(Fagaras, Sibiu)+h(Sibiu)=338+253=591**
  - ***f(Bucharest)=c(Fagaras,Bucharest)+h(Bucharest)=450+0=450***
- Best choice is Pitesti !!!

A* Search



- Expand Pitesti and determine *f(n)* for each node
  - ***f(Bucharest)=c(Pitesti,Bucharest)+h(Bucharest)=418+0=418***
- Best choice is Bucharest !!!
  - **Optimal solution (only if *h(n)* is admissable)**
- Note values along optimal path !!

PR C2(2 Maret 2009)

Buat pelacakan untuk Bucharest ke arad (initial state: bucharest).

Metode Greedy dan A*.

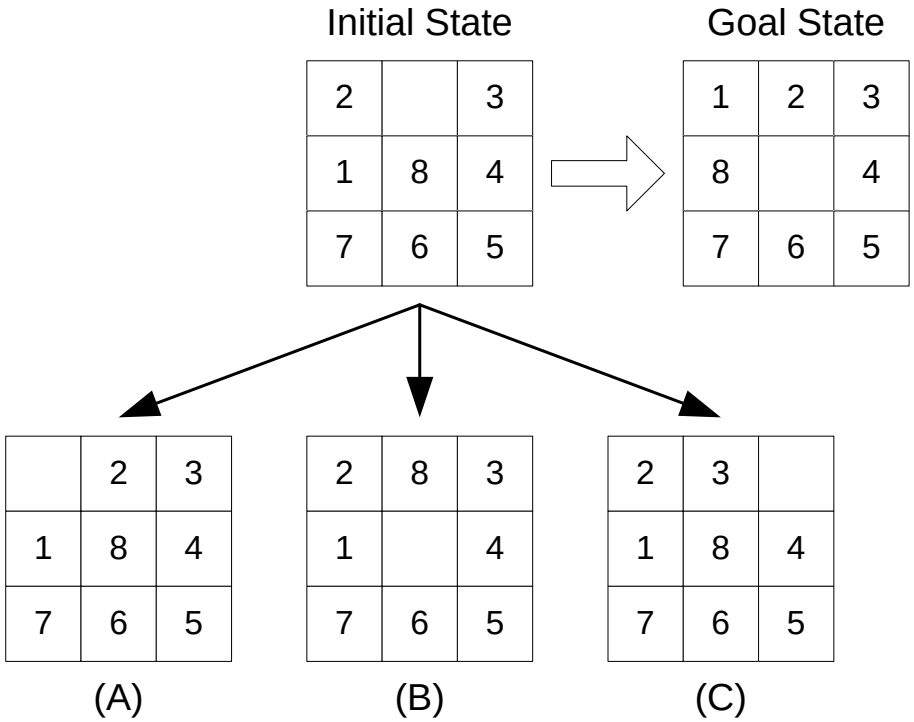
-grafik

-daftar open

-lakukan sesuai dengan algorithm

PR C1 (3 maret 2009)
Buat Pelacakan dari Urizeni ke Oradea
dengan BFS, A*
cat: Heuristik + 50
- Daftar open
- grafik
-sesuai algorithm

Kasus: Puzzle8



Heuristic

h_1 = number of mismatched tiles

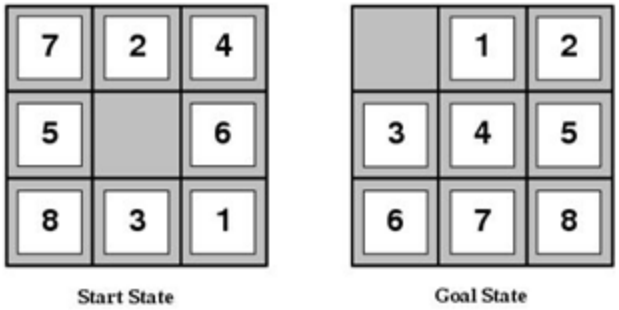
h_2 = sum of the (horizontal and vertical) disatancer of mismatched tiles (manhattan distance)

$h_1(A) = 2,\ h_1(B) = 3,\ h_1(C) = 4$

$h_2(A) = 2,\ h_2(B) = 4,\ h_2(C) = 4$

The state A is the one closest to the goal.

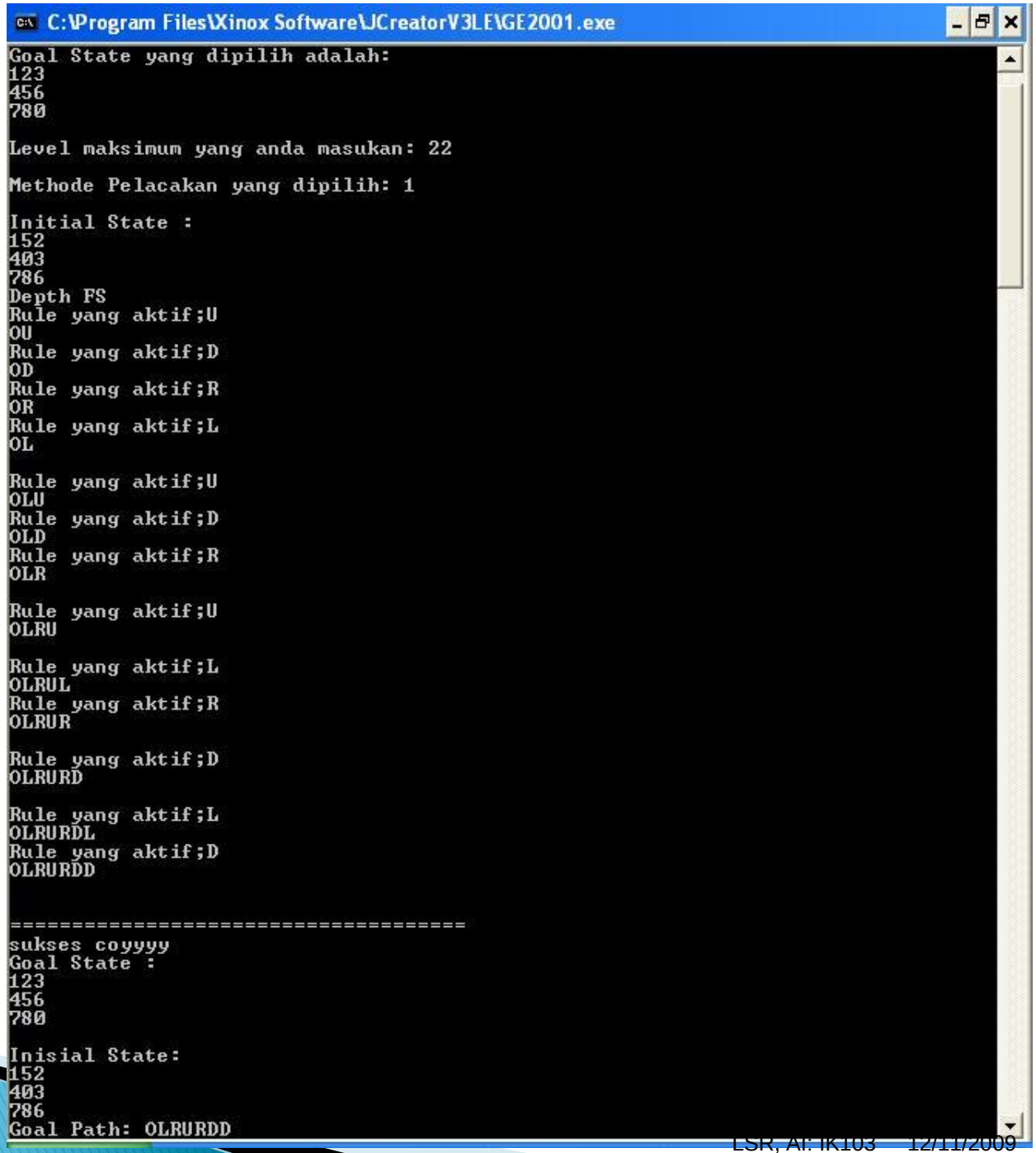
Contoh Heuristik Puzzle8



- E.g for the 8-puzzle knows two commonly used heuristics
- h_1 = the number of misplaced tiles
  - **$h_1(s)=8$**
- h_2 = the sum of the distances of the tiles from their goal positions (manhattan distance).
  - **$h_2(s)=3+1+2+2+2+3+3+2=18$**

Langkah Implementasi

- Representasi masalah (matrik 3x3, string).
- Definisikan rule (up, down, left, right).
- Cek rule yang aktif.
- Catat perubahan, rule yang aktif, parent untuk mencatat solusi.
- Hitung nilai heuristik.
- Implementasikan sesuai metode.


C:\Program Files\Xinox Software\JCreatorV3LE\GE2001.exe
Goal State yang dipilih adalah:
123
456
780
Level maksimum yang anda masukan: 22
Methode Pelacakan yang dipilih: 1
Initial State :
152
403
786
Depth FS
Rule yang aktif;U
OU
Rule yang aktif;D
OD
Rule yang aktif;R
OR
Rule yang aktif;L
OL
Rule yang aktif;U
OLU
Rule yang aktif;D
OLD
Rule yang aktif;R
OLR
Rule yang aktif;U
OLRU
Rule yang aktif;L
OLRUL
Rule yang aktif;R
OLRUR
Rule yang aktif;D
OLRURD
Rule yang aktif;L
OLRURDL
Rule yang aktif;D
OLRURDD
==========
sukses coyyyy
Goal State :
123
456
780
Inisial State:
152
403
786
Goal Path: OLRURDD

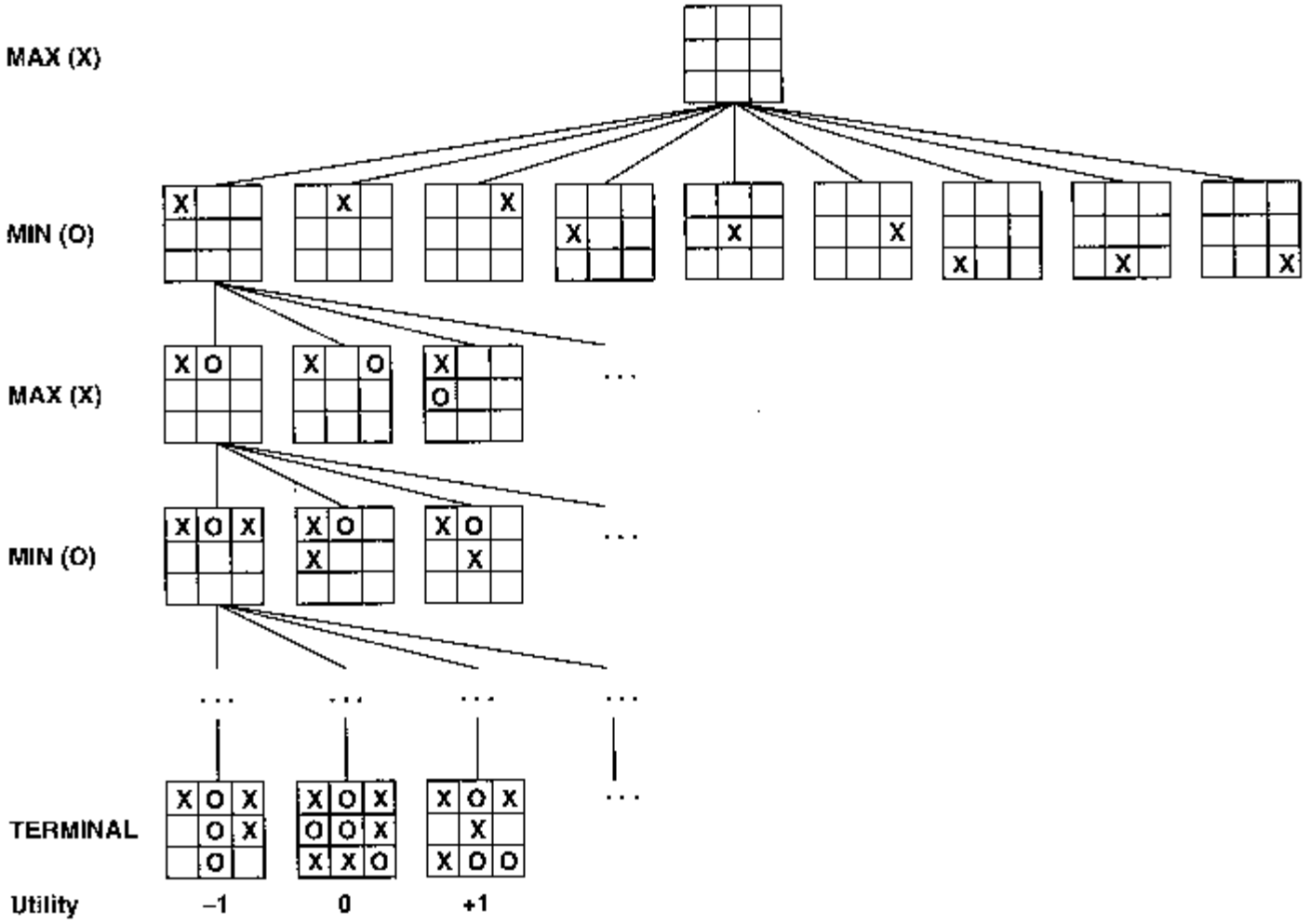
Pelacakan Pada Game

Minimax

Pengantar

- Ruang keadaan/repesentasi masalah -> graph tree,
- Tiap node merepresentasikan keadaan yang mungkin terjadi.
- **Persoalannya** : menentukan child state yang terbaik.
- Salah satu metodenya adalah minimax (untuk 2 atau lebih pemain)
- Minimax ditemukan pertama kali oleh von neumann Morgenstern tahun 1944, sebagai bagian dari game theory

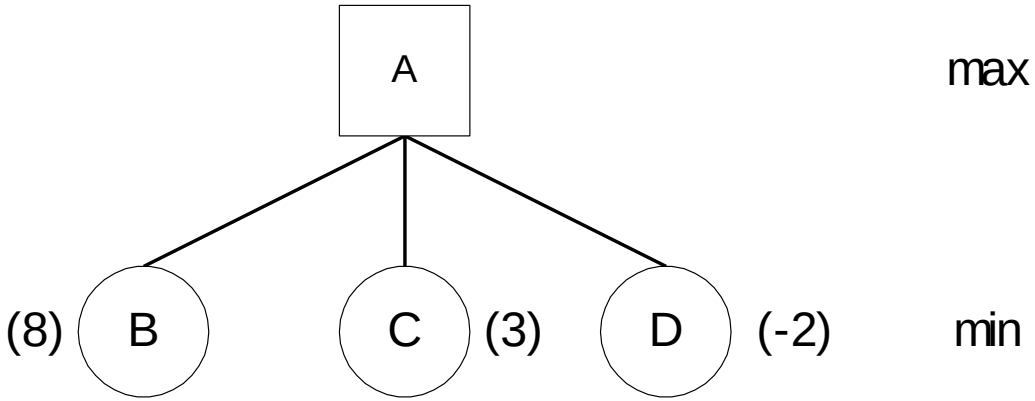
Pelacakan: Game



9!+1 = 362,880

Minimax

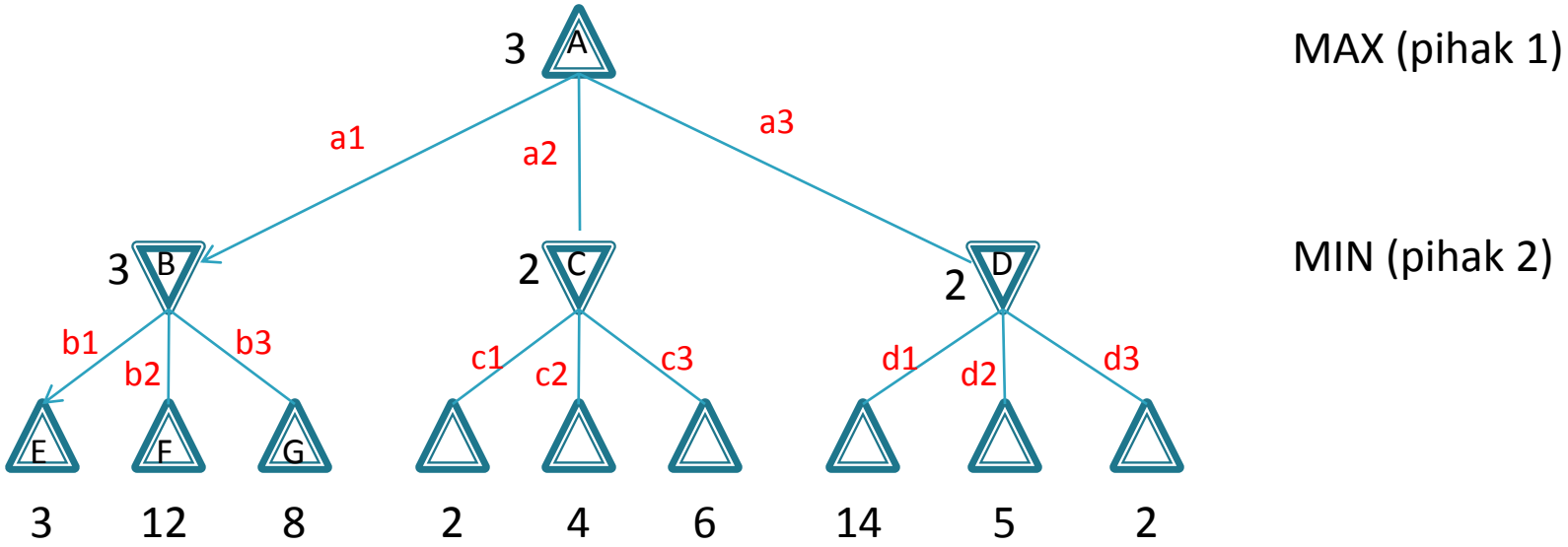
One-ply search



Static evaluation function : [-10, 10]

win for opponent win for us

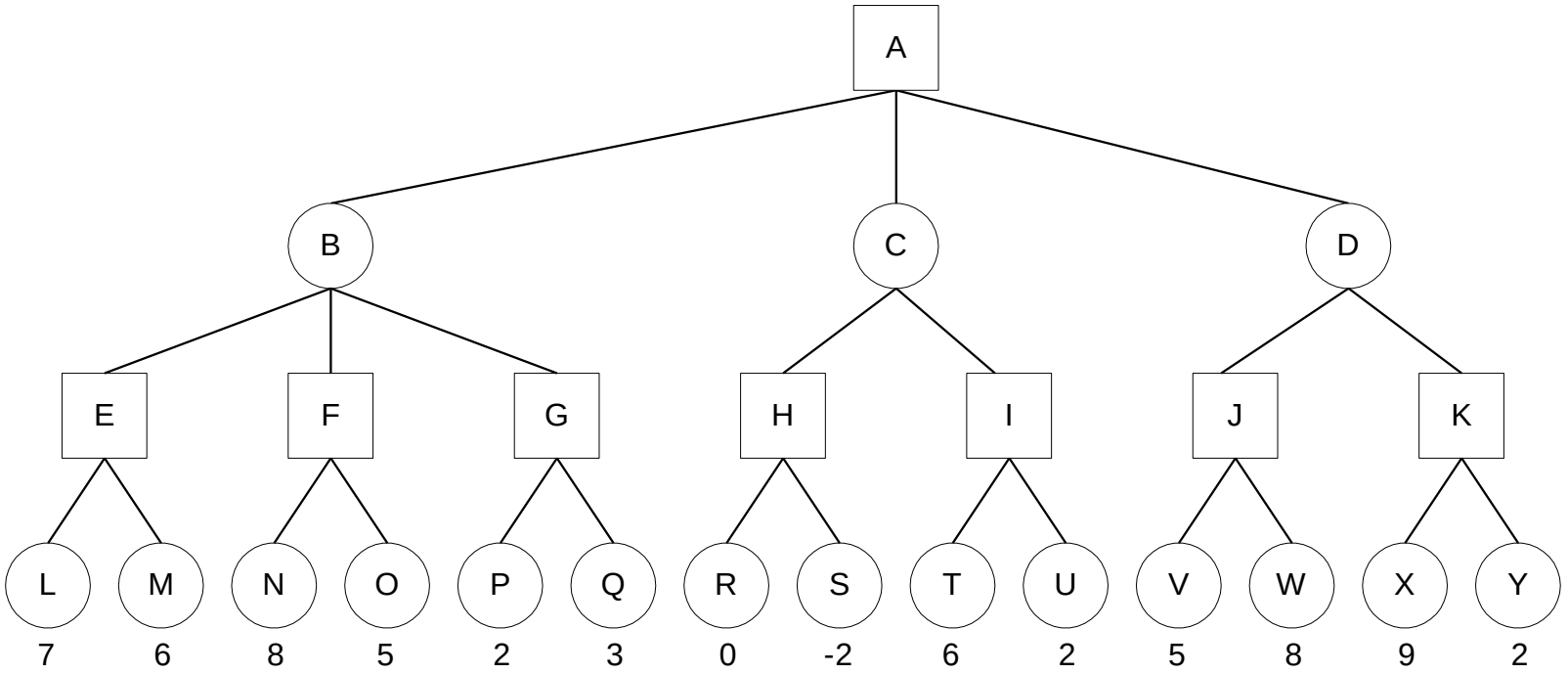
Ply = gerakan "lawan" atau "kita"



- Pihak 1: mendapatkan giliran melangkah (dinode A) dan sebagai pihak “max”.
- Pihak 1 akan memilih langkah “a1” karena memiliki bobot **tertinggi** diantara “B”, “C”, “D”.
- Sedangkan pihak 2 akan memilih langkah “b1” karena memiliki bobot **terendah** diantara “E”, “F”, “G”.

A = maximizing player

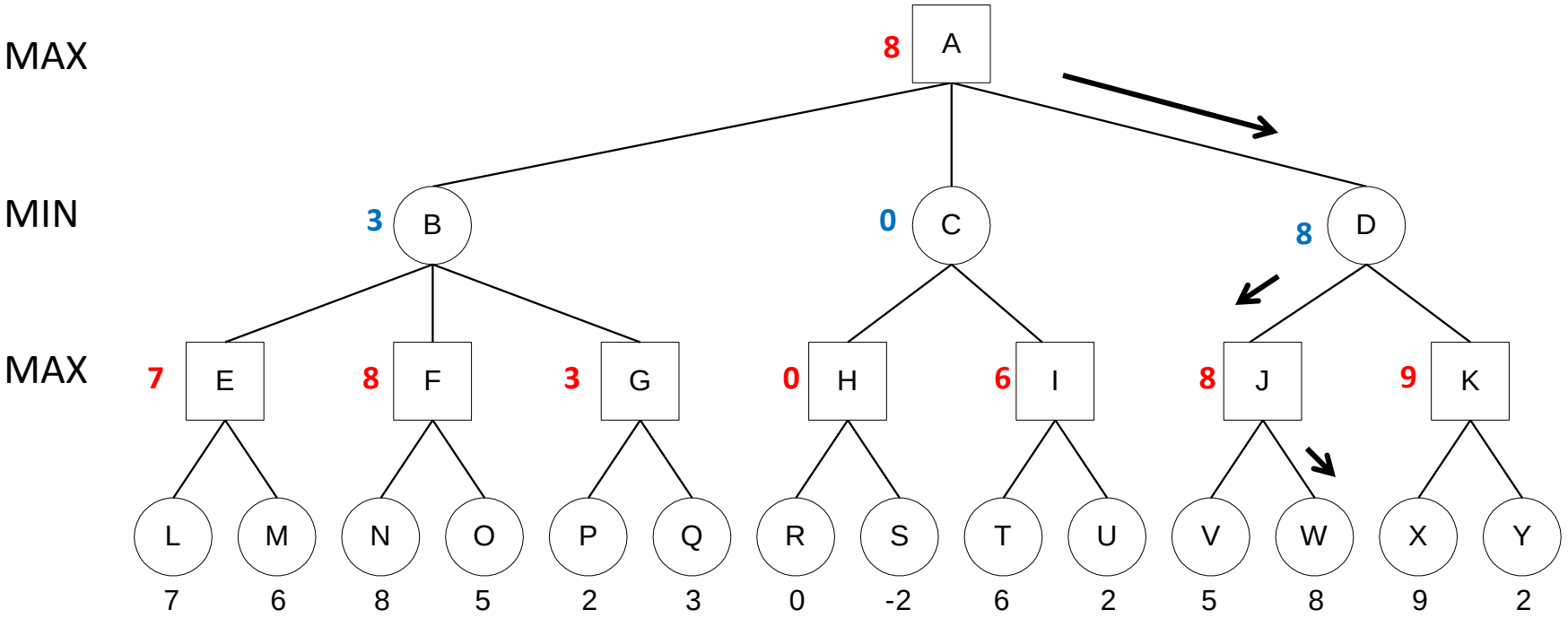
Question : What move should he choose ?



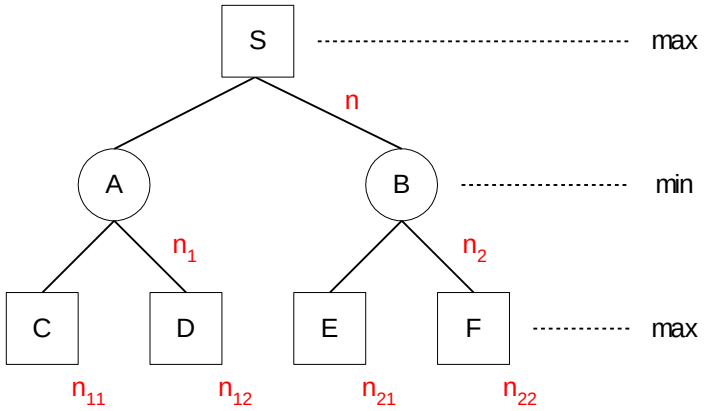
A = maximizing player

Question : What move should he choose ?

Solusi :



Minimax



n = Max node
n_i = child of max node, i = 1 – m
n_{ij} = child node of each max node, j = 1 – i_m
f = evaluation function

$$MAX : f(n_i) = \max_i f(n_i)$$

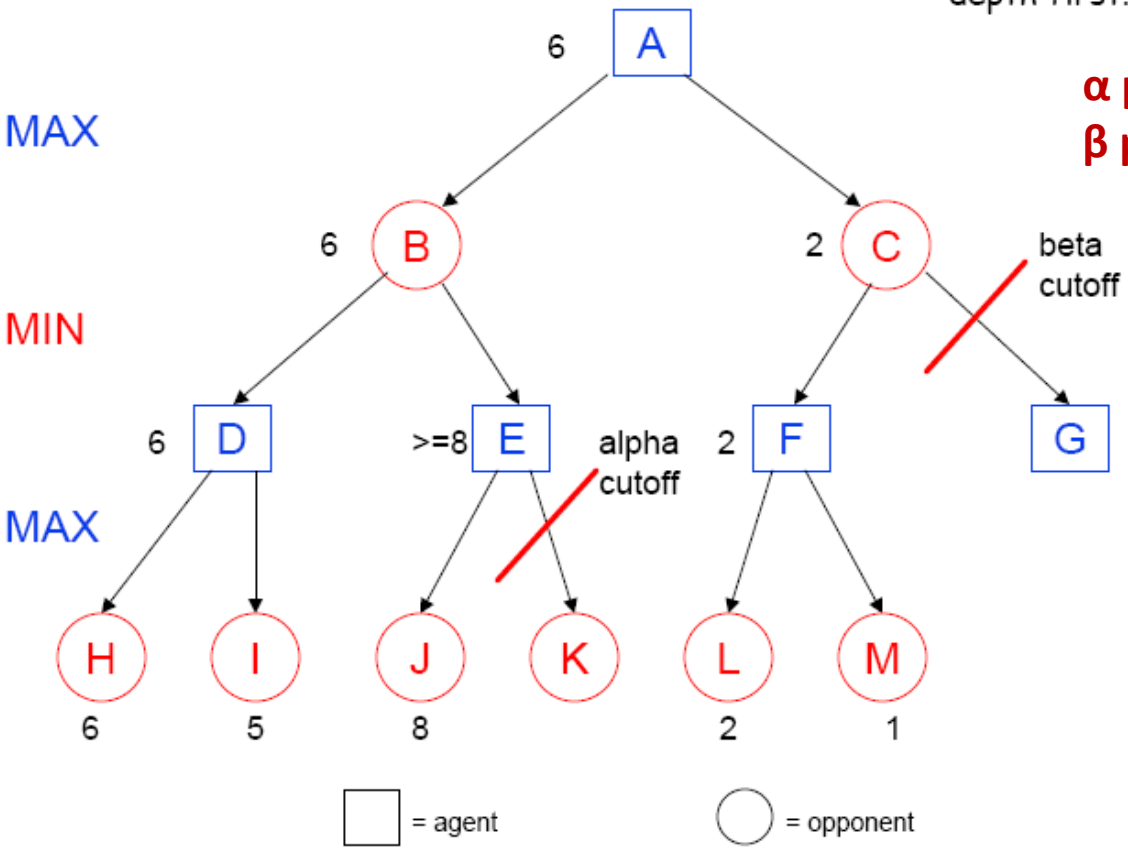
$$MIN : f(n_{ij}) = \min_j f(n_{ij})$$

$$MAX : f(n_{ij}) = \max_i \{ \min_j f(n_{ij}) \}$$

k = depth of the tree
MAX should select i_1 s.t:

$$f(n_{i_1 i_2 \ldots i_k}) = \max_{i_1} \min \ldots \{ f(n_{i_1 i_2 \ldots i_k}) \}$$

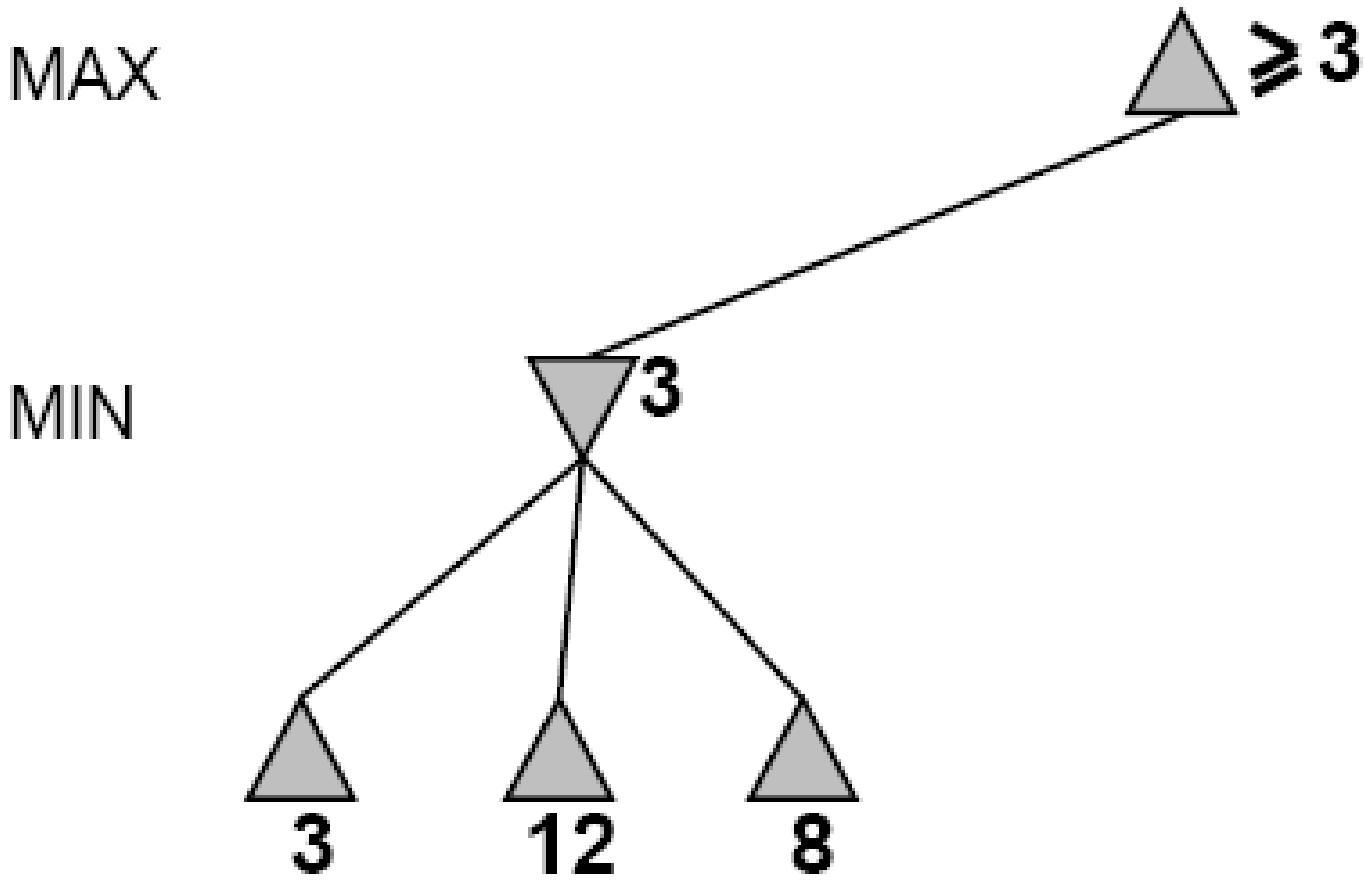
Alpha-beta Pruning



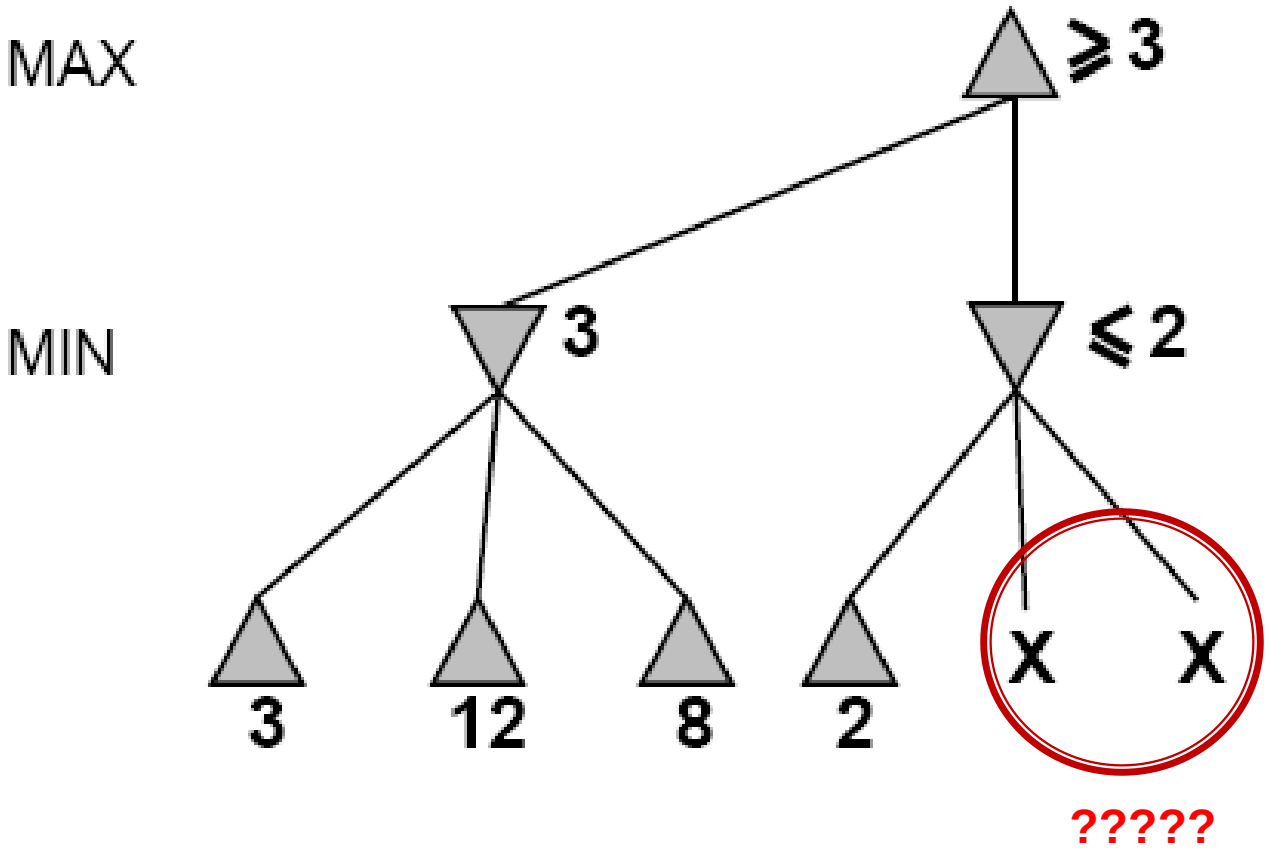
α pruning = pemotongan 1 level,
β pruning = pemotongan > 1 level

- **Penghitungan bobot dimulai dari kiri ke kanan, bawah ke atas.**
- **α cutoff krn sudah jelas node "E" dengan bobot 8 tidak dipilih oleh MIN B,**
- **β cutoff krn sudah jelas node C dengan bobot 2 akan terpilih oleh MAX A.**

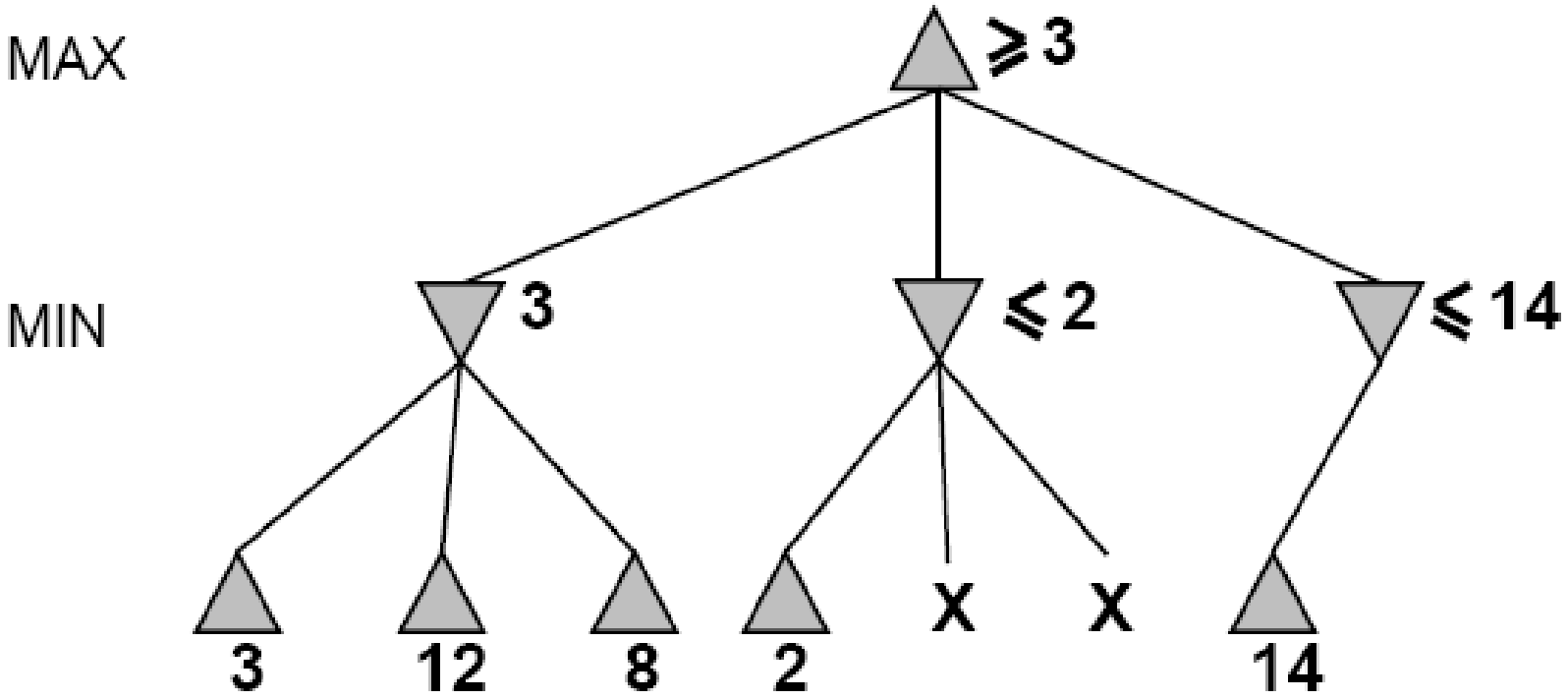
Contoh α/β pruning



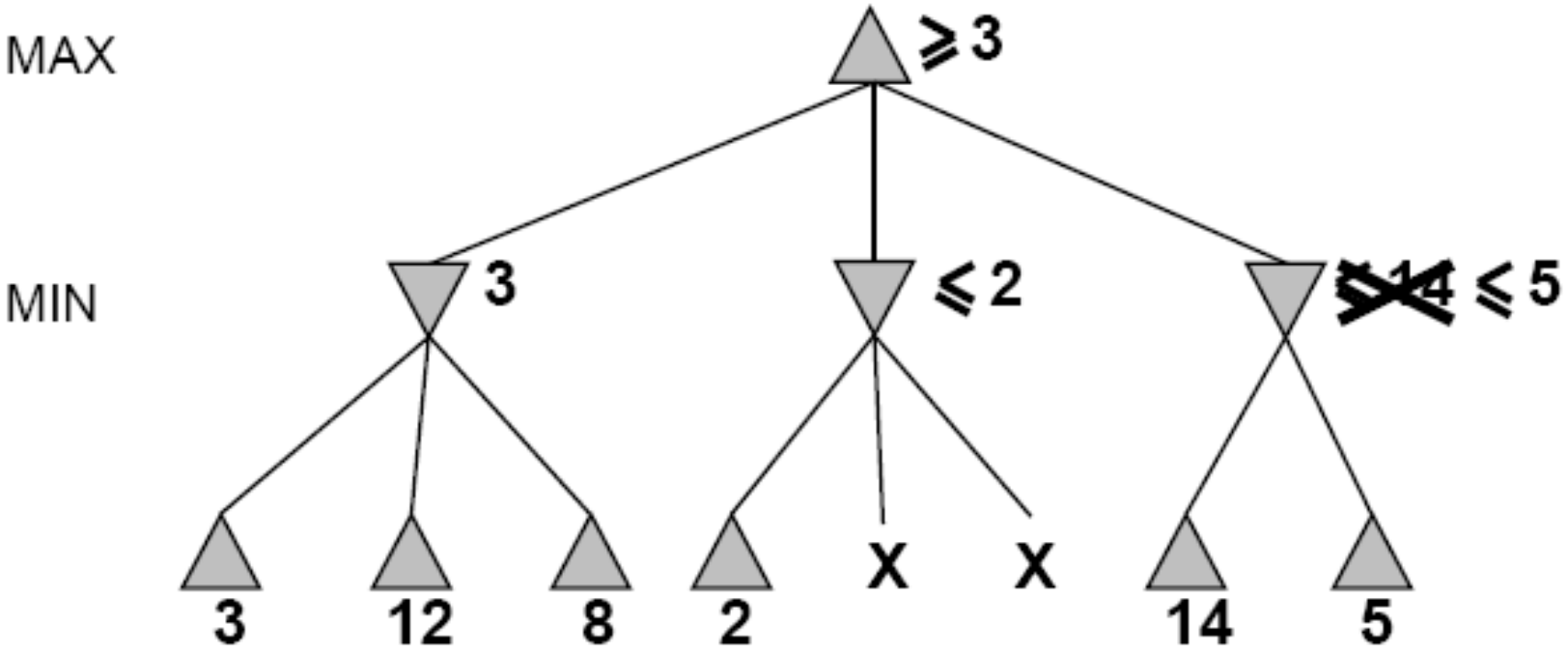
Contoh α/β pruning



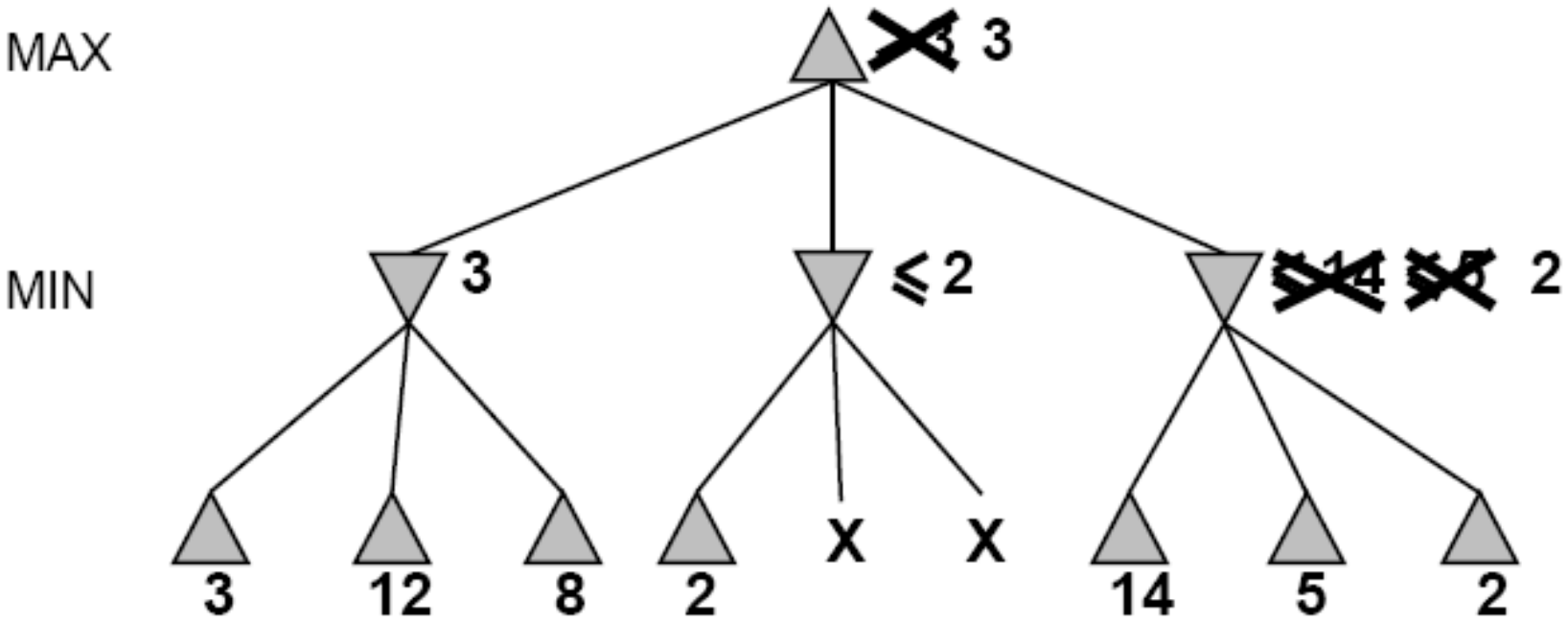
Contoh α/β pruning



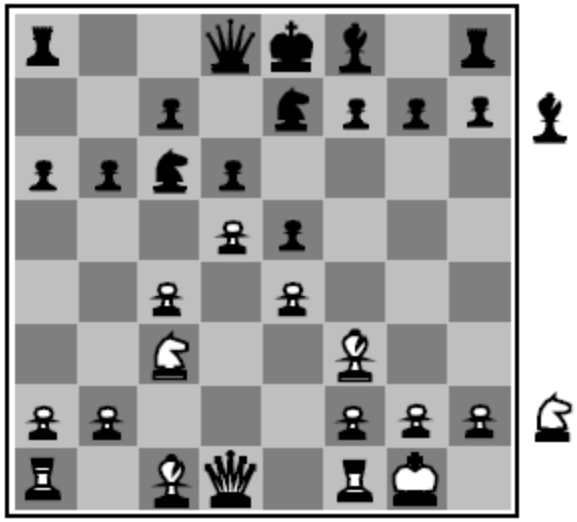
Contoh α/β pruning



Contoh α/β pruning

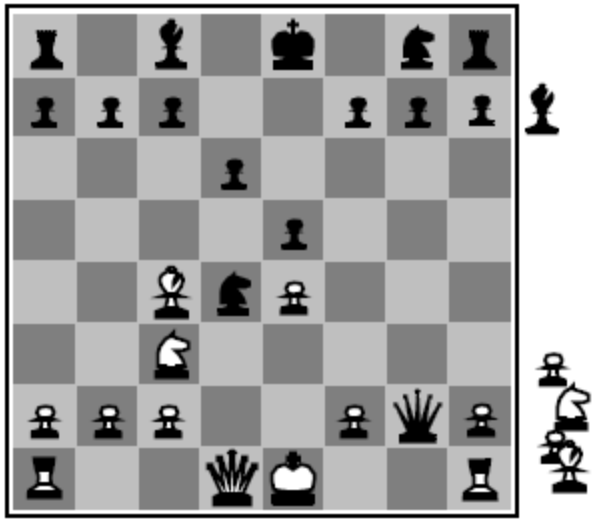


Fungsi Evaluasi



Black to move

White slightly better



White to move

Black winning

Untuk game catur, fungsi evaluasinya:

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

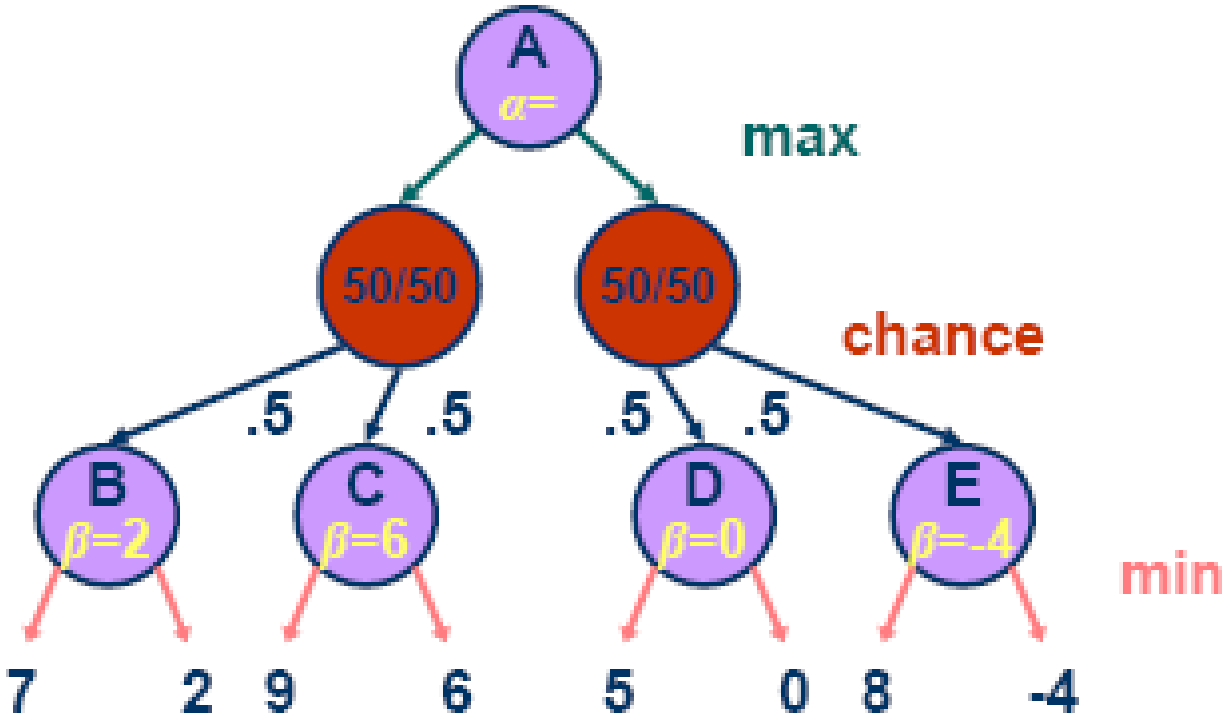
e.g., $w_1 = 9$ with

$f_1(s) =$ (number of white queens) – (number of black queens), etc.

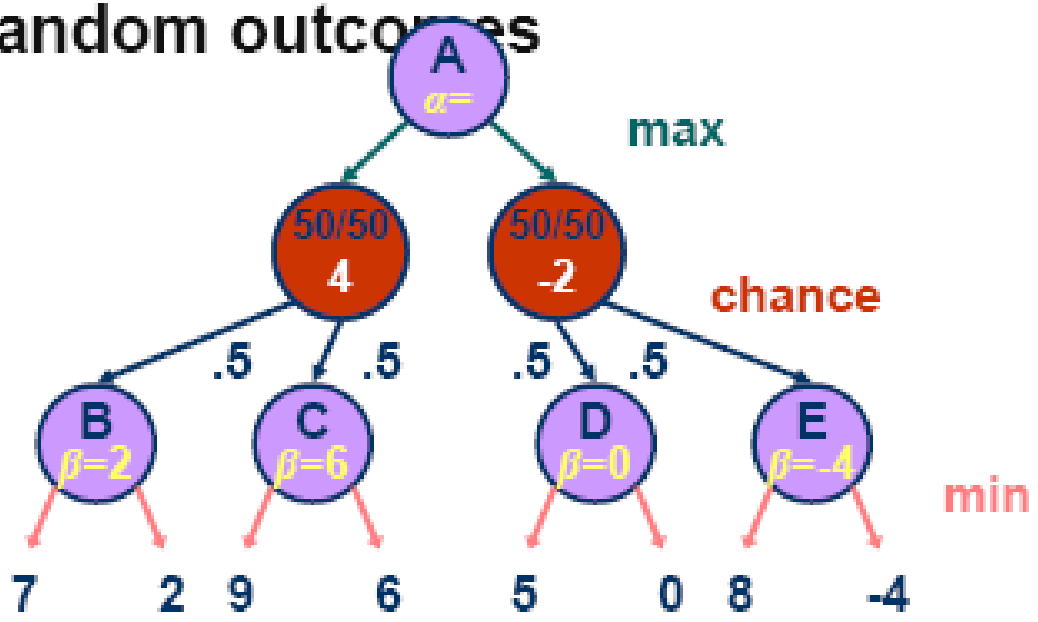
Non Deterministic games

- Beberapa games melibatkan peluang. Contoh
  - lempar dadu,
  - Memutar game wheel
- Representasi pohon game ditambahkan node chance:
  - Computer moves (max)
  - Chance node
  - Opponent moves (min)

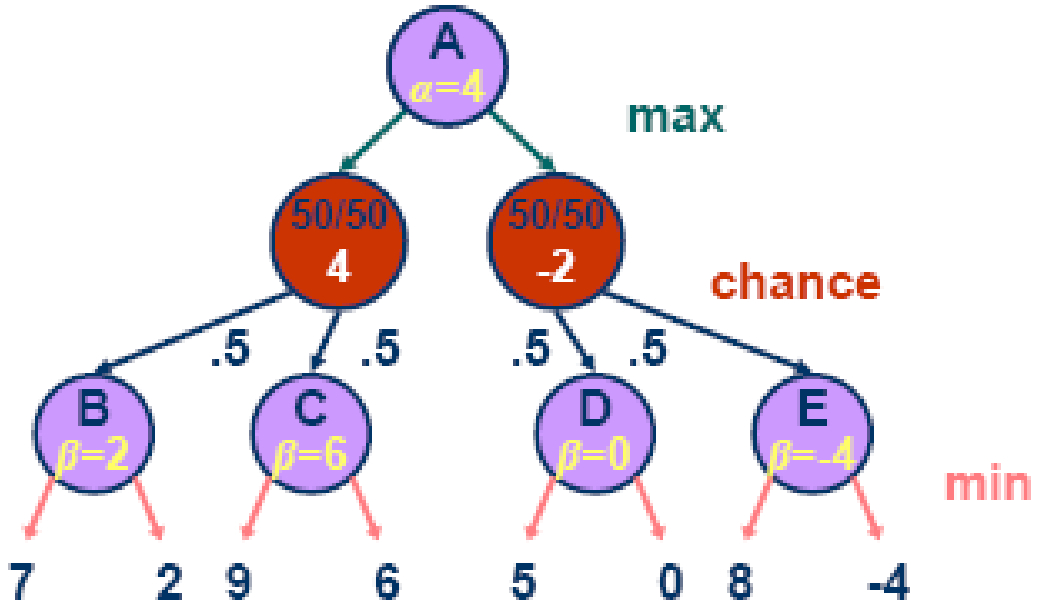
contoh



- Weight score by the probabilities that move occurs
- Use **expected value** for move: weighted sum of possible random outcomes



- Choose move with highest expected value



Deep Blue

"Deep Blue" (IBM)

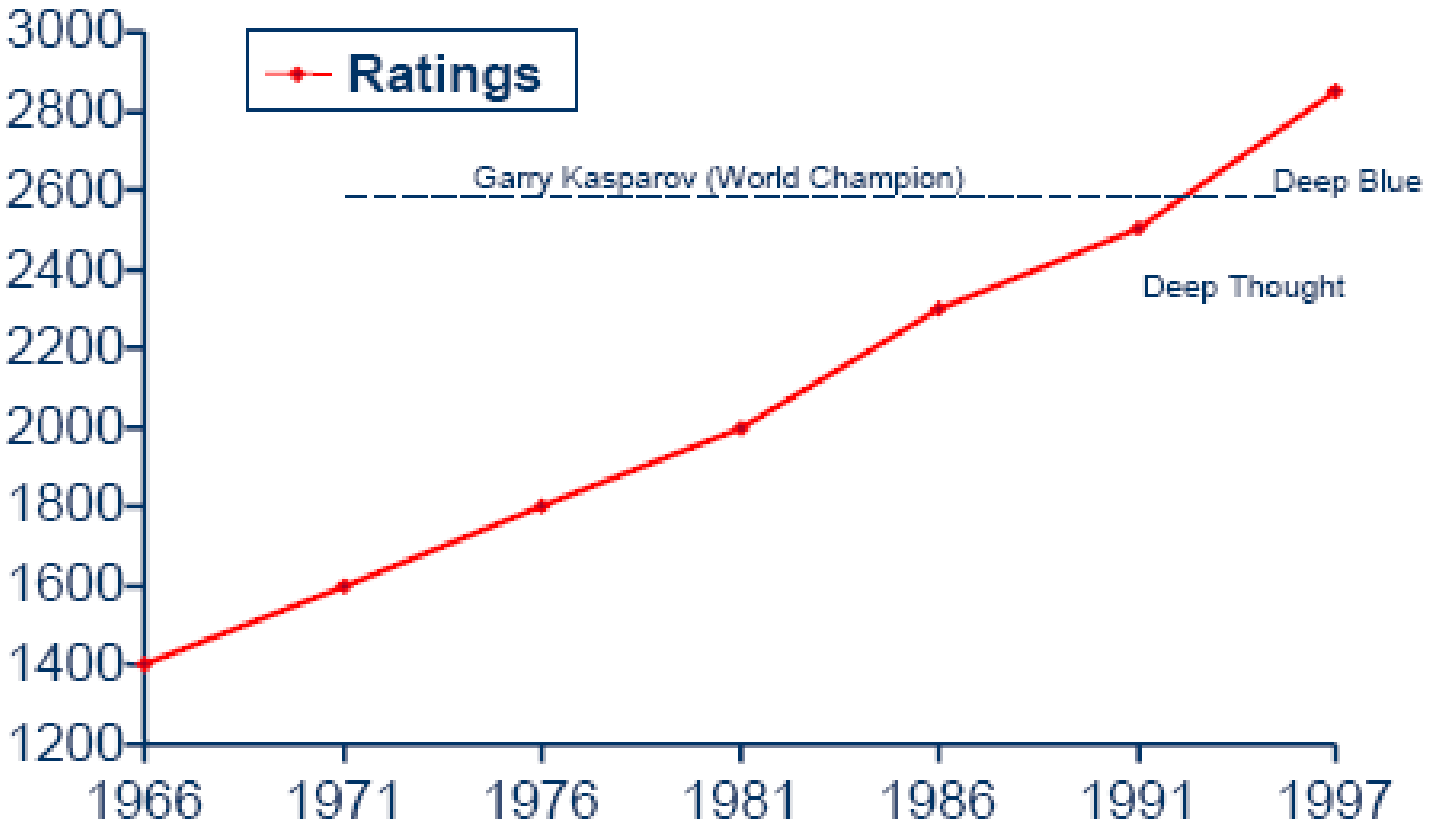
- Parallel processor, 32 nodes
- Each node had 8 dedicated VLSI "chess chips"
- Searched 200 million configurations/second
- Used minimax, alpha-beta, sophisticated heuristics
- In 2001 searched to 14 ply (i.e., 7 pairs of moves)
- Avoided horizon effect by searching as deep as 40 ply
- Used book moves

Deep Blue

Kasparov vs. Deep Blue, May 1997

- 6 game full-regulation chess match sponsored by ACM
- Kasparov lost the match 2 wins to 3 wins and 1 tie
- This was a historic achievement for computer chess being the first time a computer became the best chess player on the planet
- **Note that Deep Blue plays by "brute force" (i.e., raw power from computer speed and memory); it uses relatively little that is similar to human intuition and cleverness**

Chess Rating Scale



Deterministic games in practice

Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions.

Chess: Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.

Othello: human champions refuse to compete against computers, who are too good.

Go: human champions refuse to compete against computers, who are too bad. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves.

Key Point:

Langkah membuat deskripsi formal:

1. Definisikan Ruang keadaan/State Space (konfigurasi yang mungkin dari objek-objek yang relevan)
2. Definisikan Initial State
3. Definisikan Goal State
4. Tentukan kaidah/operator