

Breadth/Depth First Search (BFS/DFS)

(Bagian 2)

Bahan Kuliah IF2211 Strategi Algoritma

Oleh: Rinaldi M & Nur Ulfa Maulidevi



Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika ITB
2026

Pencarian Rute

(Route Finding/Path Finding)

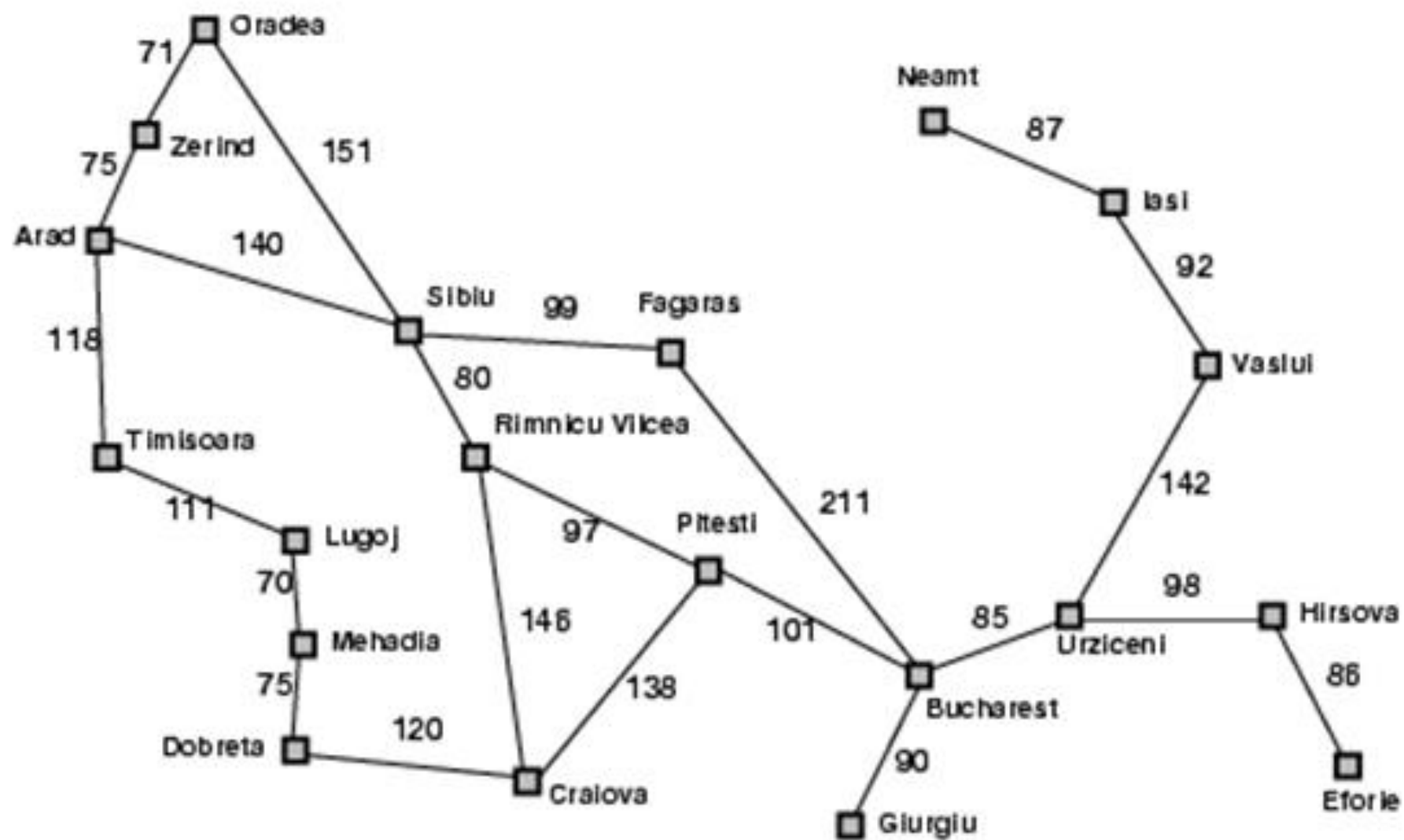
Materi Kuliah IF2211 – Strategi Algoritma
Teknik Informatika - ITB

Referensi

- Materi kuliah IF3170 Inteligensi Buatan Teknik Informatika ITB, Course Website:
<http://kuliah.itb.ac.id> → STEI → Teknik Informatika → IF3170
- Stuart J Russell & Peter Norvig, Artificial Intelligence: A Modern Approach, 3rd Edition, Prentice-Hall International, Inc, 2010, Textbook
Site: <http://aima.cs.berkeley.edu/> (2nd edition)
- Free online course materials | MIT OpenCourseWare Website:
Site: <http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/>

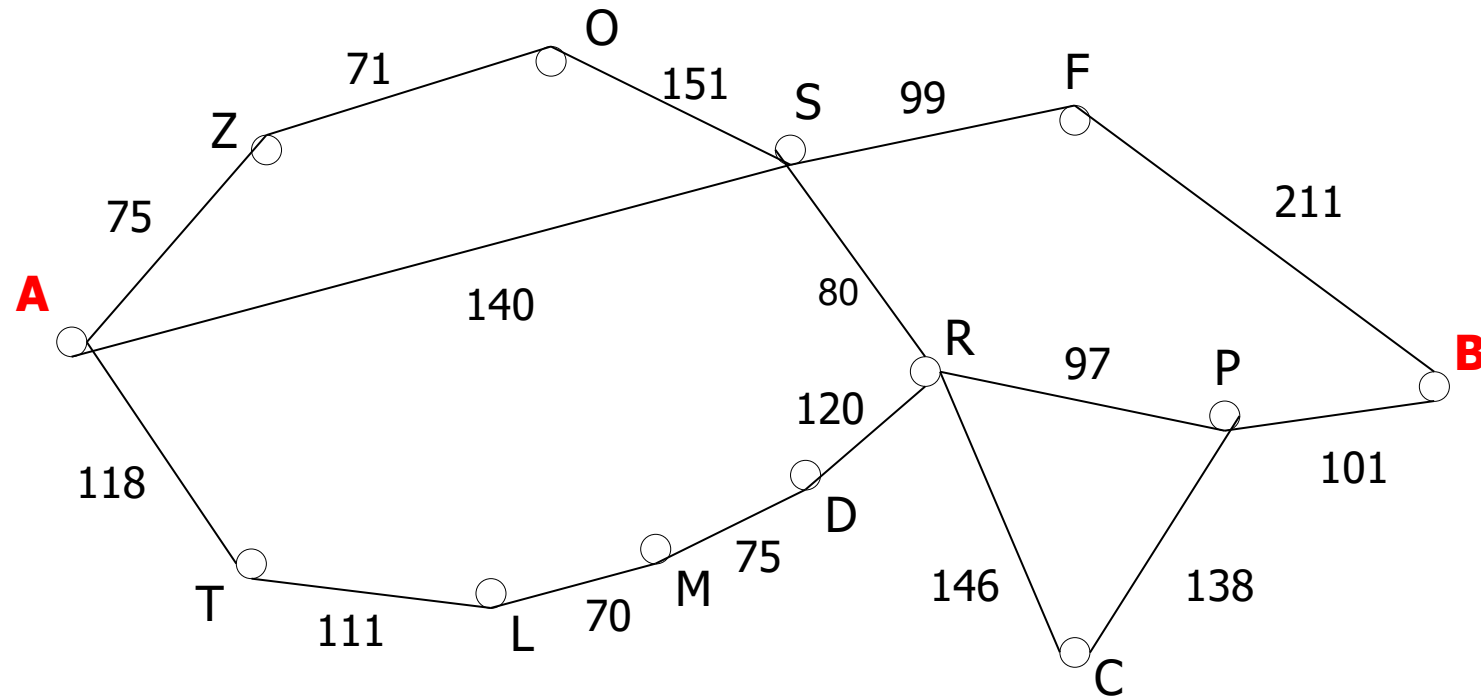
Route Planning





Source: Russell's book

Search



(a part of graph of Romania)

S: set of cities

i.s: A (Arad)

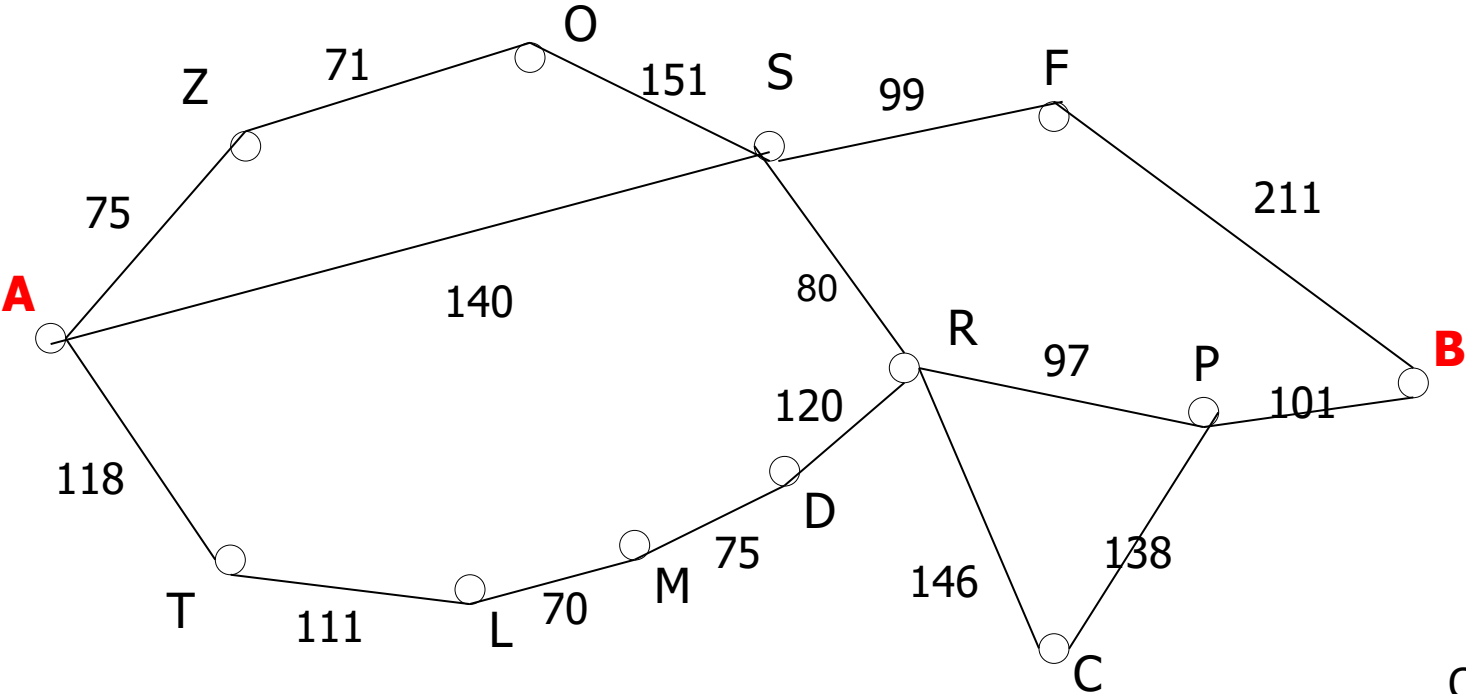
g.s: B (Bucharest)

Goal test: $s = B$?

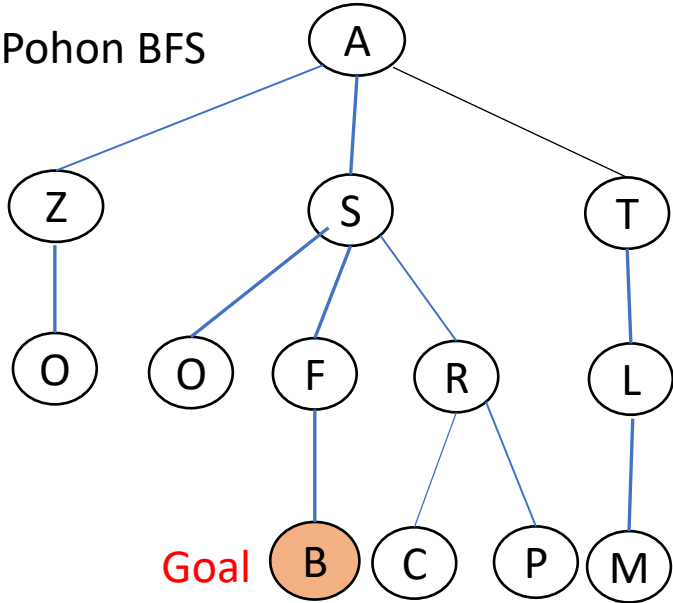
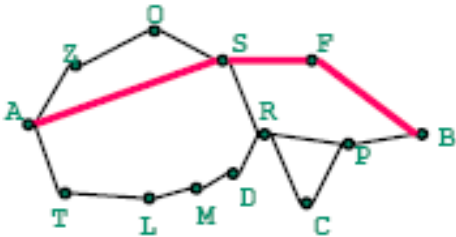
Path cost: time $\sim$ distance

Breadth-First Search (BFS)

Treat agenda as a queue (FIFO)



Path: **A → S → F → B**,
 Path-cost = 450

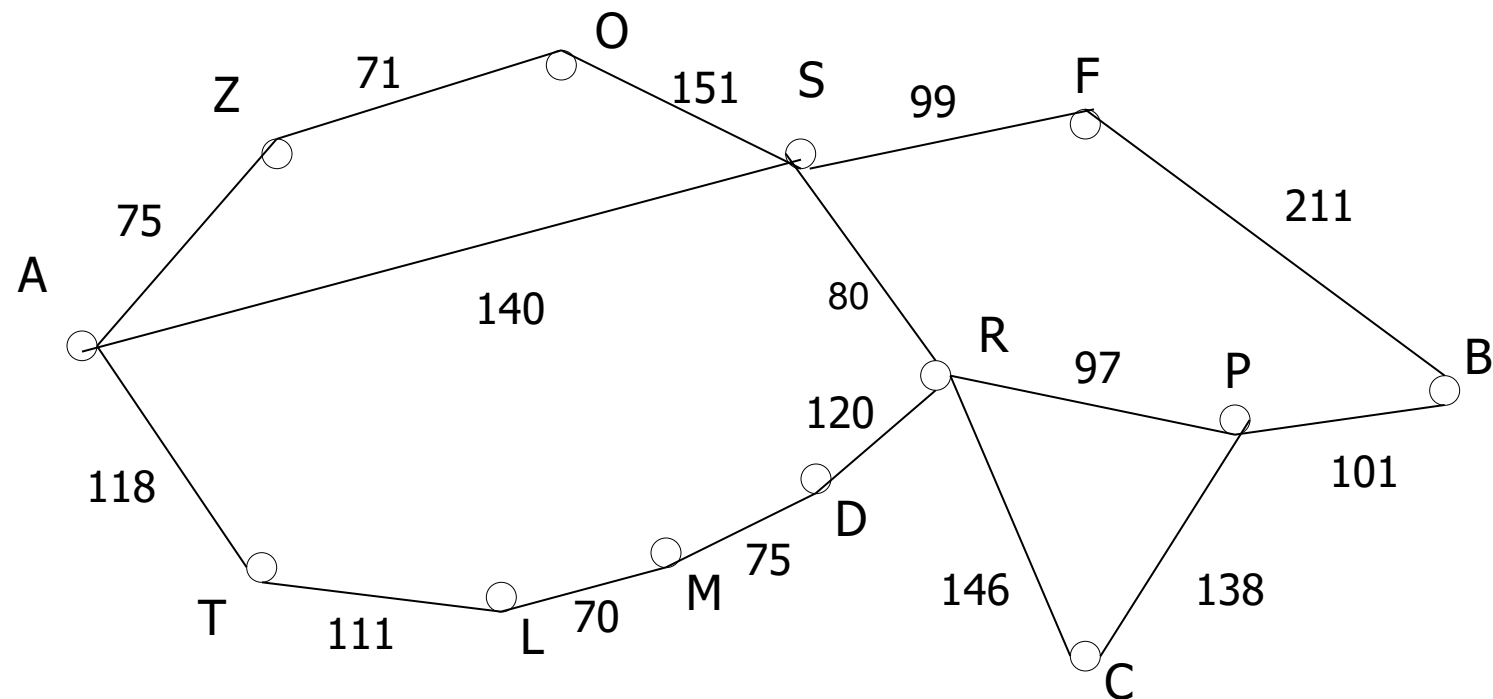


Simplu-E	Simplu Hidup
A	Z_A, S_A, T_A
Z_A	S_A, T_A, O_{AZ}
S_A	$T_A, O_{AZ}, O_{AS}, F_{AS}, R_{AS}$
T_A	$O_{AZ}, O_{AS}, F_{AS}, R_{AS}, L_{AT}$
O_{AZ}	$O_{AS}, F_{AS}, R_{AS}, L_{AT}$
O_{AS}	F_{AS}, R_{AS}, L_{AT}
F_{AS}	R_{AS}, L_{AT}, B_{ASF}
R_{AS}	$L_{AT}, B_{ASF}, D_{ASR}, C_{ASR}, P_{ASR}$
L_{AT}	$B_{ASF}, D_{ASR}, C_{ASR}, P_{ASR}, M_{ATL}$
B_{ASF}	Solusi ketemu

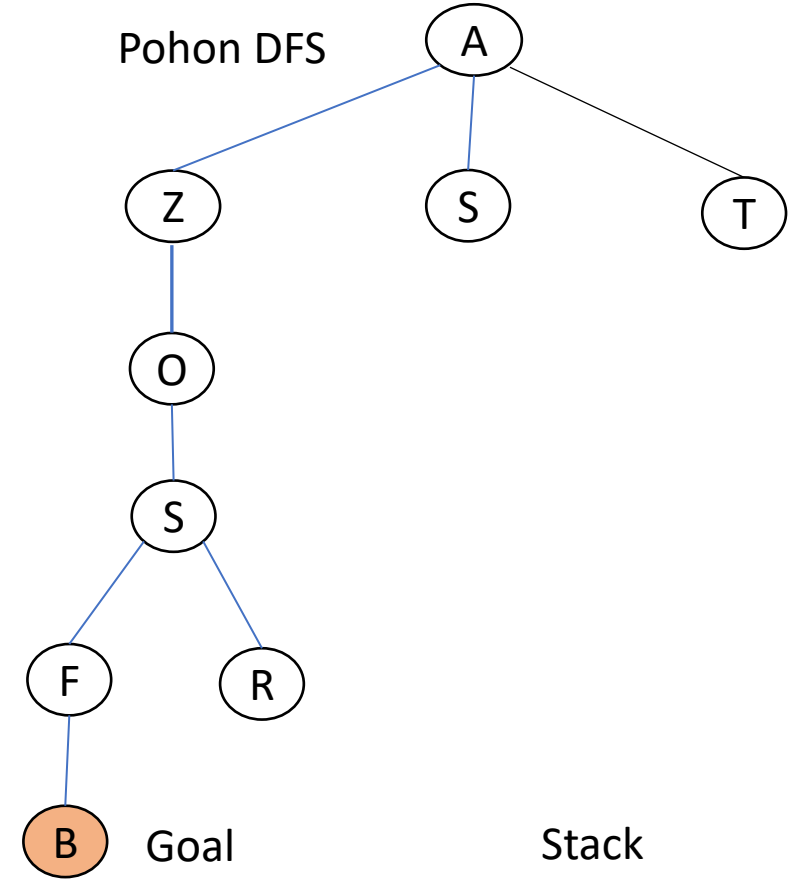
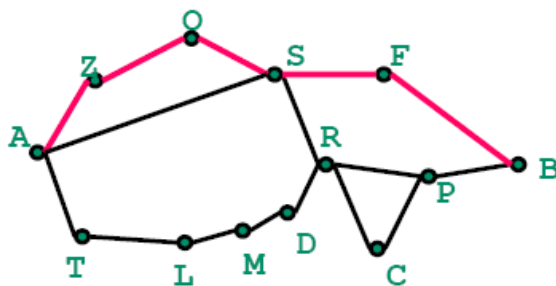
Queue

Depth-First Search (DFS)

Treat agenda as a stack (LIFO)

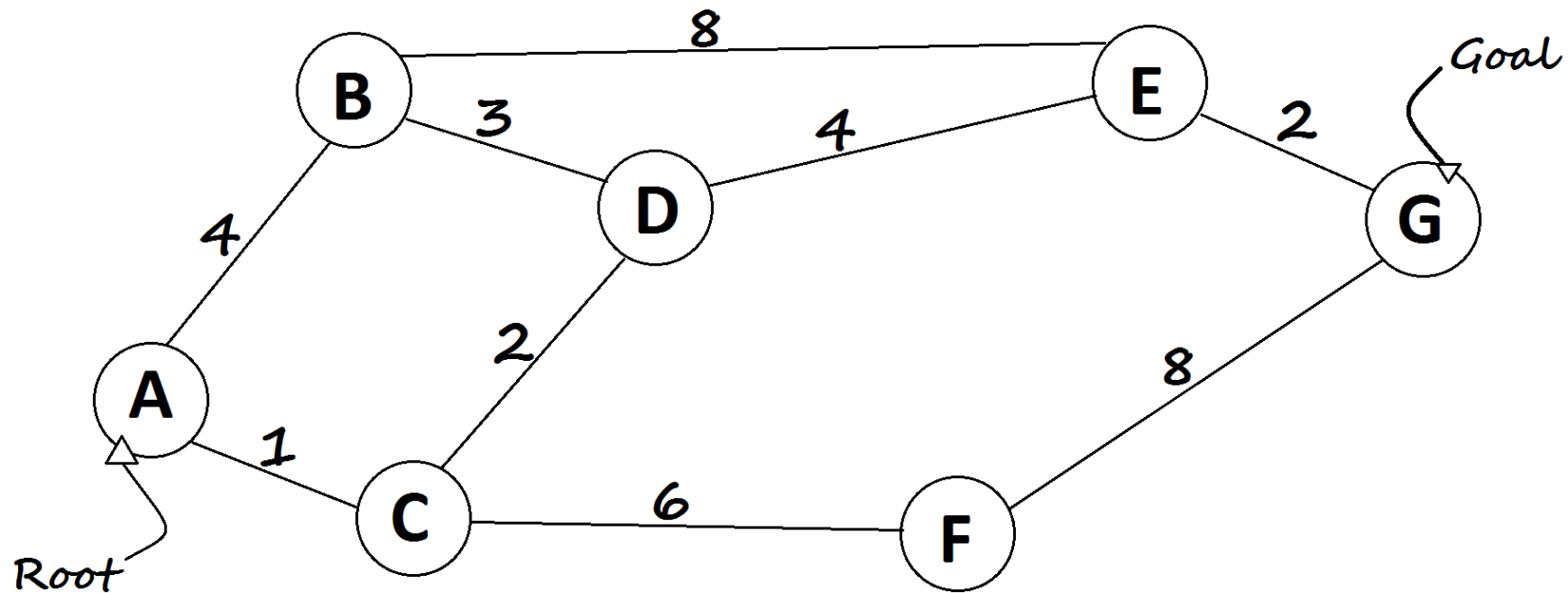


Path: $A \rightarrow Z \rightarrow O \rightarrow S \rightarrow F \rightarrow B$
Path-cost = 607



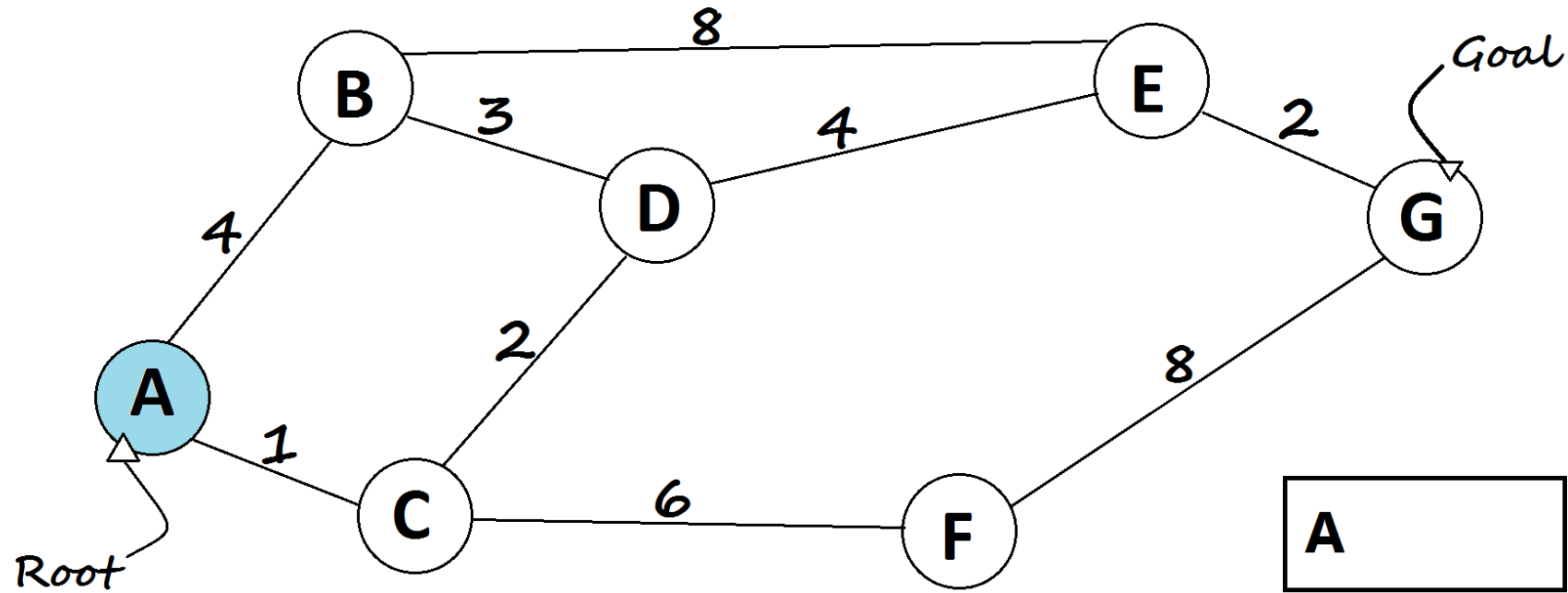
Simpul-E	Simpul Hidup
A	Z_A, S_A, T_A
Z_A	O_{AZ}, S_A, T_A
O_{AZ}	S_{AZO}, S_A, T_A
S_{AZO}	$F_{AZOS}, R_{AZOS}, S_A, T_A$
F_{AZOS}	$B_{AZOSF}, R_{AZOS}, S_A, T_A$
B_{AZOSF}	Solusi ketemu

Contoh *path finding* lainnya:



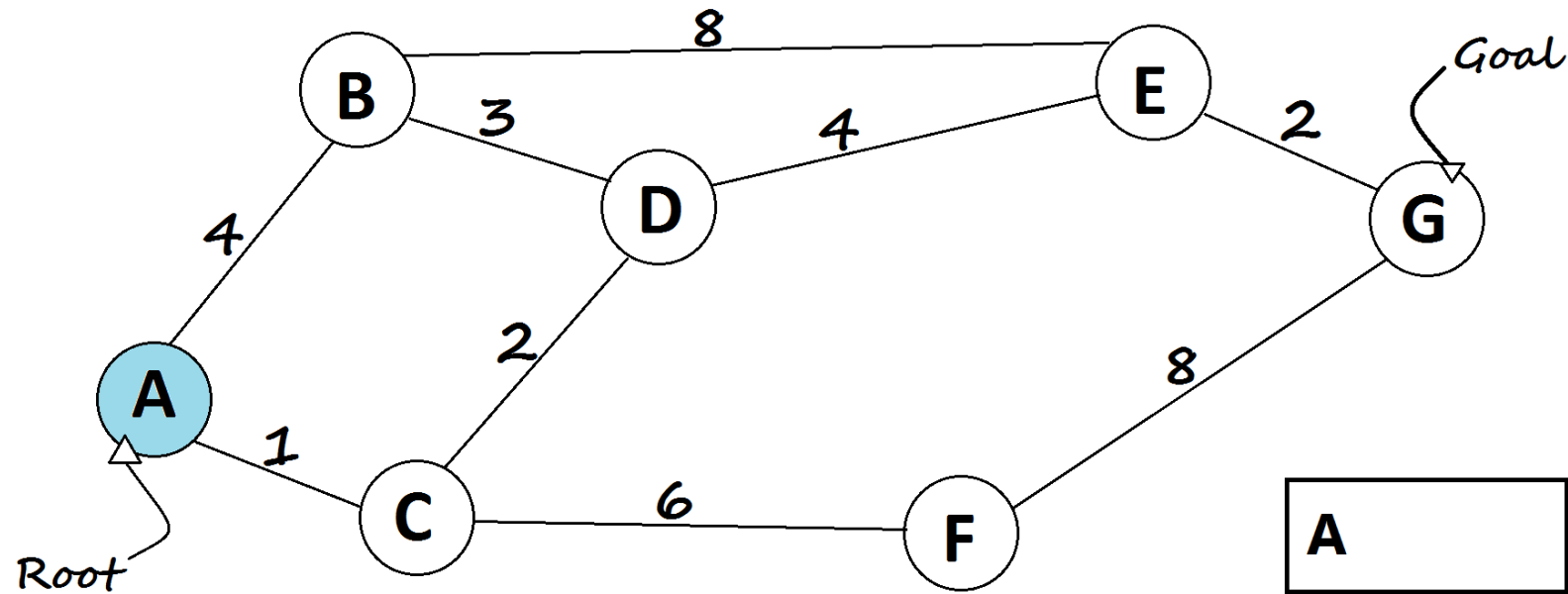
Sumber: <https://medium.com/omarelgabrys-blog/path-finding-algorithms-f65a8902eb40>

BFS:



Sumber: <https://medium.com/omarelgabrys-blog/path-finding-algorithms-f65a8902eb40>

DFS:



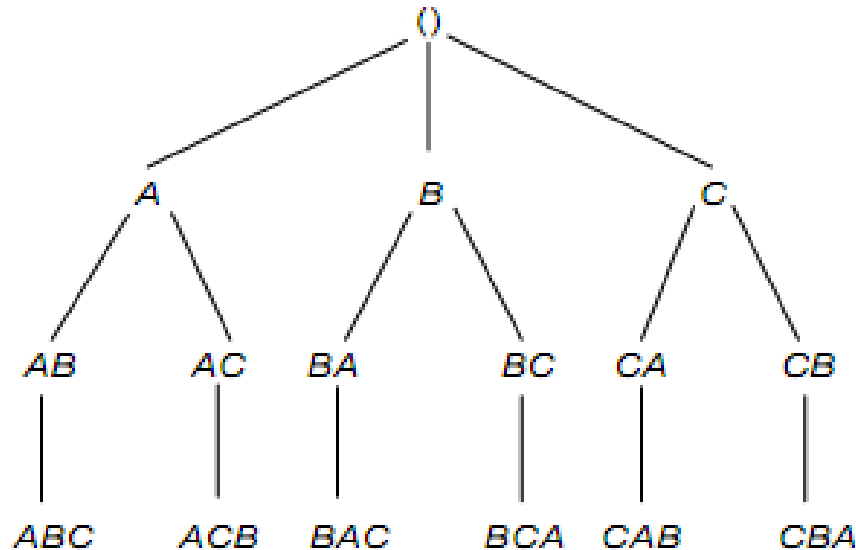
Sumber: <https://medium.com/omarelgabrys-blog/path-finding-algorithms-f65a8902eb40>

Graf Dinamis

Pencarian Solusi dengan BFS/DFS

- Menyelesaikan persoalan dengan melakukan pencarian solusi sambil membangun graf (pohon) secara bertahap
- Pencarian solusi $\approx$ pembentukan pohon dinamis
 - Simpul di-*create*, lalu diperiksa apakah merupakan simpul solusi (goal) atau bukan. Jika simpul merupakan solusi, pencarian selesai (jika hanya satu solusi ditemukan) atau dilanjutkan mencari solusi lain (semua solusi ditemukan).
- Representasi pohon dinamis:
 - Dinamakan pohon ruang status (*state space tree*)
 - Simpul menyatakan status persoalan (*problem state*)
 - Akar: *initial state*
 - Daun: *solution/goal state*
 - Cabang: operator/langkah dalam persoalan
 - Ruang status (*state space*): himpunan semua simpul
 - Ruang solusi: himpunan status solusi
- Solusi: lintasan dari akar ke simpul solusi

Pohon Dinamis: Permutasi A,B,C



Ket: () = status kosong

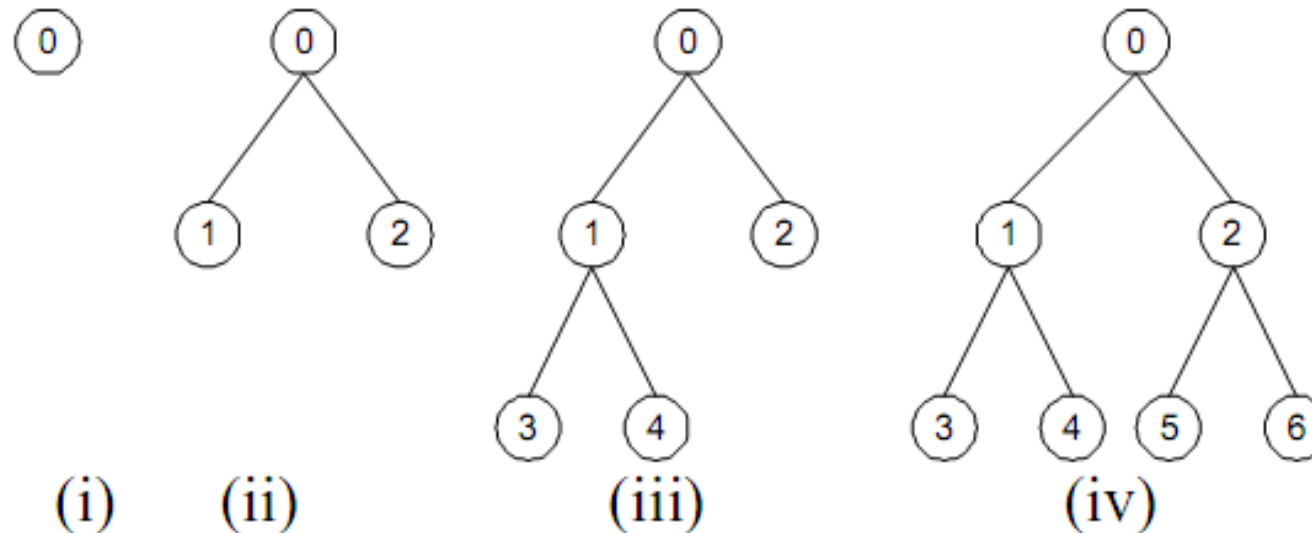
Pohon ruang status

- Operator: add X
- Akar : status awal (status “kosong”)
- Simpul: problem state
 - Status persoalan (*problem state*): simpul-simpul di dalam pohon dinamis yang memenuhi kendala (constraints).
- Daun: status solusi
 - Status solusi (*solution state*): satu atau lebih status yang menyatakan solusi persoalan.
- Ruang solusi:
 - Ruang solusi (*solution space*): himpunan semua status solusi.
- Ruang status (*state space*): Seluruh simpul di dalam pohon dinamis dan pohonnya dinamakan juga pohon ruang status (*state space tree*).

Pembangkitan Status

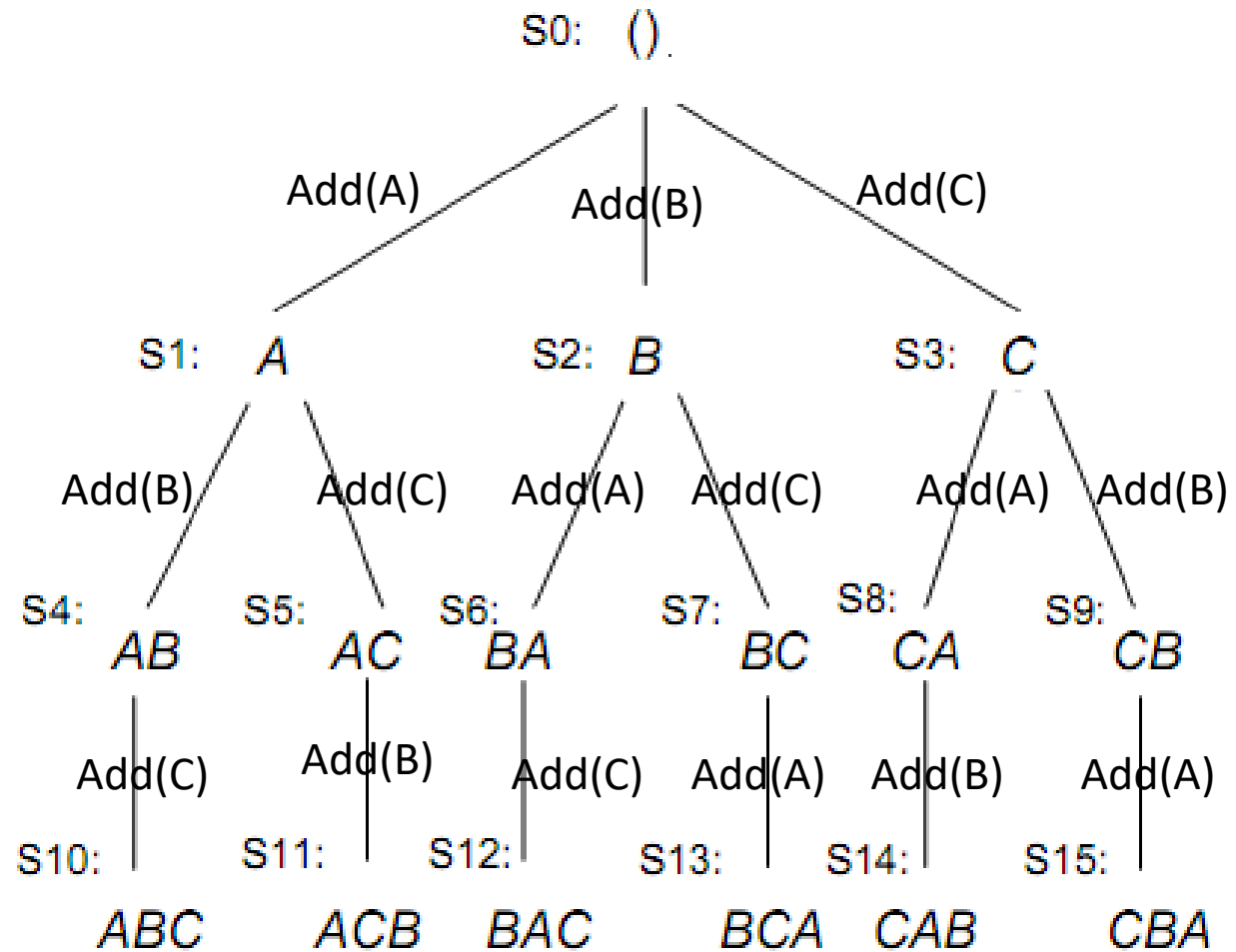
- Pembangkitan status baru dengan cara mengaplikasikan operator (langkah legal) kepada status (simpul) pada suatu jalur
- Jalur dari simpul akar sampai ke simpul (daun) goal berisi rangkaian operator yang mengarah pada solusi persoalan

BFS untuk Pembentukan Pohon Ruang Status



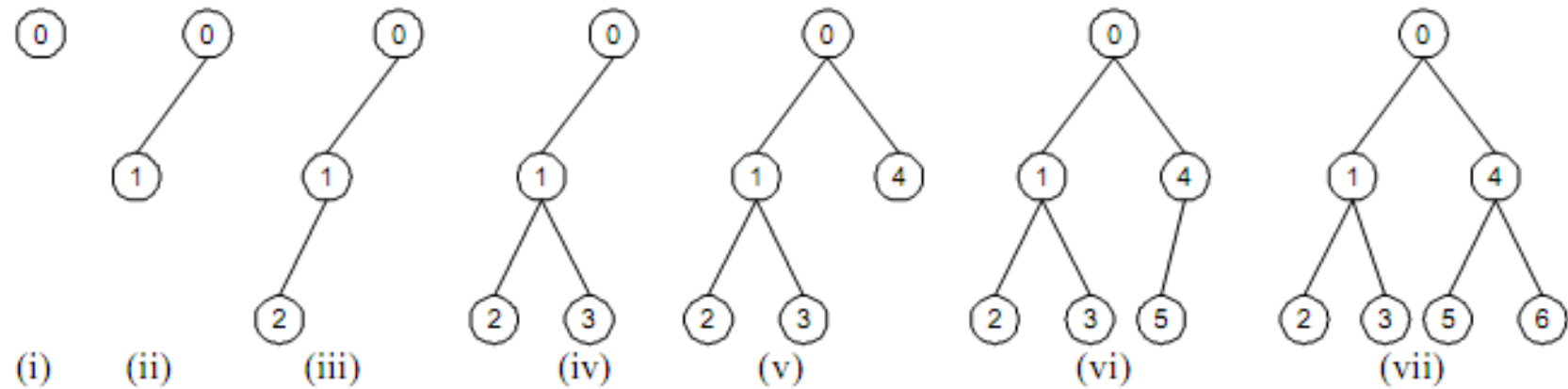
- Inisialisasi dengan status awal sebagai akar, lalu tambahkan simpul anaknya, dst.
- Semua simpul pada level d dibangkitkan terlebih dahulu sebelum simpul-simpul pada level $d+1$

BFS untuk Permutasi

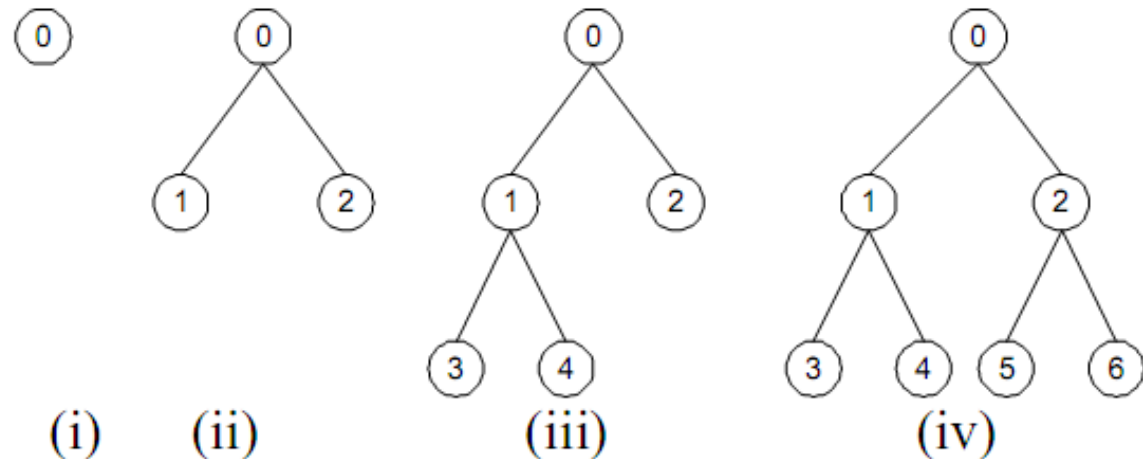


DFS untuk Pembentukan Pohon Ruang Status

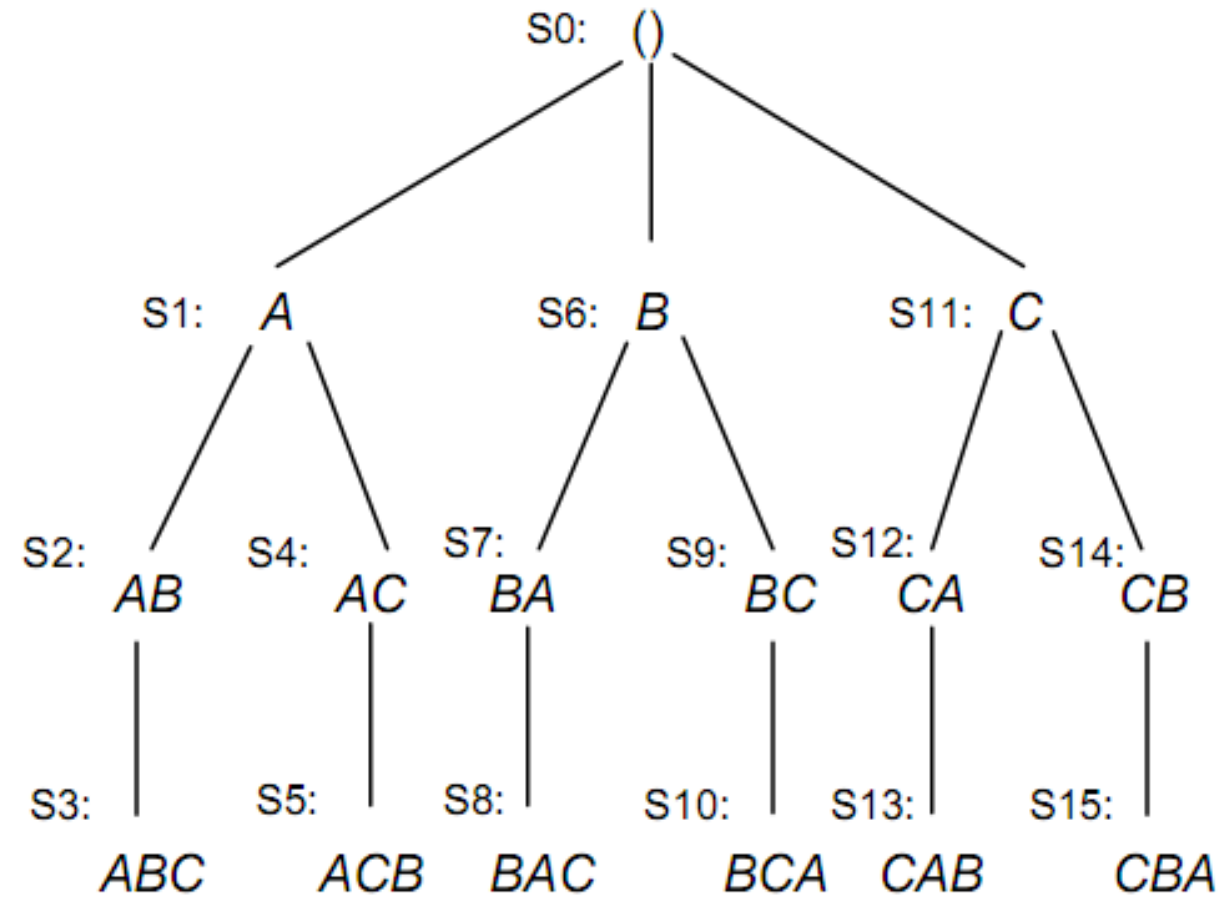
DFS:



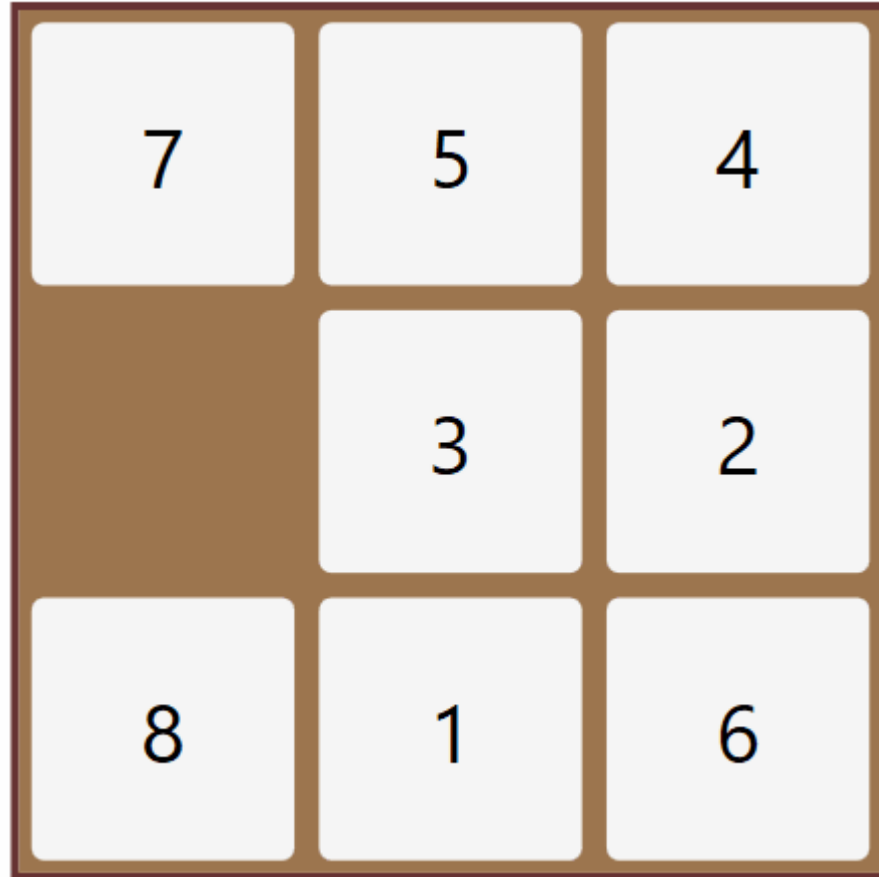
BFS:



DFS Permutasi A,B,C



Permainan 8-*Puzzle*



2	1	6
4		8
7	5	3

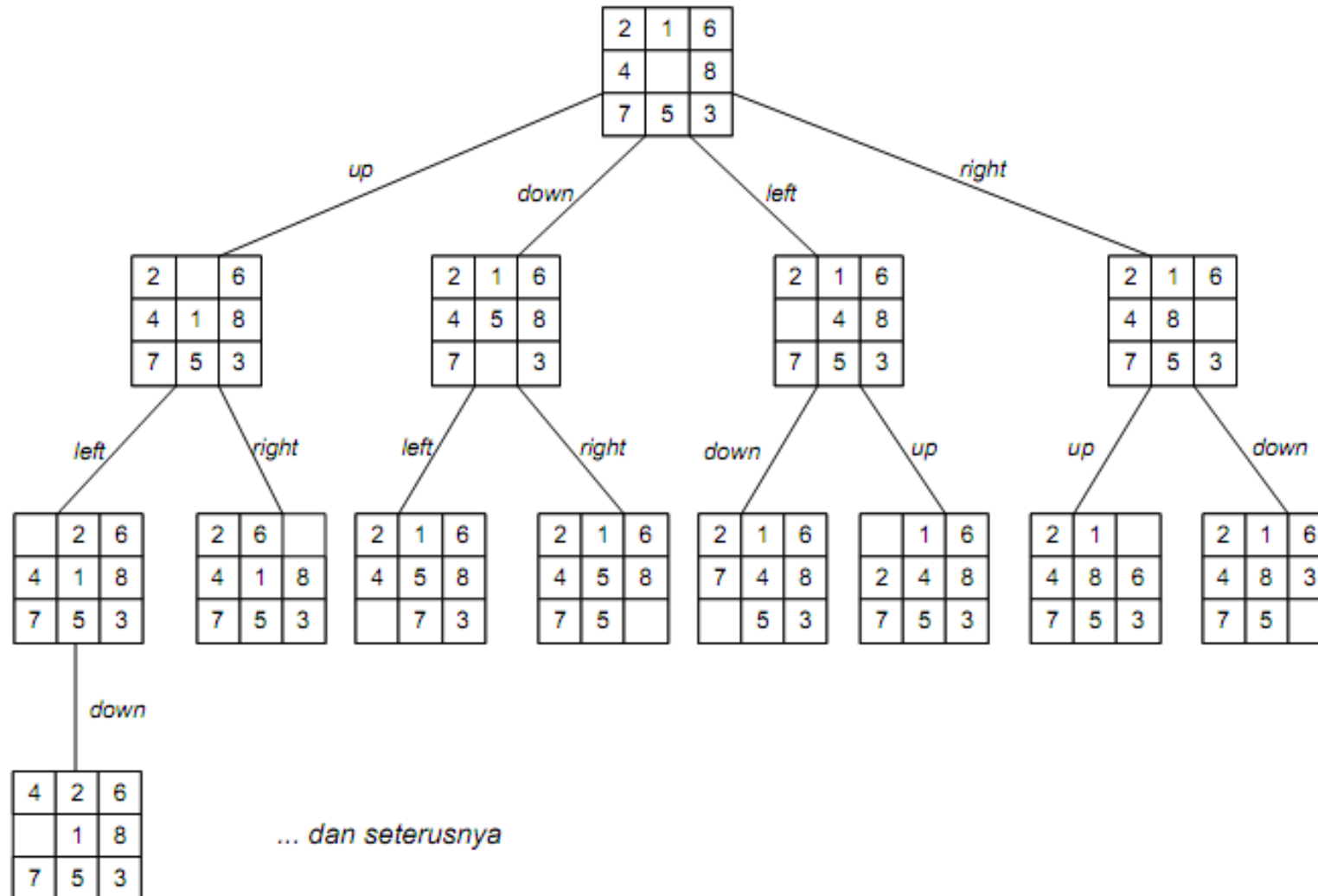
(a) Susunan awal
(*initial state*)

1	2	3
8		4
7	6	5

(b) Susunan akhir
(*goal state*)

- State berdasarkan ubin kosong (*blank*)
- Operator: *up, down, left, right*

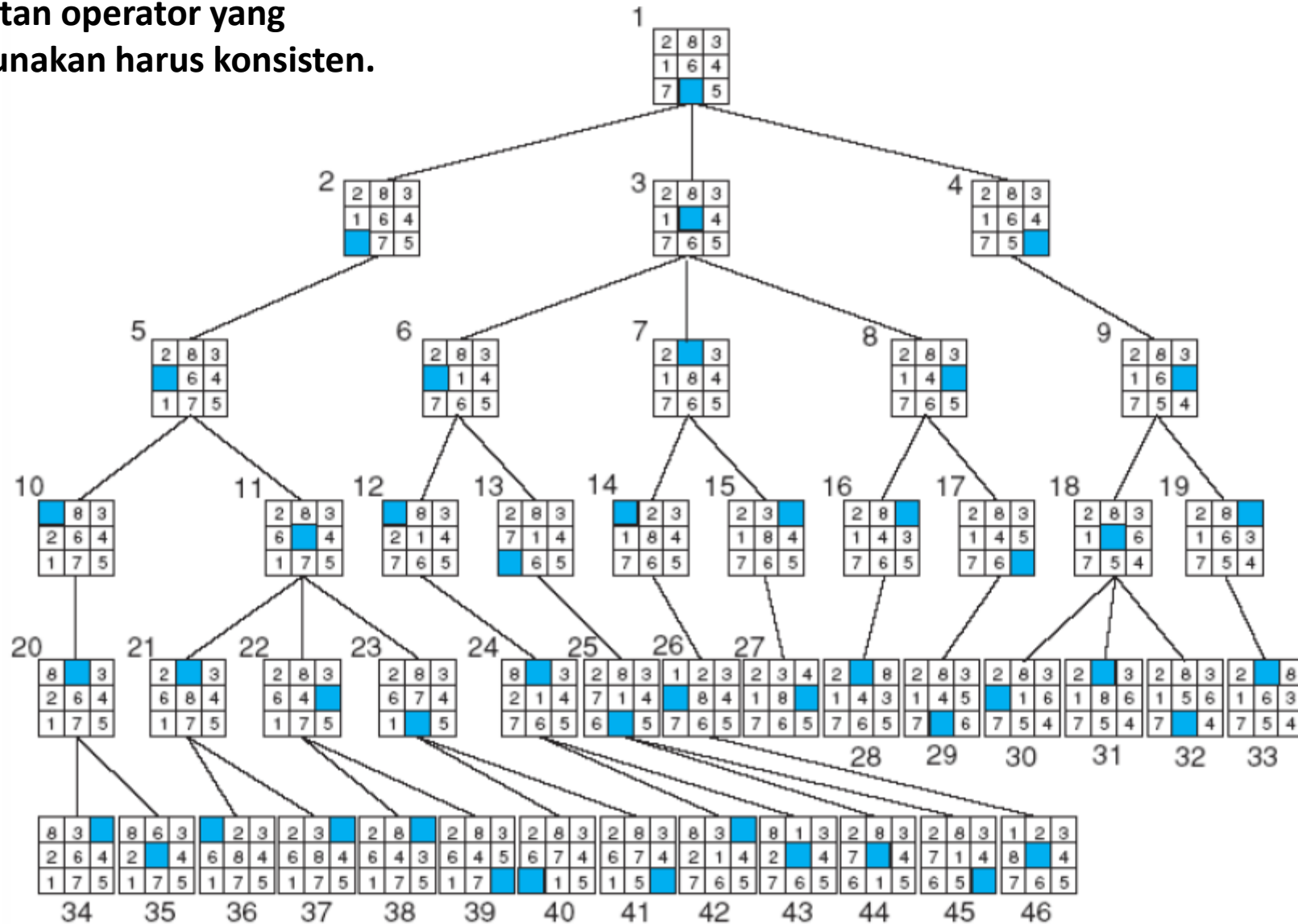
8-Puzzle: Pohon Ruang Status



BFS untuk 8-Puzzle

Catatan:

Urutan operator yang digunakan harus konsisten.



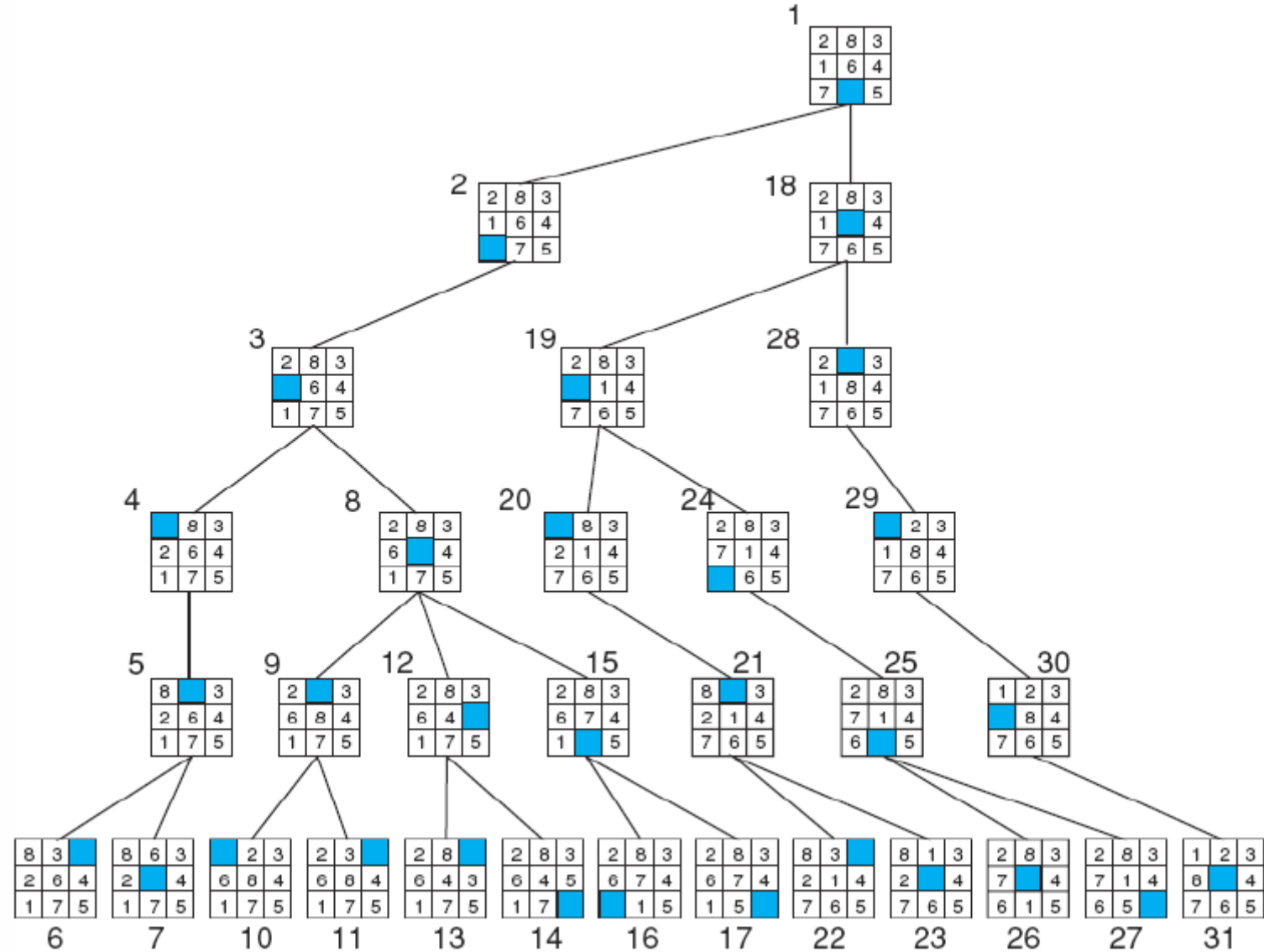
Evaluasi Teknik Pencarian

- Properti untuk mengukur seberapa 'baik' suatu teknik pencarian:
 - *Completeness*: apakah teknik pencarian menjamin ditemukannya solusi (jika ada)
 - *Optimality*: apakah teknik pencarian menjamin mendapatkan solusi yang optimal (*e.g*: *lowest path cost*)?
 - *Time Complexity*: waktu yang diperlukan untuk mencapai simpul solusi
 - *Space Complexity*: ruang memori yang diperlukan selama melakukan pencarian
- Kompleksitas waktu dan ruang diukur dengan menggunakan istilah berikut.
 - *b*: (*branching factor*) maksimum pencabangan yang mungkin dari suatu simpul
 - *d*: (*depth*) kedalaman dari solusi terbaik (*cost* terendah)
 - *m*: maksimum kedalaman dari ruang status (bisa ∞)

Bagaimana properti BFS?

- *Completeness?*
 - Ya (selama nilai b terbatas)
- *Optimality?*
 - Ya, jika langkah = ongkos
- Kompleksitas waktu:
 - $1+b+b^2+b^3+\dots+b^d = O(b^d)$
- Kompleksitas ruang:
 - $O(b^d)$
- Kurang baik dalam kompleksitas ruang

DFS untuk 8-Puzzle



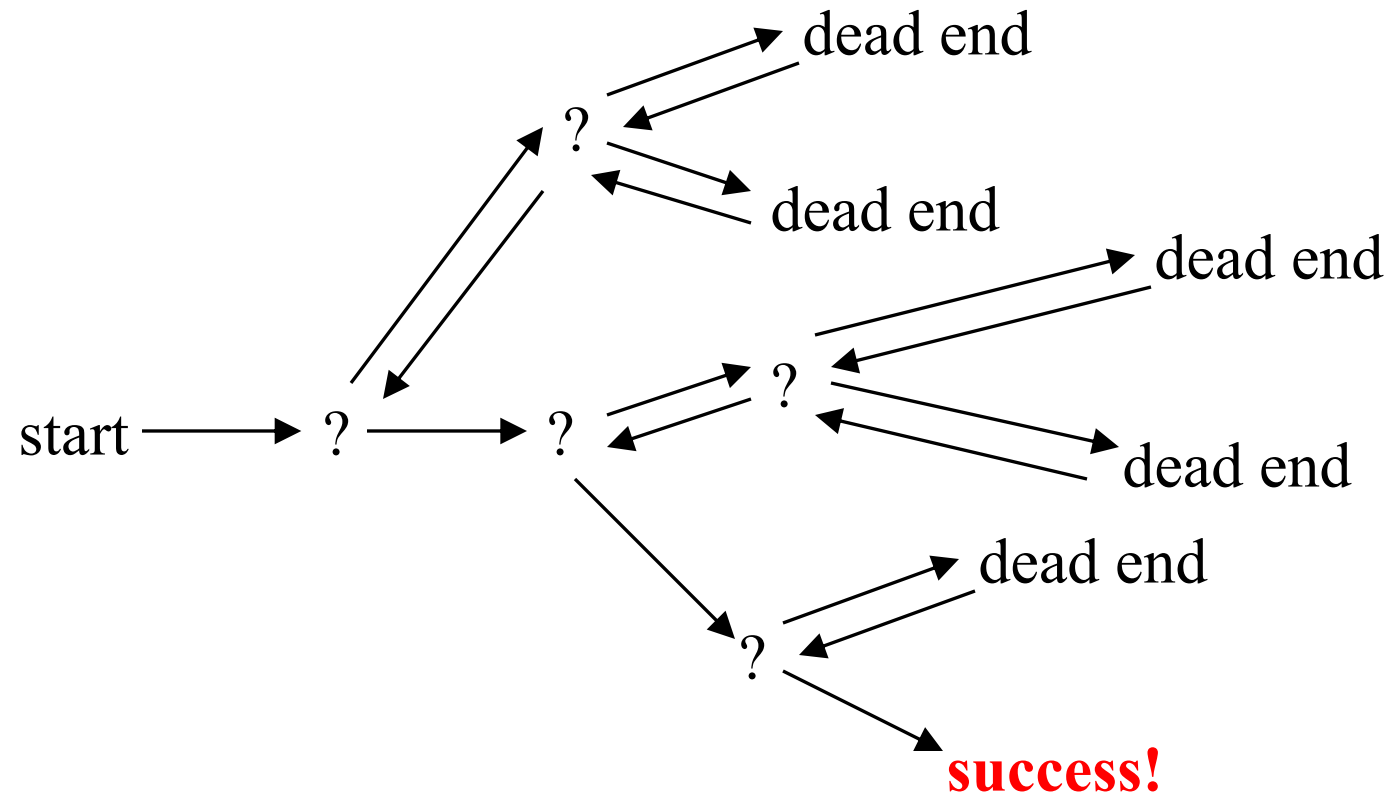
Bagaimana properti DFS?

- *Completeness?*
 - Ya (selama nilai b terbatas, dan ada penanganan '*redundant paths*' dan '*repeated states*')
- *Optimality?*
 - Tidak
- Kompleksitas waktu:
 - $O(b^m)$
- Kompleksitas ruang:
 - $O(bm)$
- Kurang baik dalam kompleksitas waktu, lebih baik dalam kompleksitas ruang

Aplikasi *Backtracking* di dalam DFS untuk Pencarian Solusi

- Karakteristik *backtracking* di dalam algoritma DFS berguna untuk memecahkan persoalan pencarian solusi yang memiliki banyak alternatif pilihan selama pencarian.
- Solusi diperoleh dengan mengekspansi pencarian menuju *goal* dengan aturan “depth-first”
 - Anda tidak punya cukup informasi untuk mengetahui apa yang akan dipilih
 - Tiap keputusan mengarah pada sekumpulan pilihan baru
 - Beberapa sekuens pilihan (bisa lebih dari satu) mungkin merupakan solusi persoalan
- **Backtracking** di dalam algoritma DFS adalah cara yang sistematis mencoba beberapa sekuens keputusan,
- sampai Anda menemukan sekuens yang “bekerja”

Animasi *Backtracking* *)

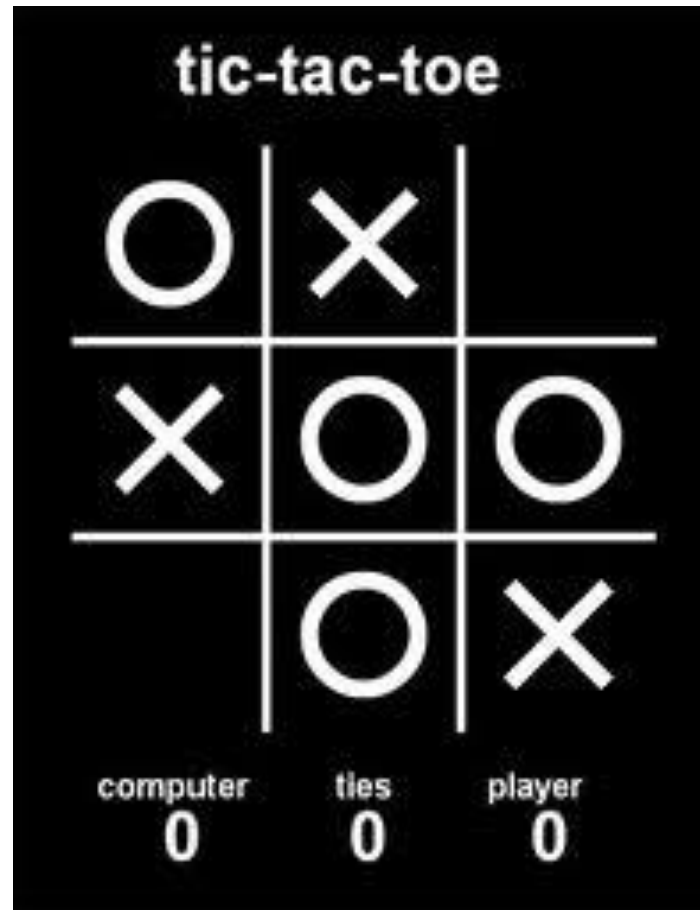


- Backtracking di dalam algoritma DFS banyak diterapkan untuk mencari solusi persoalan *game* seperti:
 - permainan *tic-tac-toe*,
 - menemukan jalan keluar dalam sebuah labirin,
 - Catur, *crossword puzzle*, *sudoku*, dan masalah-masalah pada bidang kecerdasan buatan (*artificial intelligence*).

Crossword puzzle:



Tic-Tac-Toe



Sudoku

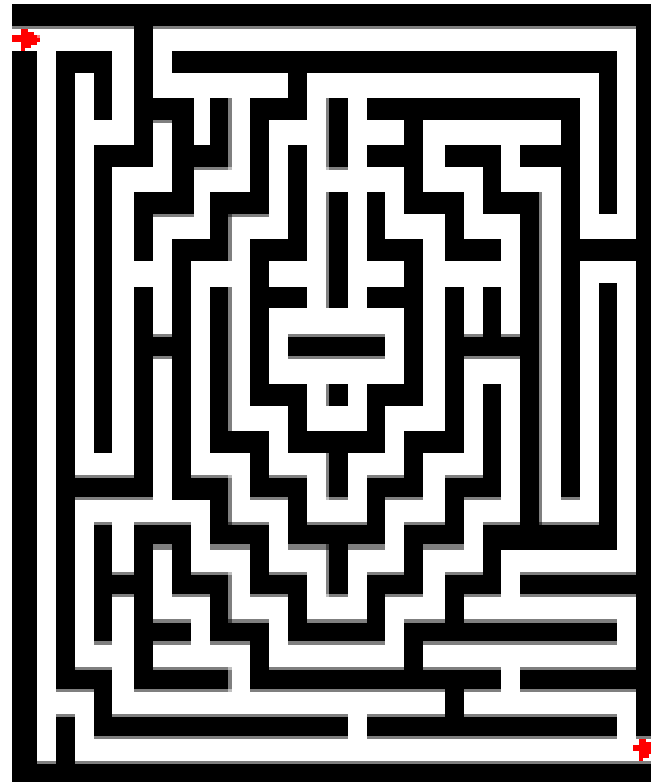
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Catur



Maze Problem

Diberikan sebuah labirin (*maze*), temukan lintasan dari titik awal sampai titik akhir



Source			
			Destination

Maze

1	0	0	0
1	1	0	1
0	1	0	0
1	1	1	1

Matriks

- Pada tiap perpotongan, anda harus memutuskan satu diantara tiga pilihan:
 - Maju terus
 - Belok kiri
 - Belok kanan
- Anda tidak punya cukup informasi untuk memilih pilihan yang benar (yang mengarah ke titik akhir)
- Tiap pilihan mengarah ke sekumpulan pilihan lain
- Satu atau lebih sekuens pilihan mengarah ke solusi.

Penyelesaian *Maze Problem* dengan DFS:

- Bagi lintasan menjadi sederetan langkah.
- Sebuah langkah terdiri dari pergerakan satu unit sel pada arah tertentu.
- Arah yang mungkin: lurus (*straight*), kiri (*left*), ke kanan (*right*).

Garis besar algoritma DFS:

```
while belum sampai pada tujuan do  
    if terdapat arah yang benar sedemikian sehingga kita belum pernah  
        berpindah ke sel pada arah tersebut  
    then  
        pindah satu langkah ke arah tersebut  
    else  
        backtrack langkah sampai terdapat arah seperti yang disebutkan  
        di atas  
    endif  
endwhile
```

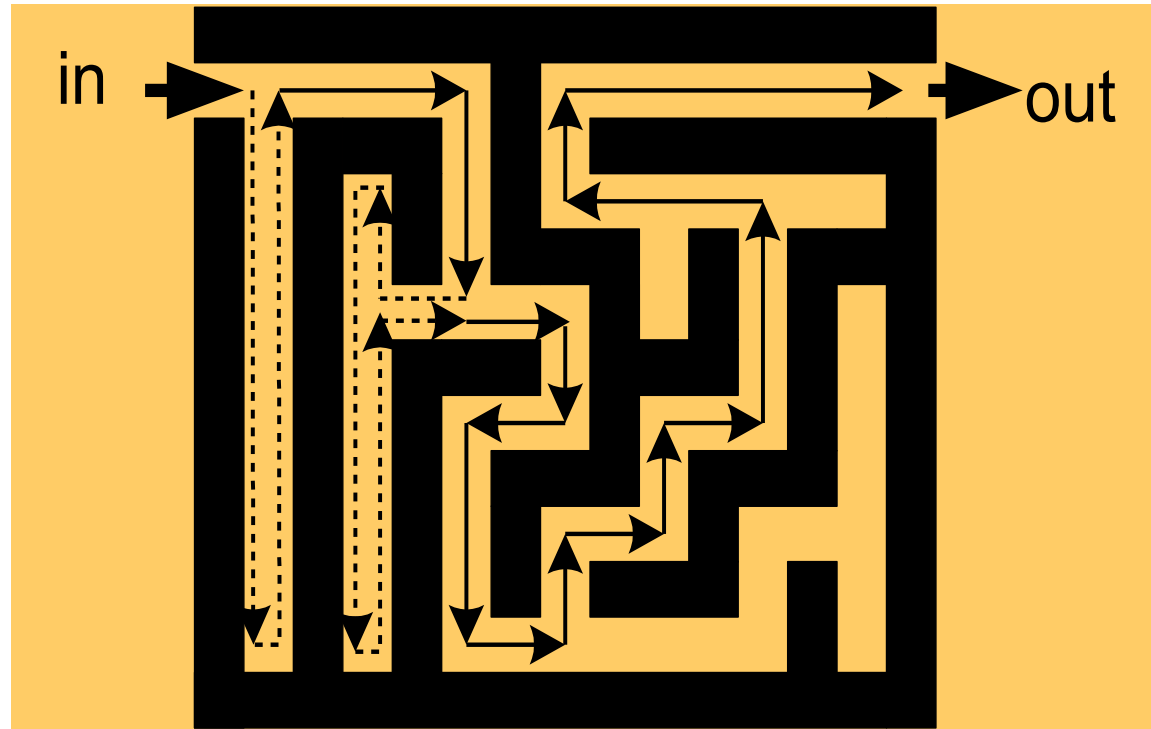
- Bagaimana mengetahui langkah yang mana yang perlu dijejaki kembali?
- Ada dua solusi untuk masalah ini:
 1. Simpan semua langkah yang pernah dilakukan, atau
 2. gunakan rekursi (yang secara implisit menyimpan semua langkah).
- Rekursi adalah solusi yang lebih mudah.

```
Depth-First-Search-Kickoff( Maze m )  
    Depth-First-Search( m.StartCell )  
End procedure
```

```
Depth-First-Search( MazeCell c )  
    If c is the goal  
        Exit  
    Else  
        Mark c "Visit In Progress"  
        Foreach neighbor n of c  
            If n "Unvisited"  
                Depth-First-Search( n )  
        Mark c "Visited"  
End procedure
```

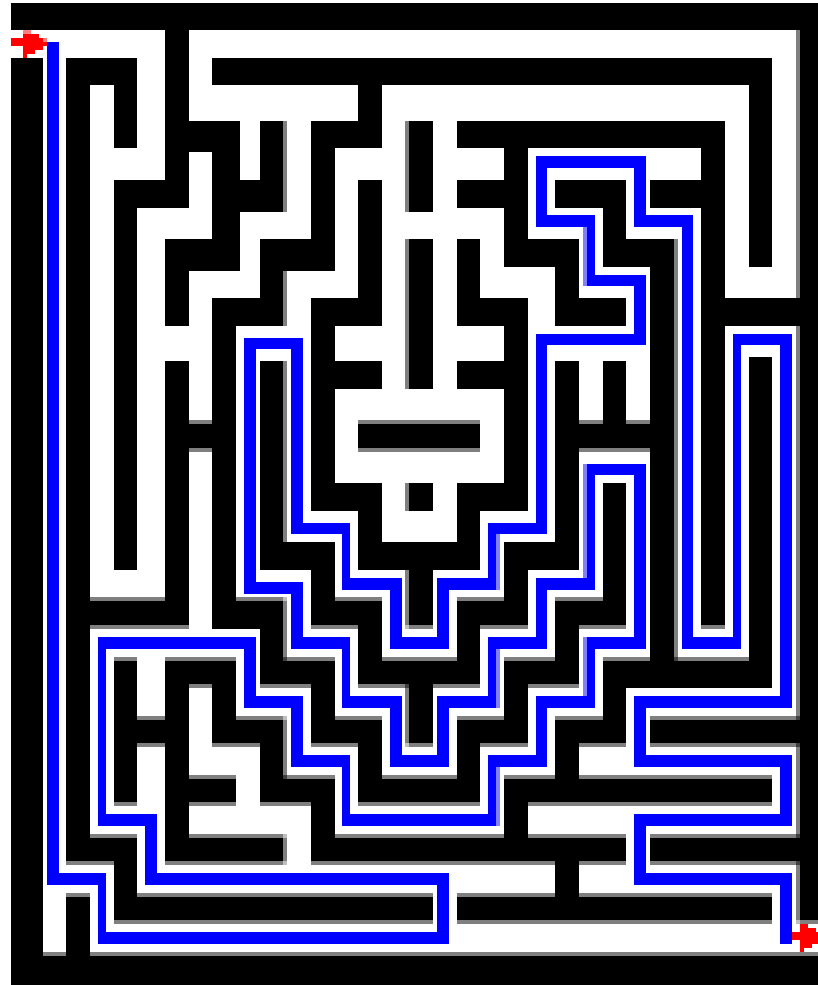
{Source:

<https://courses.cs.washington.edu/courses/cse326/03su/homework/hw3/dfs.html>}



Contoh runut-balik pada sebuah labirin. Runut-balik diperlihatkan dengan garis putus-putus.

Contoh lainnya:



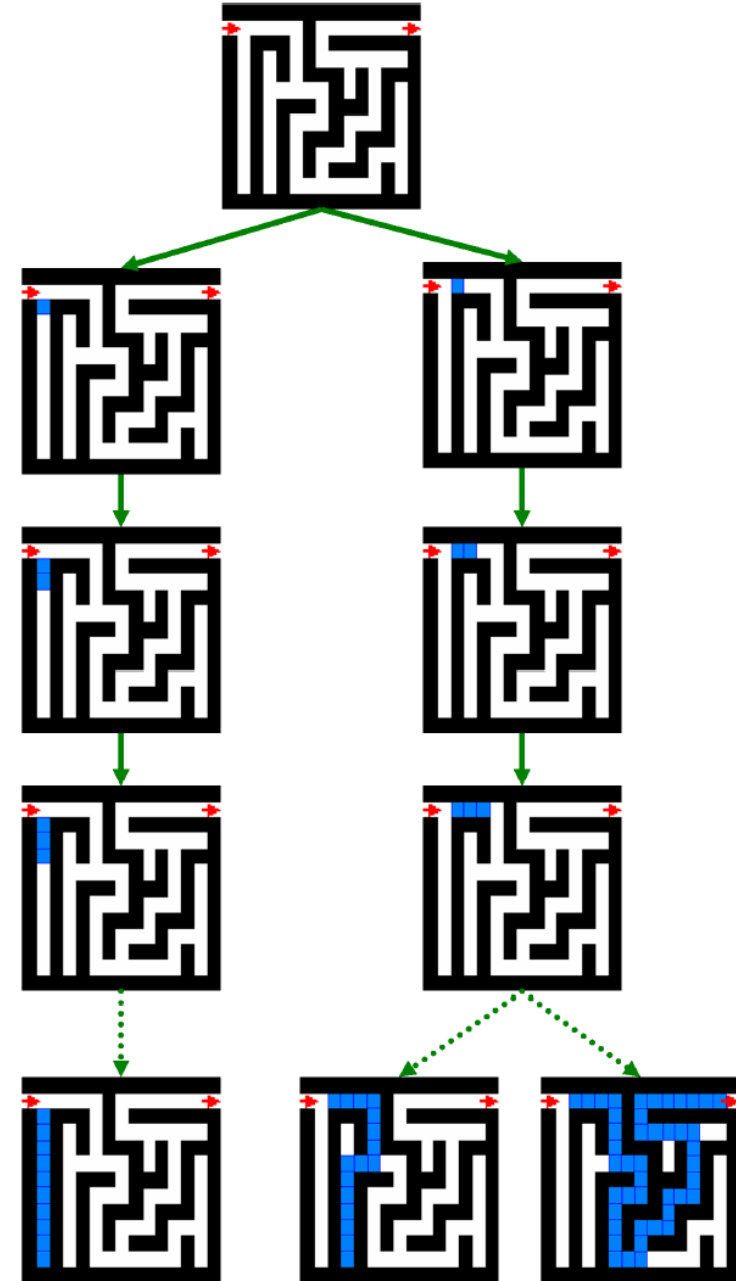
Jika kita menggambarkan sekuens pilihan yang kita lakukan, maka diagram berbentuk seperti pohon.

Simpul daun merupakan:

1. Titik *backtrack*, atau
2. Simpul *goal*

Pada titik *backtrack*, simpul tersebut menjadi mati (tidak bisa diekspansi lagi)

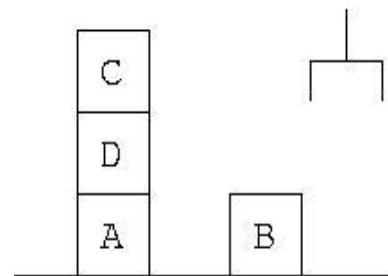
Aturan pembentukan simpul: DFS



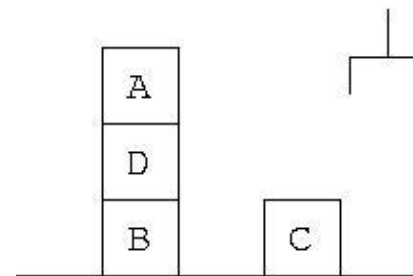
Block World Problem

(Soal UTS 2018)

- Terdapat beberapa buah balok berbentuk kubus yang ditempatkan di atas meja atau di atas balok yang lain sehingga membentuk sebuah konfigurasi. Sebuah robot yang memiliki lengan bercapit harus memindahkan balok-balok kubus tersebut sehingga membentuk konfigurasi lain dengan jumlah perpindahan yang minimum. Persyaratannya adalah hanya boleh memindahkan satu balok setiap kali ke atas balok lain atau ke atas meja. Gambarkan pohon ruang status pencarian solusi secara BFS dan DFS untuk *initial state* dan *goal state* di bawah ini. Setiap status digambarkan sebagai tumpukan balok kubus setelah pemindahan satu balok. Beri nomor setiap status sesuai aturan BFS dan DFS. Hitung berapa banyak status yang dibangkitkan sampai ditemukan *goal state*.



Initial state

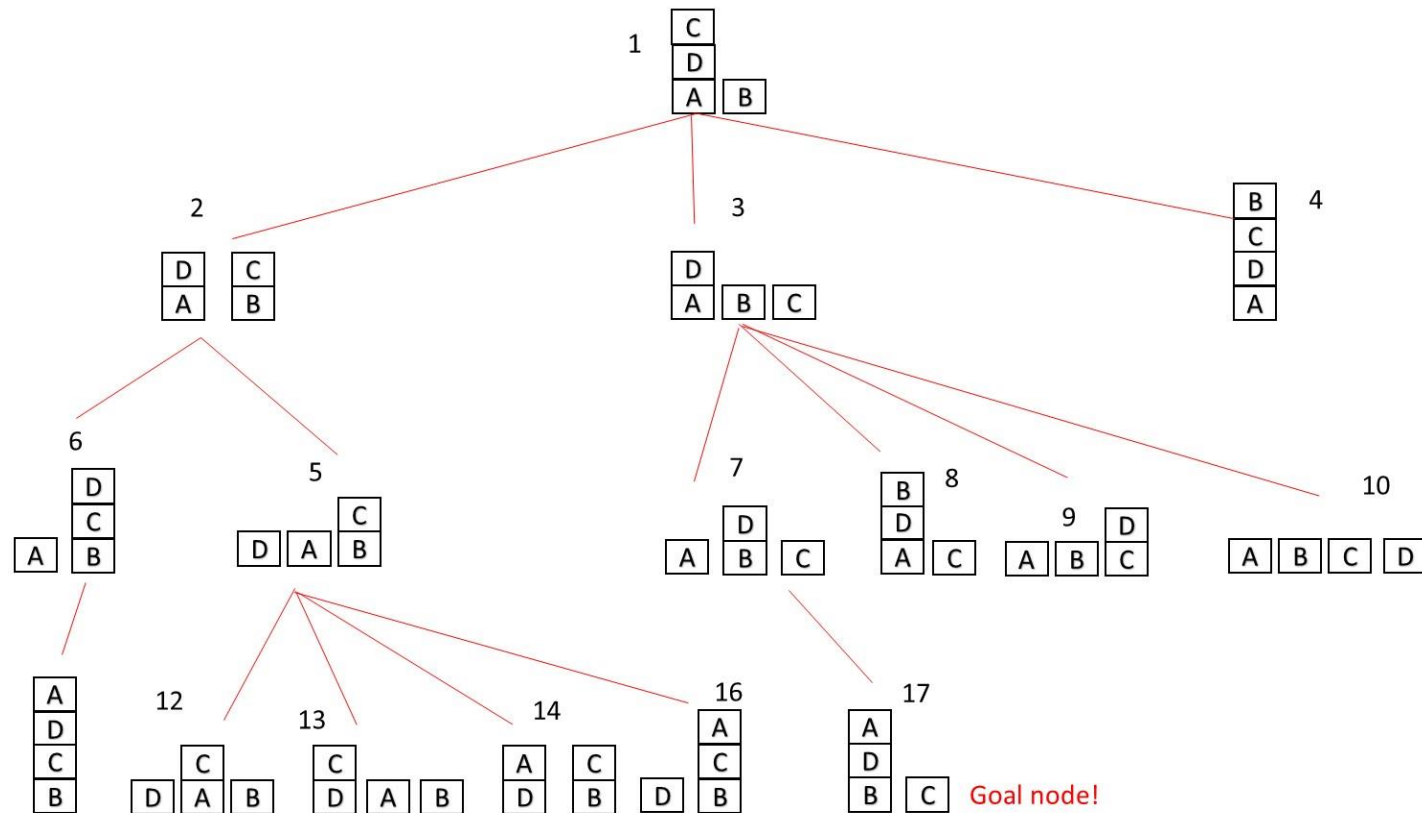


Goal state

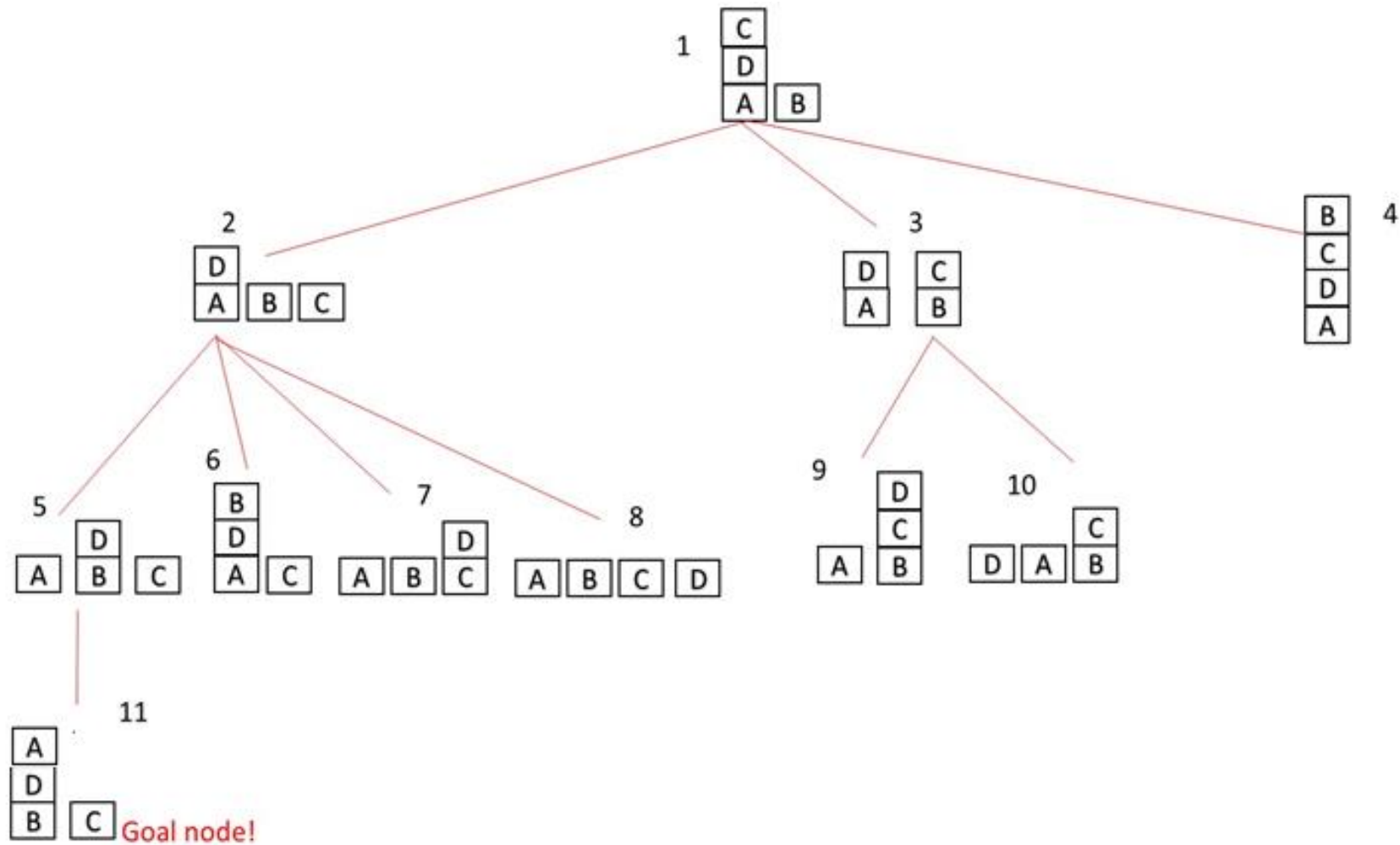
Penyelesaian:

(a) BFS

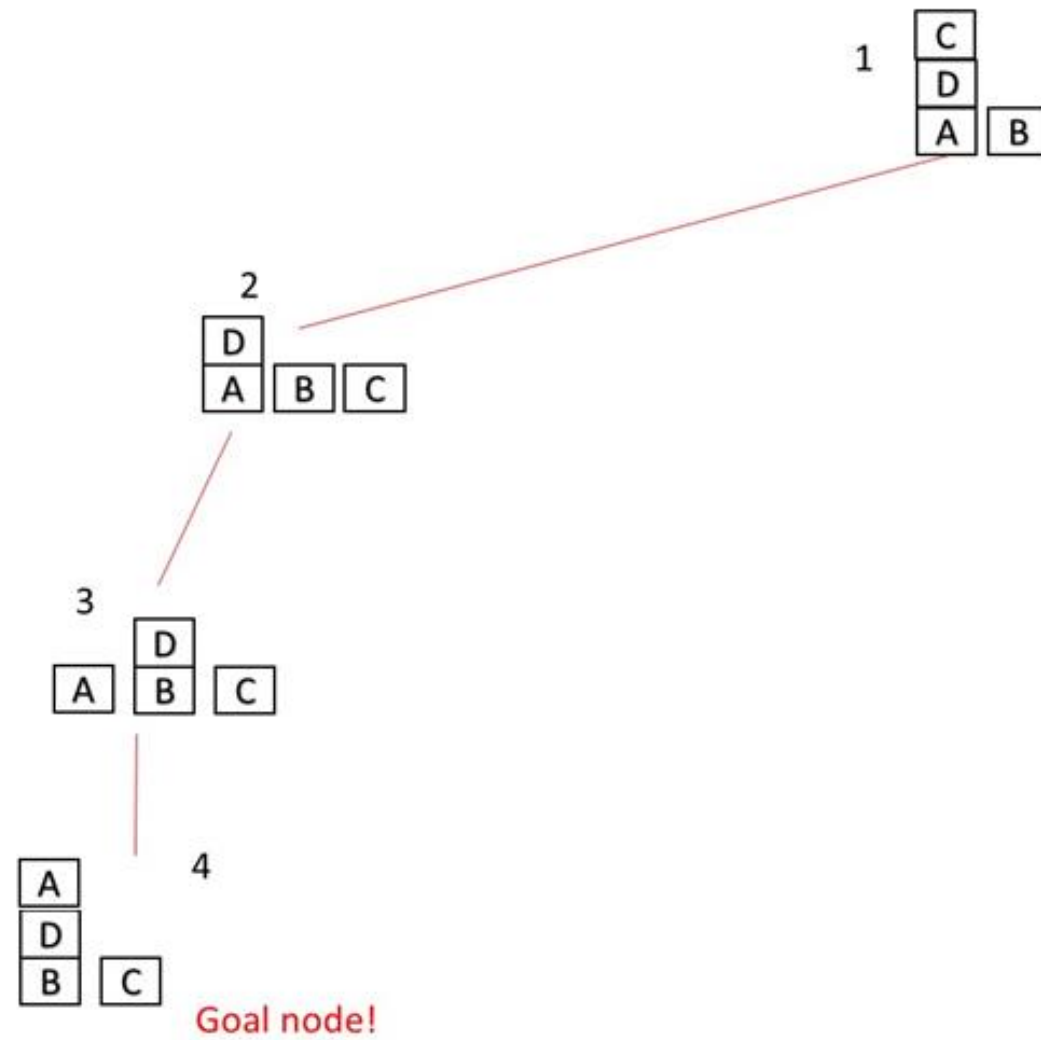
Salah satu kemungkinan solusi:



Kemungkinan lain:



(b) DFS



Persoalan Missionaris dan Kanibal

- Terdapat 3 misionaris dan 3 kanibal yang harus menyebrang ke sisi sungai menggunakan sebuah perahu. Perahu hanya boleh berisi penumpang maksimal 2 orang. Jumlah kanibal tidak boleh lebih banyak dari jumlah misionaris di salah satu sisi, jika tidak maka misionaris akan dimakan oleh kanibal.



Initial State



Goal State

- Bagaimana menyeberangkan keenam orang tadi sehingga semuanya selamat sampai di sisi sungai seberangnya? Selesaikan dengan BFS dan DFS

Press on Play

Please help the 3 cannibals
and the 3 missionaries to move
to the other side of the lake.

notice that: when there is on one side
more cannibals than
missionaries , they eat them.

play



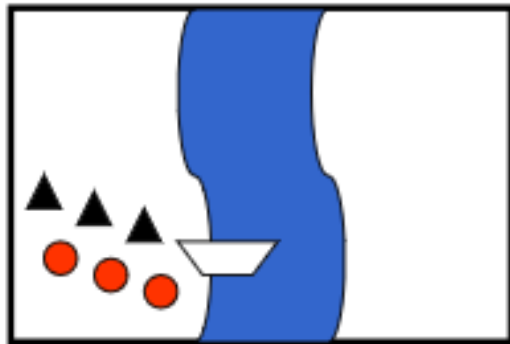
2

1

Missionaries and Cannibals: Initial State and Actions

- initial state:

- all missionaries, all cannibals, and the boat are on the left bank

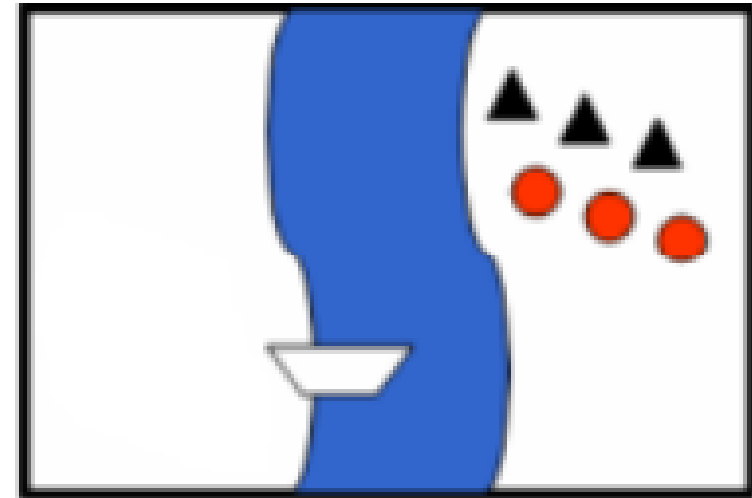


- 5 possible actions:

- one missionary crossing
- one cannibal crossing
- two missionaries crossing
- two cannibals crossing
- one missionary and one cannibal crossing

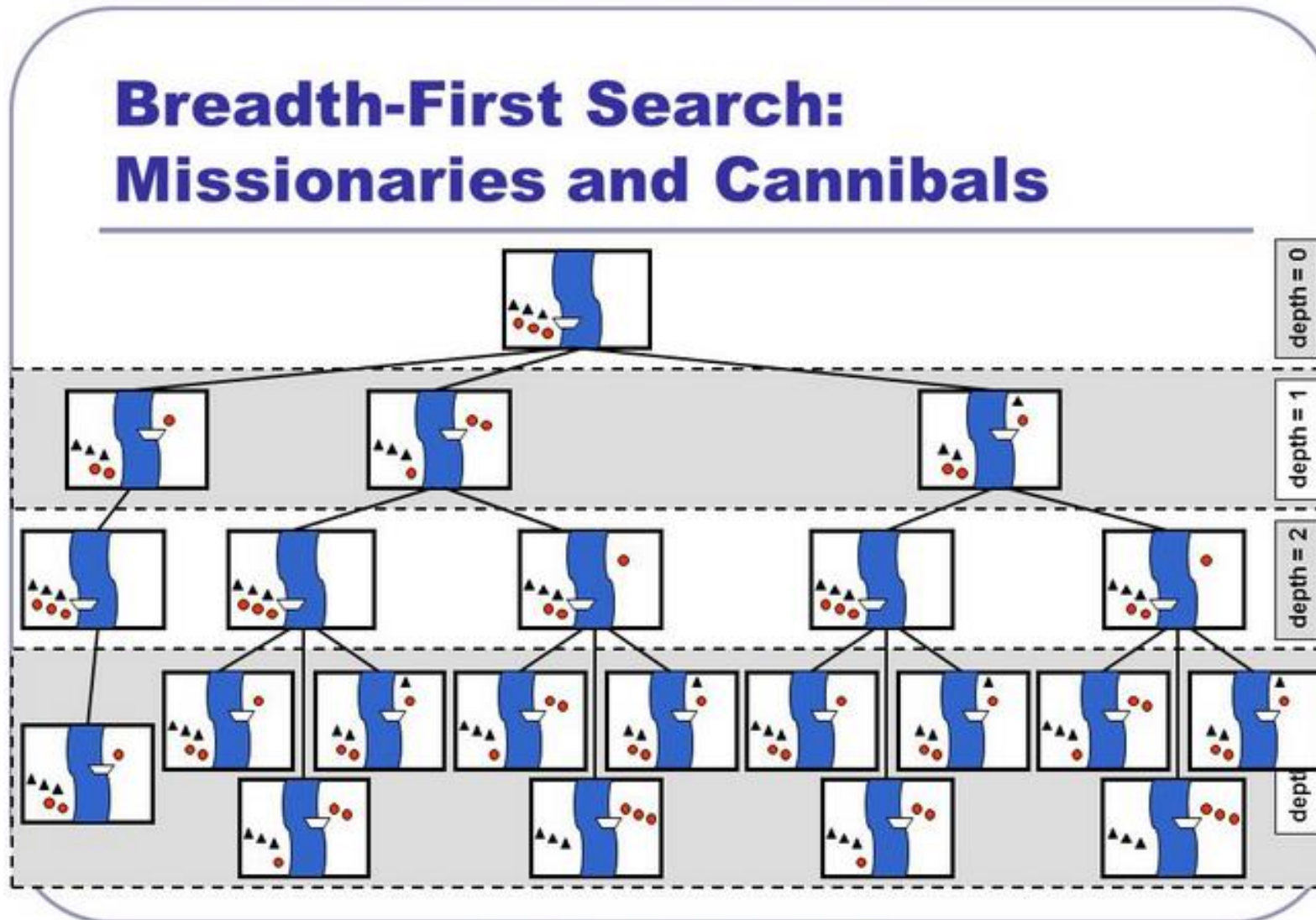


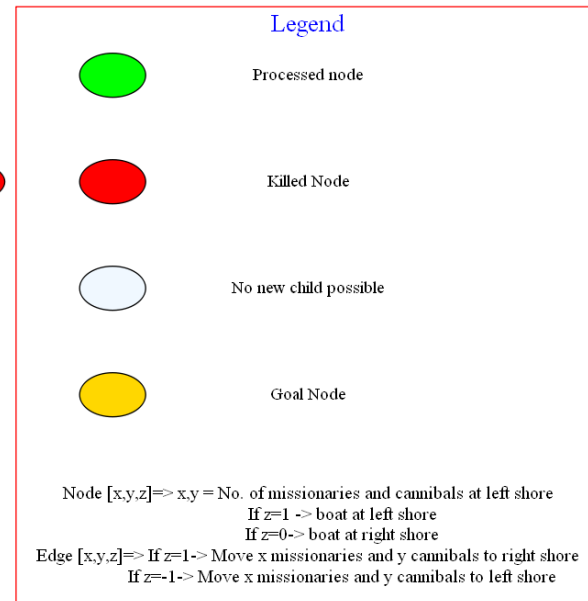
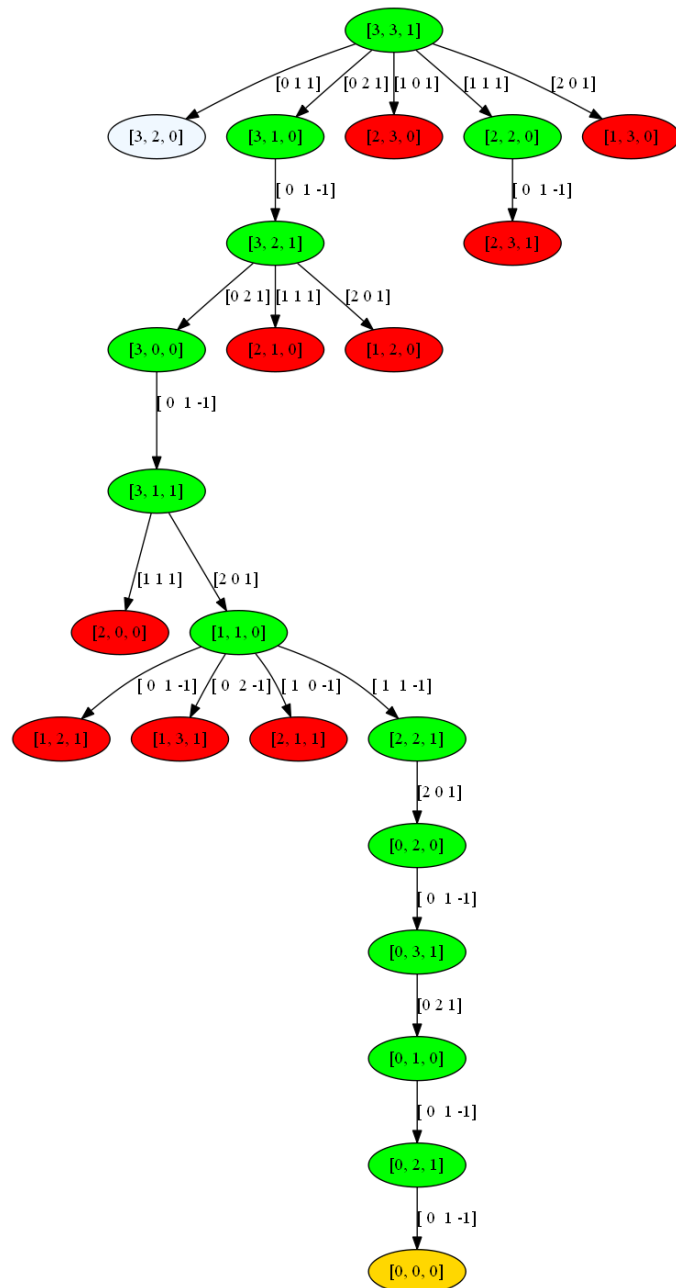
Initial State



Final State

Sebagian pohon ruang status dengan BFS





Sumber: <https://github.com/ayushkumarshah/Missionaries-and-Cannibals-State-Space-Tree?tab=readme-ov-file>

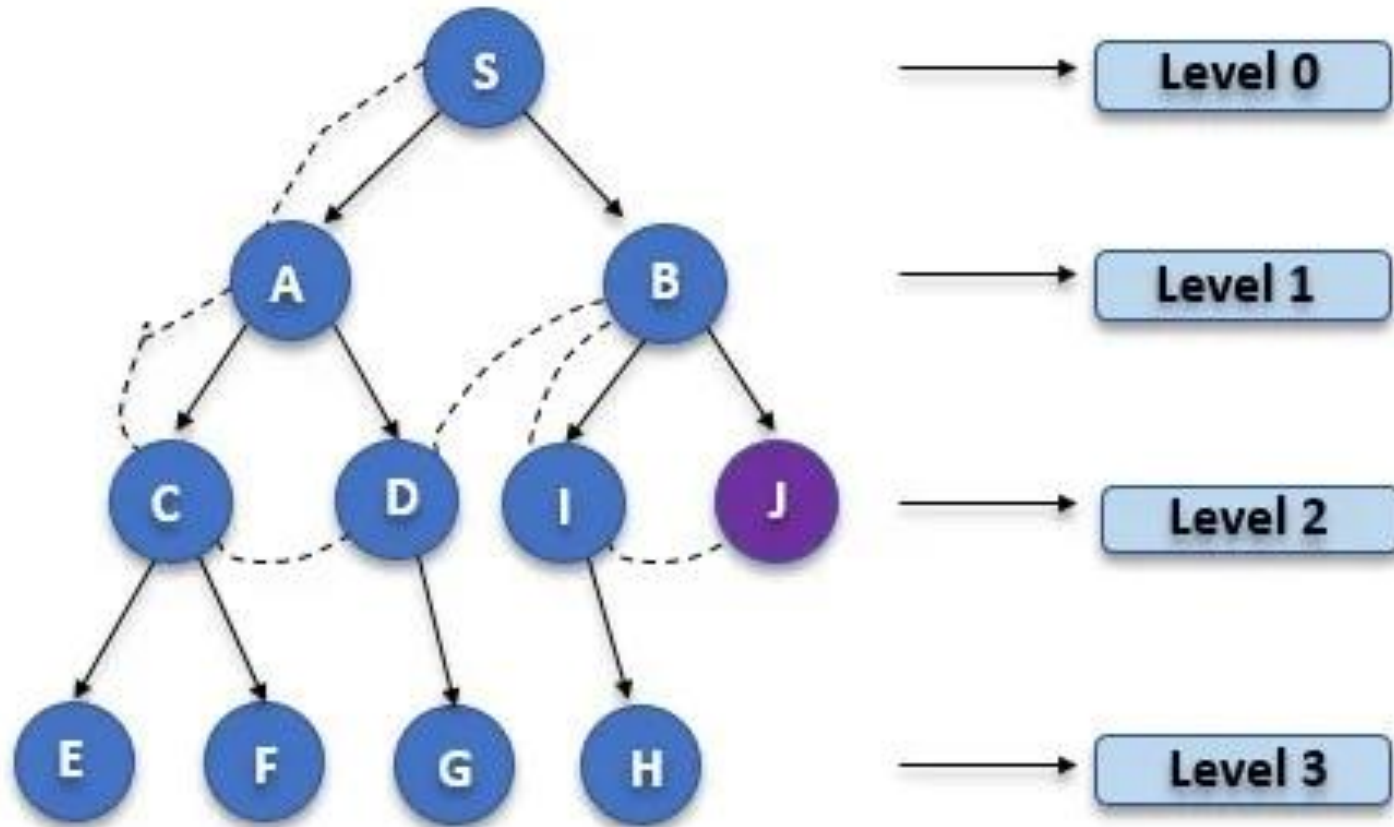
Algoritma Pencarian Lainnya

- *Depth-limited search* (DLS)
- *Iterative deepening search* (IDS)

Depth-Limited Search

- BFS: dijamin menemukan path dgn langkah minimum tapi membutuhkan ruang status yang besar
- DFS: efisien, tetapi tidak ada jaminan solusi dgn langkah minimum
 - DFS dapat memilih langkah yang salah, sehingga path panjang bahkan infinite. Pemilihan langkah sangat penting
- Salah satu solusi: DFS-limited search
 - DFS dengan pembatasan kedalaman sampai *limit*
 - Simpul pada level *limit* dianggap tidak memiliki successor
 - Masalah: penentuan batas level ($\geq$ *shallowest goal*)

Depth Limited Search



www.educba.com

DLS Algorithm

Function DLS (problem, limit)

→ Rec_DLS(make_node(init_state), problem, limit)

Function Rec_DLS (node, problem, limit)

if isGoal(node) then → solution(node)

else if depth(node)=limit then → cutoff

else

for each successor in Expand(node, problem) do

result ← Rec_DLS(successor, problem, limit)

if result=cutoff then cutoff_occured ← true

else if result≠failure then → result

if cutoff_occured then → cutoff

else → failure

Bagaimana properti DLS?

- *Completeness?*
 - Tidak
- *Optimality?*
 - Tidak
- Kompleksitas waktu:
 - $O(b')$
- Kompleksitas ruang:
 - $O(bl)$

Contoh DLS

Contoh. Sebuah bidak (pion) bergerak di dalam sebuah matriks pada Gambar 6.11. Bidak dapat memasuki elemen matriks mana saja pada baris paling atas. Dari elemen matriks yang berisi 0, bidak dapat bergerak ke bawah jika elemen matriks di bawahnya berisi 0; atau berpindah horizontal (kiri atau kanan) jika elemen di bawahnya berisi 1. Bila bidak berada pada elemen yang berisi 1, ia tidak dapat bergerak kemanapun. Tujuan permainan ini adalah mencapai elemen matriks yang mengandung 0 pada baris paling bawah.

DOWN pindahkan bidak satu posisi ke bawah

LEFT pindahkan bidak satu posisi ke kiri

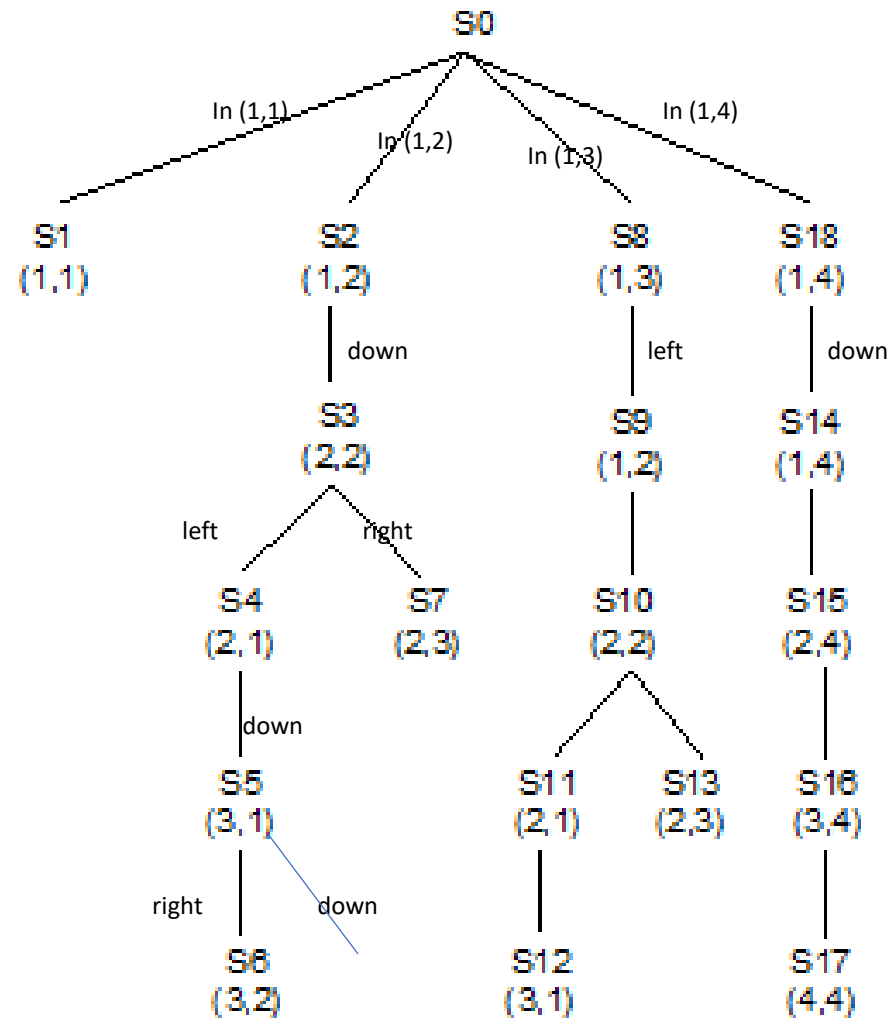
RIGHT pindahkan bidak satu posisi ke kanan

Batas kedalaman maksimum pohon ruang status diandaikan 5.

	1	2	3	4
1	1	0	0	0
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0

Gambar 6.11 Matriks bidak

Pohon Ruang Status



	1	2	3	4
1	1	0	0	0
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0

Gambar 6.11 Matriks bidak

Iterative Deepening Search (IDS)

- IDS: melakukan serangkaian DFS, dengan peningkatan nilai kedalaman-cutoff, sampai solusi ditemukan
- Asumsi: simpul sebagian besar ada di level bawah, sehingga tidak menjadi persoalan ketika simpul pada level-level atas dibangkitkan berulang kali

Depth $\leftarrow$ 0

Iterate

 result $\leftarrow$ DLS (problem, depth)

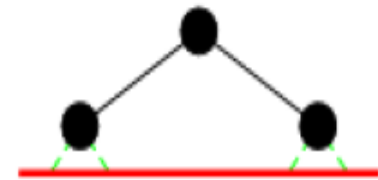
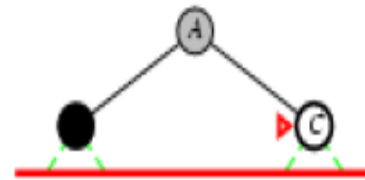
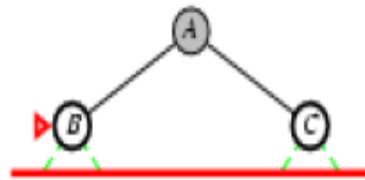
stop: result $\neq$ cutoff

 depth $\leftarrow$ depth+1

$\rightarrow$ result

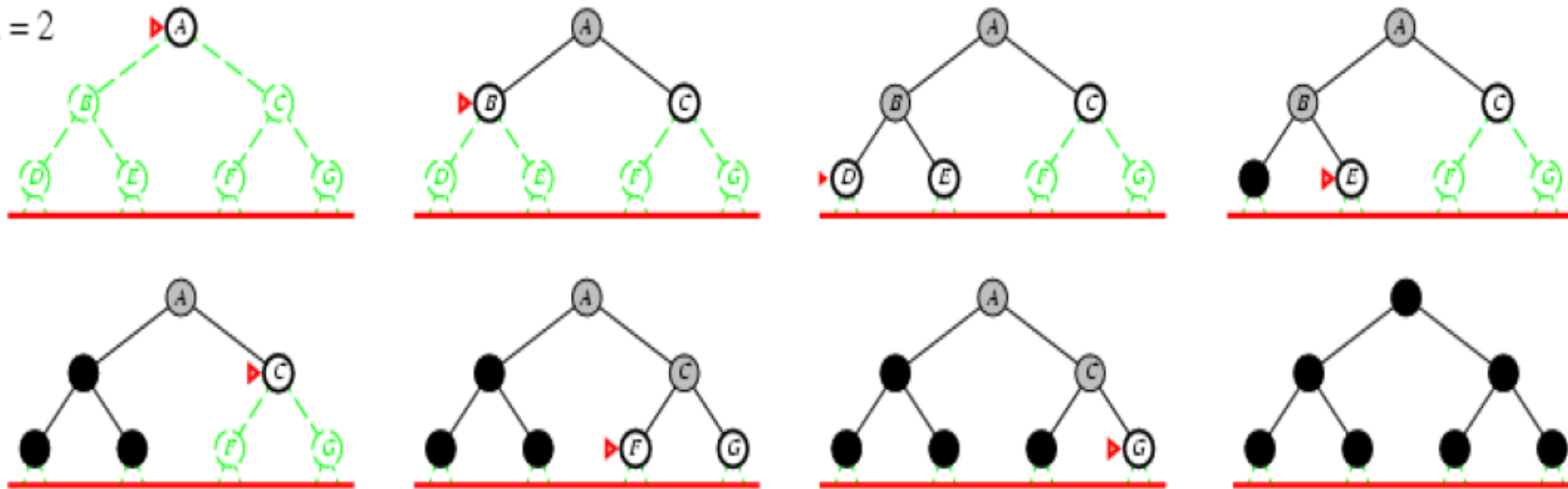
IDS dengan $d=1$

Limit = 1



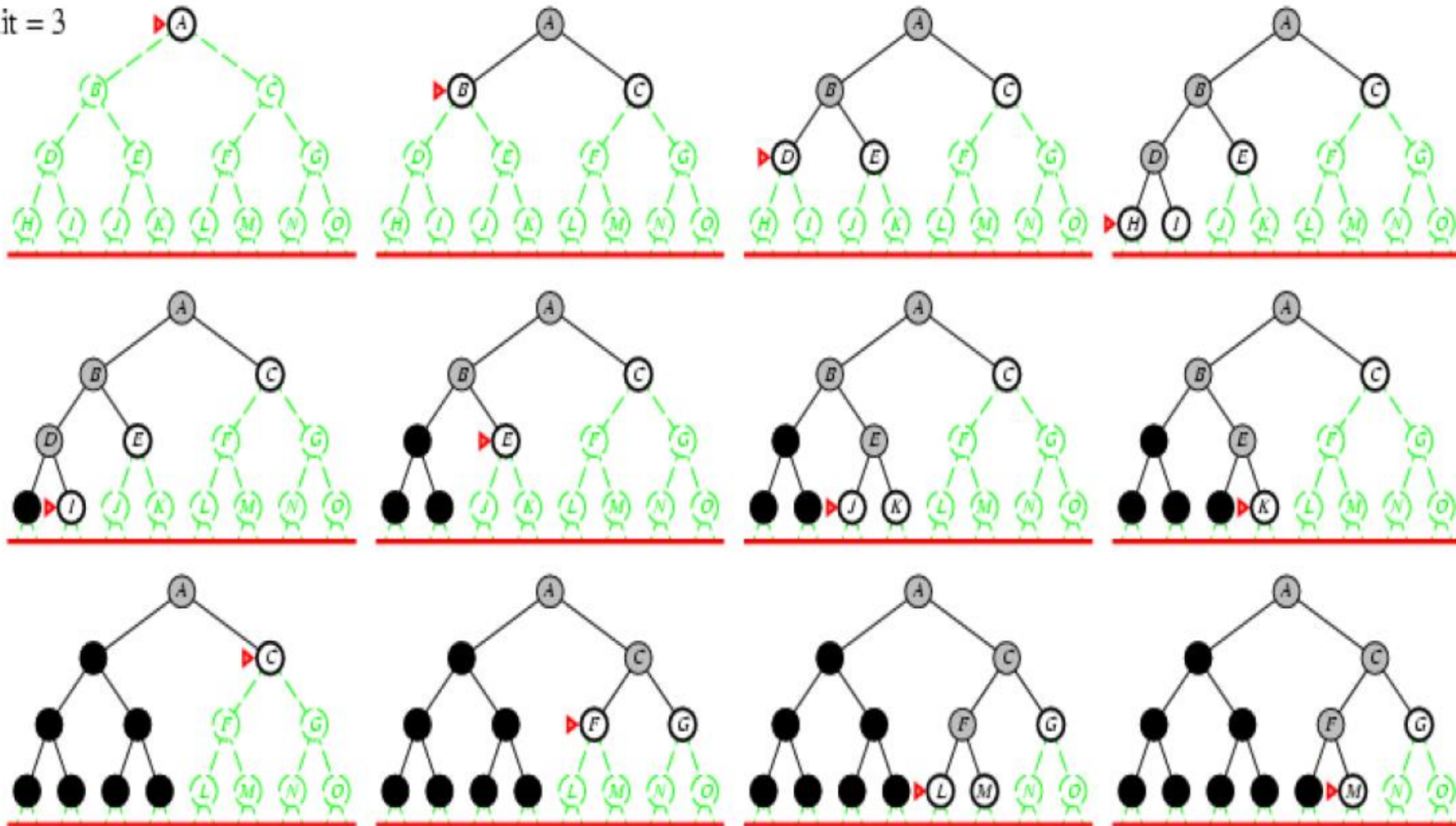
IDS dengan $d=2$

Limit = 2

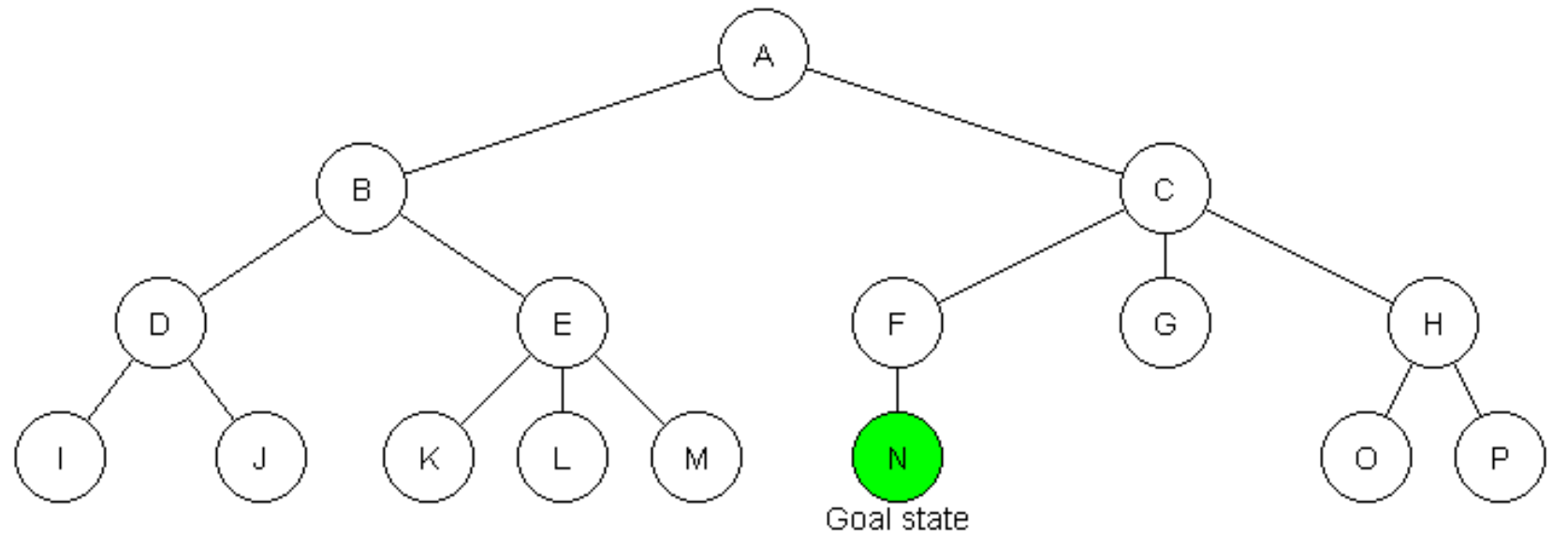


IDS dengan $d=3$

Limit = 3



IDS (animasi)

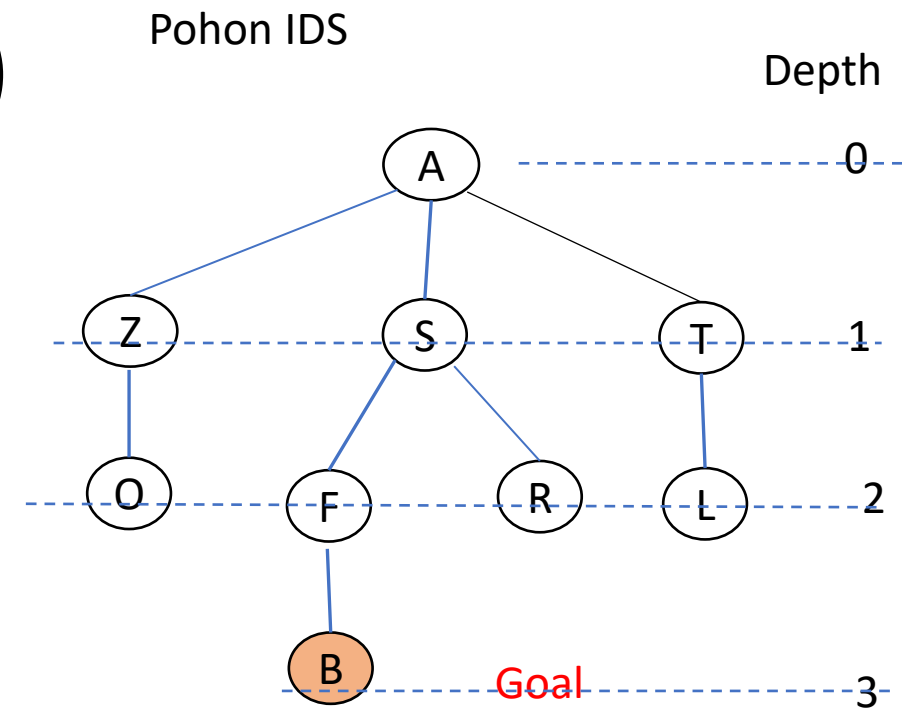
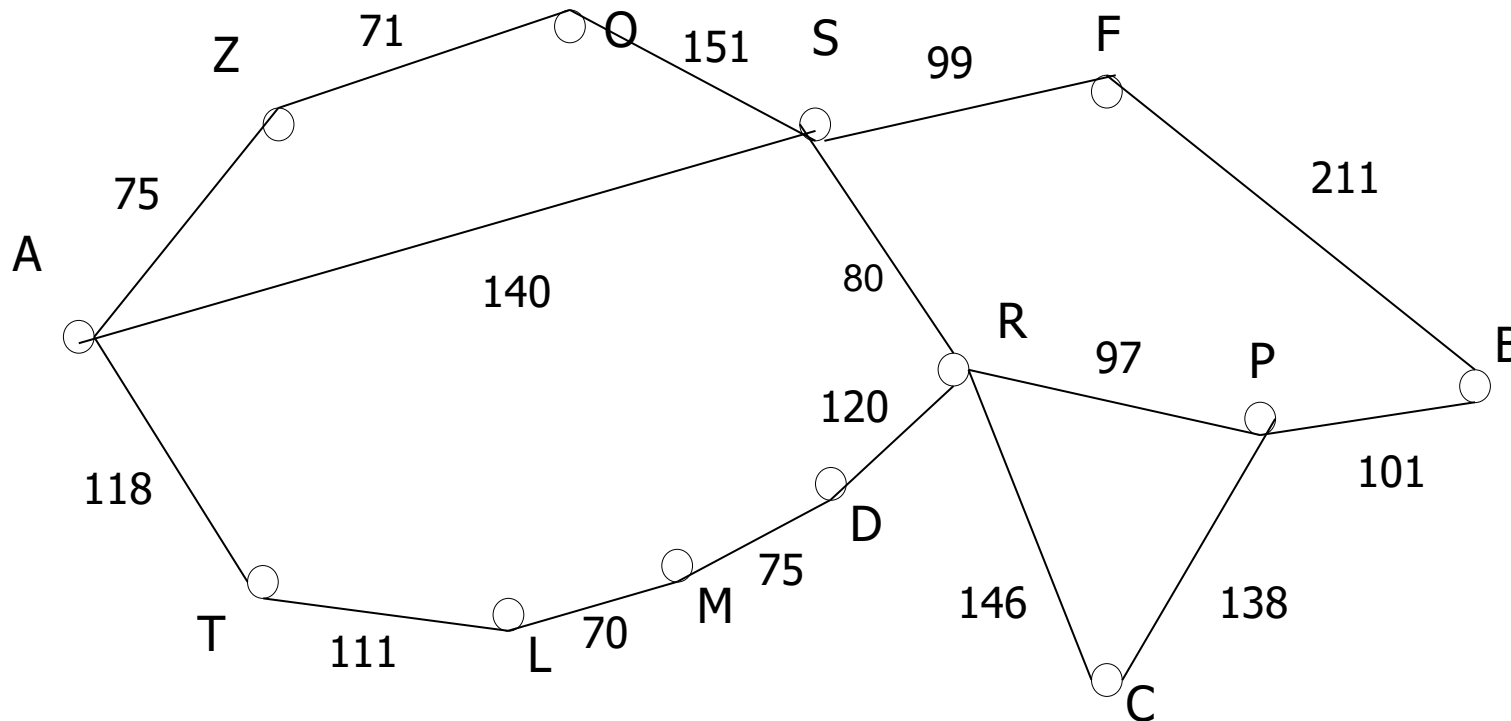


how2examples.com

Bagaimana property dari IDS?

- Completeness?
 - Ya, jika b terbatas
- Optimality?
 - Ya, jika langkah = biaya
- Kompleksitas waktu:
 - $O(b^d)$
- Kompleksitas ruang:
 - $O(bd)$

Iterative Deepening Search (IDS)



Depth=0: A: cutoff

Depth=1: A $\rightarrow$ Z_A, S_A, T_A $\rightarrow$ Z_A : cutoff, S_A : cutoff, T_A : cutoff

Depth=2: A $\rightarrow$ Z_A, S_A, T_A $\rightarrow$ O_{AZ}, S_A, T_A $\rightarrow$ O_{AZ} : cutoff $\rightarrow$ F_{AS}, R_{AS}, T_A $\rightarrow$ F_{AS} : cutoff $\rightarrow$ R_{AS} : cutoff $\rightarrow$ L_{AT}
 $\rightarrow$ L_{AT} : cutoff

Depth=3: A $\rightarrow$ Z_A, S_A, T_A $\rightarrow$ O_{AZ}, S_A, T_A $\rightarrow$ S_{AZO}, S_A, T_A $\rightarrow$ S_{AZO} : cutoff $\rightarrow$ F_{AS}, R_{AS}, T_A $\rightarrow$ B_{ASF}, R_{AS}, T_A $\rightarrow$ B_{ASF}

Stop: B=goal, path: A $\rightarrow$ S $\rightarrow$ F $\rightarrow$ B, path-cost = 450

TAMAT