

# TEKNIK SEARCHING

SEMESTER II  
PERTEMUAN KE – 5,6  
DOSEN : ELISAWATI,M.KOM

TIK :

- ▶ Mahasiswa mampu memahami konsep dasar penyelesaian permasalahan dengan metode search
- ▶ Mahasiswa mampu menggunakan metode uninformed search dalam pencarian solusi permasalahan
- ▶ Mahasiswa mampu menggunakan metode informed search dalam pencarian solusi permasalahan

# Searching (Pencarian)

Teknik penyelesaian masalah dg cara merepresentasikan masalah ke dalam state dan ruang masalah serta menggunakan strategi pencarian untuk menemukan solusi.

## Langkah–langkah dalam teknik searching :

- ▶ Mendefinisikan ruang masalah/ruang keadaan untuk suatu masalah yg dihadapi.
- ▶ Mendefinisikan aturan produksi yang digunakan untuk mengubah suatu “state” ke “state” lainnya.
- ▶ Aturan produksi adalah operasi yg mengubah suatu state ke state lainnya.
- ▶ Memilih metode pencarian yg tepat sehingga dapat menemukan solusi terbaik dengan usaha yang minimal.

# Ruang Masalah atau Ruang Keadaan

- ▶ Ruang masalah dpt digambarkan sebagai himpunan keadaan (state) atau bisa juga himpunan rute dari keadaan awal (initial state) menuju keadaan tujuan (goal state).
- ▶ Masalah utama dalam membangun sistem berbasis AI : bagaimana mengkonversi situasi yg diberikan ke dalam situasi lain yg diinginkan menggunakan sekumpulan operasi tertentu.

## Contoh :

Menara Hanoi adalah sebuah permainan yang terdiri dari tiga tiang dan sejumlah cakram dengan ukuran yang berbeda-beda yang bisa dimasukkan ke tiang mana saja. Permainan dimulai dengan cakram-cakram yang tertumpuk rapi berurutan berdasarkan ukurannya dalam salah satu tiang, cakram terkecil diletakkan di urutan teratas sehingga membentuk kerucut. Tujuan dari teka-teki ini adalah untuk memindahkan seluruh tumpukan ke tiang yang lain, mengikuti aturan berikut :

- ▶ Hanya satu cakram yang boleh dipindahkan dalam satu waktu
- ▶ Setiap perpindahan berupa pengambilan cakram teratas dari satu tiang dan memasukkannya ke tiang lain, di atas cakram lain yang mungkin sudah ada di tiang tersebut.
- ▶ Tidak boleh meletakkan cakram di atas cakram lain yang lebih kecil

## Jawaban :

- ▶ Dekscripsi

Ruang keadaan untuk masalah ini adalah setiap keadaan di mana posisi cakram yang lebih besar selalu berada di bawah cakram yang lebih kecil.

- ▶ Keadaan Awal

Tumpukan cakram berada di tiang kiri

- ▶ Tujuan

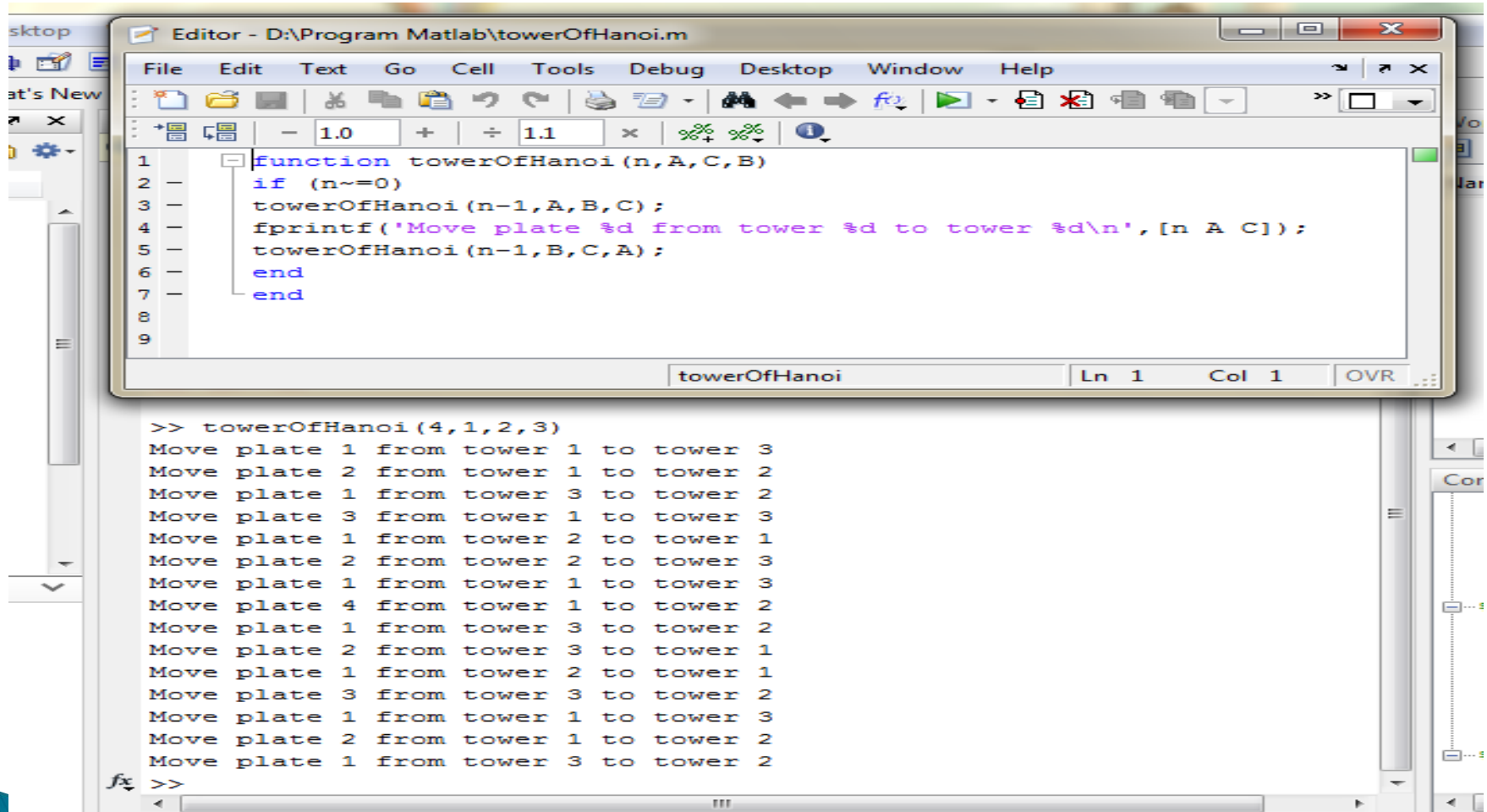
Tumpukan cakram berada di tiang kanan



# Keadaan Awal ke Keadaan Tujuan



# Program Matlab (Hanoi)



The screenshot displays the MATLAB environment. The Editor window, titled 'Editor - D:\Program Matlab\towerOfHanoi.m', contains the following code:

```
1 function towerOfHanoi(n,A,C,B)
2     if (n~=0)
3         towerOfHanoi(n-1,A,B,C);
4         fprintf('Move plate %d from tower %d to tower %d\n',[n A C]);
5         towerOfHanoi(n-1,B,C,A);
6     end
7 end
```

The Command Window shows the execution of the function with 4 disks and 3 towers, resulting in 15 moves:

```
>> towerOfHanoi(4,1,2,3)
Move plate 1 from tower 1 to tower 3
Move plate 2 from tower 1 to tower 2
Move plate 1 from tower 3 to tower 2
Move plate 3 from tower 1 to tower 3
Move plate 1 from tower 2 to tower 1
Move plate 2 from tower 2 to tower 3
Move plate 1 from tower 1 to tower 3
Move plate 4 from tower 1 to tower 2
Move plate 1 from tower 3 to tower 2
Move plate 2 from tower 3 to tower 1
Move plate 1 from tower 2 to tower 1
Move plate 3 from tower 3 to tower 2
Move plate 1 from tower 1 to tower 3
Move plate 2 from tower 1 to tower 2
Move plate 1 from tower 3 to tower 2
```

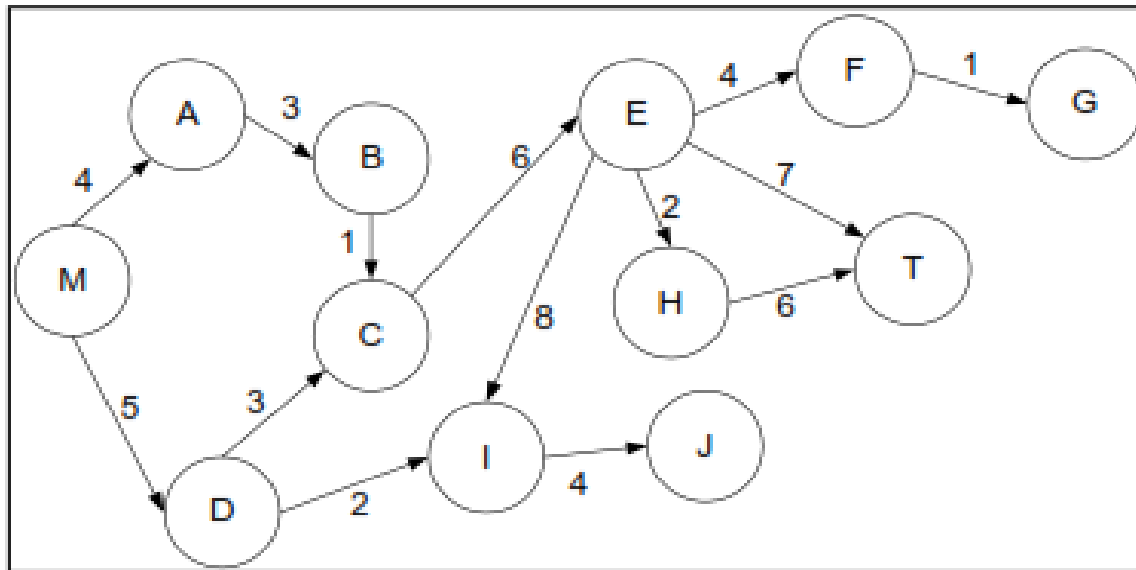
Ada beberapa cara utk merepresentasikan ruang masalah/ruang keadaan :

- ▶ **Graph Keadaan**

terdiri dari node-node yg menunjukkan keadaan yaitu keadaan awal dan keadaan baru yg akan dicapai dengan menggunakan operator. Node-node graph saling dihubungkan dg menggunakan “arc” (busur) yg diberi panah utk menunjukkan arah dari suatu keadaan ke keadaan berikutnya.

## Contoh Graph Keadaan

- ▶ Misalnya node M adalah keadaan awal, dan node T adalah tujuan, maka dari gambar graph keadaan di atas ditemukan beberapa lintasan yang menghubungkan dari M ke T



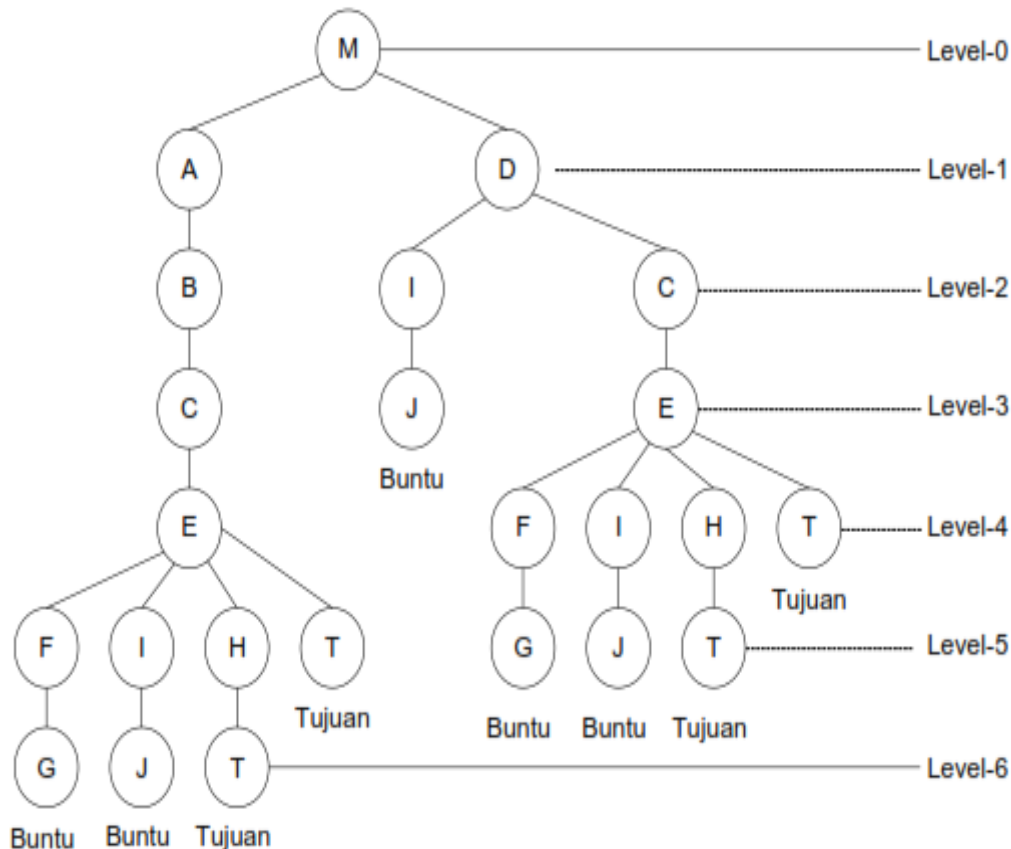
Graph berarah dengan satu tujuan (T)	Graph berarah yang menemui jalan buntu	Graph dengan siklus
<ul style="list-style-type: none"> <li>• M-A-B-C-E-T</li> <li>• M-A-B-C-E-H-T</li> <li>• M-D-C-E-T</li> <li>• M-D-C-E-H-T</li> </ul>	<ul style="list-style-type: none"> <li>• M-A-B-C-E-F-G</li> <li>• M-A-B-C-E-I-J</li> <li>• M-D-C-E-F-G</li> <li>• M-D-C-E-I-J</li> <li>• M-D-I-J</li> </ul>	D-E-C-E-I-D

- ▶ Bentuk Graph seperti ini sulit untuk direpresentasikan dalam suatu software, karena sangat memungkinkan adanya siklus node yang akan selalu berulang.

## ► **Pohon Pelacakan**

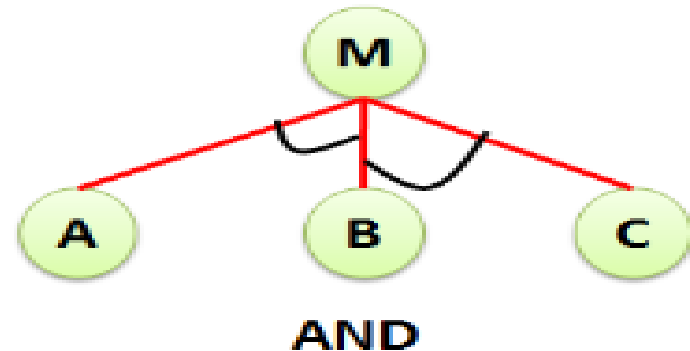
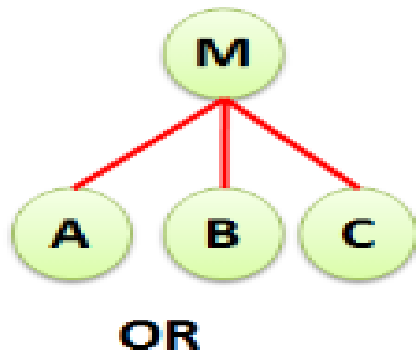
Pohon pelacakan berguna untuk menghindari adanya proses pelacakan suatu node secara berulang. Keadaan digambarkan secara Hierarkis.

# Contoh Pohon pelacakan (berdasar pada contoh soal pada Graph Keadaan)



- ▶ Node Akar → node pada level 0
- ▶ Node Cabang → disebut anak atau node perantara
- ▶ Node Daun → node yang tidak memiliki “anak” atau “cabang”, biasanya berupa akhir dari pencarian yang berupa : Tujuan (Goal) maupun Jalan Buntu (Dead End).
- ▶ Pada pohon pelacakan di atas, tidak ada siklus yang berulang karena setiap node tidak diperbolehkan punya cabang kembali ke node dengan level yang lebih rendah.

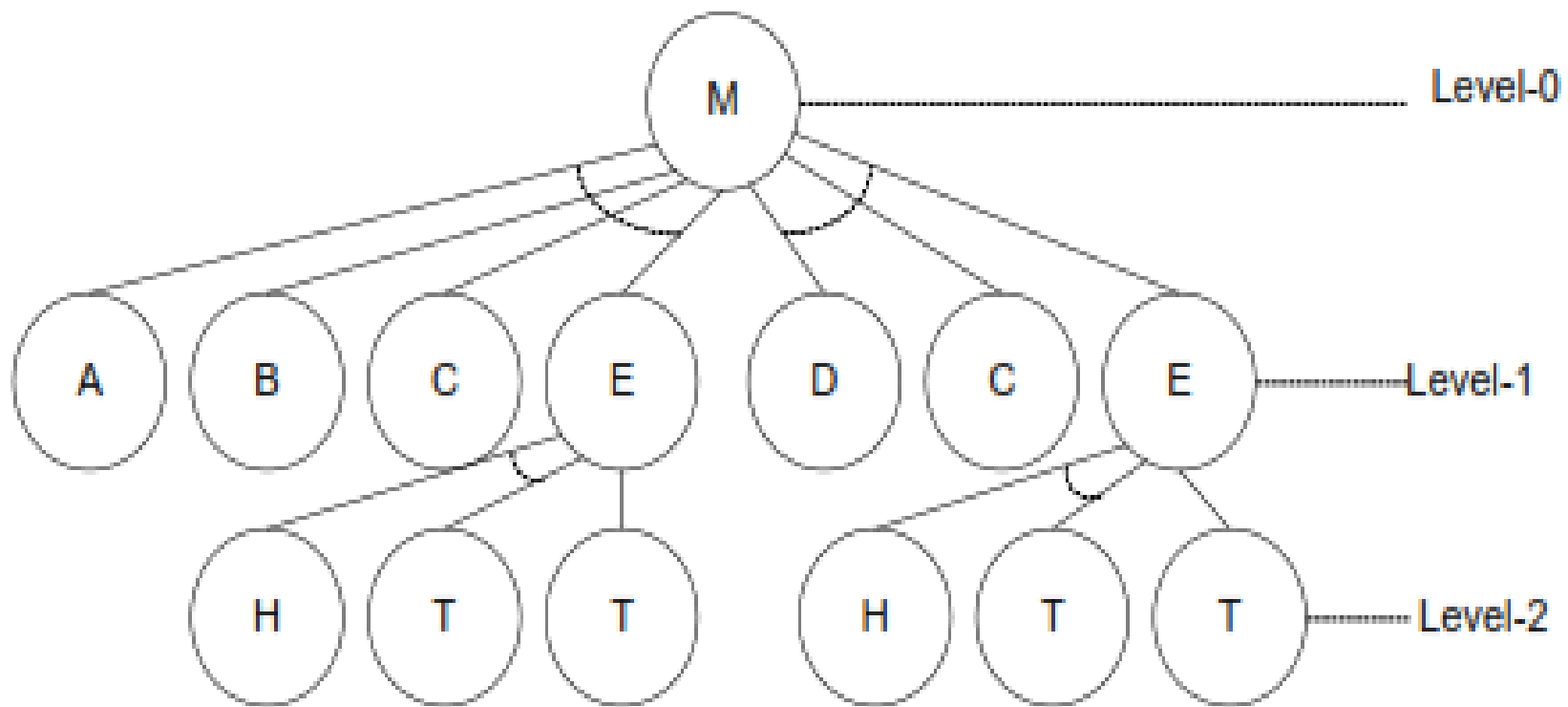
## ► Pohon AND-OR



- OR: masalah M dapat terpecahkan jika salah satu dari subgoal A,B,C terpecahkan.
- AND: masalah dari M terpecahkan jika semua subgoal A,B,C sudah terpecahkan.



- ▶ Contoh soal Dari gambar pohon sebelumnya, buatlah Pohon AND/OR sehingga penyelesaian dapat dipersingkat!



## Contoh Soal Penerapan Ruang Masalah/Ruang Keadaan : WATER JUG PROBLEM/TEKO AIR



**Note: Air Tidak Terbatas**

- ▶ Diketahui dua buah jerigen, yang satu ukuran 4 galon dan yang lain 3 galon. Kedua gelas tidak memiliki skala ukuran. Terdapat pompa yang dapat digunakan untuk mengisi jerigen dengan air. Bagaimana anda mendapatkan *tepat 2 galon air di dalam jerigen ukuran 4 galon*?

## Penyelesaian :

- ▶ *Initial state:*

Diketahui dua buah ember masing-masing berkapasitas 3 gallon dan 4 gallon, dan sebuah pompa air.

- ▶ *Goal state:*

Isi ember yang berkapasitas 4 gallon dengan 2 gallon air!

- ▶ Solusi: Buat asumsi dengan:

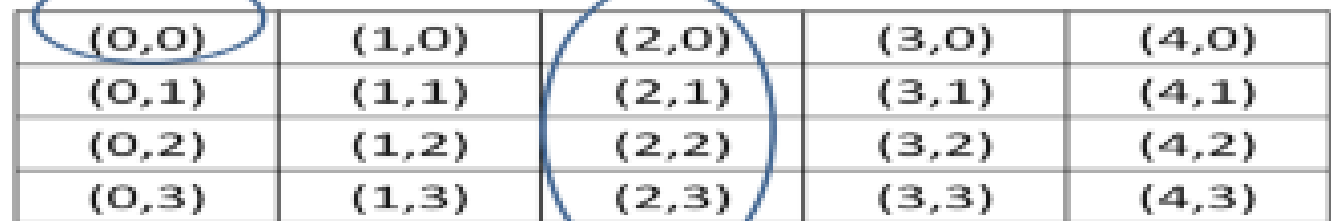
X : ember berkapasitas 4 gallon

Y : ember berkapasitas 3 gallon

- ▶ *Production Rules atau Aturan produksi:*  
Sistem Produksi/ *Production System* terdiri dari:
  - ▶ Sekumpulan Aturan (*a set of rules*)
  - ▶ Knowledge Base /Data Base
  - ▶ Sebuah strategi pengontrol (*Control Strategy*)
  - ▶ Urutan yang dipakai (*a rule applier*)

Keadaan  
Awal

GOAL



(0,0)	(1,0)	(2,0)	(3,0)	(4,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)

## Aturan produksi :

Aturan Ke-	Kondisi Awal	Kondisi Tujuan	Aturan
1	$(x,y)$ dng $x < 4$	$(4,y)$	Isi teko A sampai penuh
2	$(x,y)$ dng $y < 3$	$(x,3)$	Isi teko B sampai penuh
3	$(x,y)$ dng $x > 0$	$(x-d,y)$	Tuangkan sebagian air keluar dari teko A
4	$(x,y)$ dng $y > 0$	$(x,y-d)$	Tuangkan sebagian air keluar dari teko B
5	$(x,y)$ dng $x > 0$	$(0,y)$	Kosongkan teko A dengan membuang airnya ke tanah
6	$(x,y)$ dng $y > 0$	$(x,0)$	Kosongkan teko B dengan membuang airnya ke tanah
7	$(x,y)$ dimana $x+y \geq 4$ dan $y > 0$	$(4,y-(4-x))$	Tuangkan air dari teko B ke teko A sampai teko A penuh
8	$(x,y)$ dimana $x+y \geq 3$ dan $x > 0$	$(x-(3-y),3)$	Tuangkan air dari teko A ke teko B sampai teko B penuh
9	$(x,y)$ dimana $x+y \leq 4$ dan $y > 0$	$(x+y,0)$	Tuangkan seluruh air dari teko B ke teko A
10	$(x,y)$ dimana $x+y \leq 3$ dan $x > 0$	$(0,x+y)$	Tuangkan seluruh air dari teko A ke teko B
11	$(0,2)$	$(2,0)$	Tuangkan 2 galon air dari teko B ke teko A
12	$(2,y)$	$(0,y)$	Kosongkan 2 galon air di teko A dengan membuang airnya ke tanah

Solusi dari permasalahan tersebut :

Kondisi Awal	Kondisi Tujuan	Aturan yang dipakai
(0,0)	(0,3)	2
(0,3)	(3,0)	9
(3,0)	(3,3)	2
(3,3)	(4,2)	7
(4,2)	(0,2)	5
(0,2)	(2,0)	9

# METODE PENCARIAN

## BLIND SEARCH

- ▶ Dikatakan “blind” atau “buta” karena memang tidak ada informasi awal yang digunakan dalam proses pencarian.
- ▶ Blind Search meliputi :
  - a. Breadth First Search (BFS)
  - b. Uniform Cost Search (UCS)
  - c. Depth First Search (DFS)
  - d. Depth Limited Search (DLS)
  - e. Iterative Deepening Search (IDS)
  - f. Bi-Directional Search (BDS)
- ▶ Dari ke-enam macam pencarian buta di atas, yang sering dibahas adalah “Breadth First Search (BFS)” dan “Depth First Search (DFS)”.

## ► **BREADTH FIRST SEARCH (BFS)**

Breadth First Search (BFS) merupakan metode pencarian yang bertujuan untuk memperluas dan memeriksa semua simpul pada graph atau urutan kombinasi dengan pencarian secara sistematis melalui setiap solusi.

BFS melakukan pencarian secara mendalam pada keseluruhan graph atau urutan tanpa memperhatikan tujuan sehingga menemukan tujuan tersebut. BFS tidak menggunakan algoritma heuristik.



## Karakteristik BFS :

- ▶ Jika ada solusi, BFS akan menemukannya.
- ▶ BFS akan menemukan solusi dengan jalur erpendek.
- ▶ BFS tidak akan terjebak dalam “looping”.
- ▶ BFS membutuhkan space untuk menyimpan node list antrian dan space yang
- ▶ dibutuhkan dan mungkin space yang dibutuhkan itu cukup besar.
  - ▶ – Asumsi :
    - ▶ 1. Ada solusi dalam pohon
    - ▶ 2. Pencarian tree adalah secara terurut.

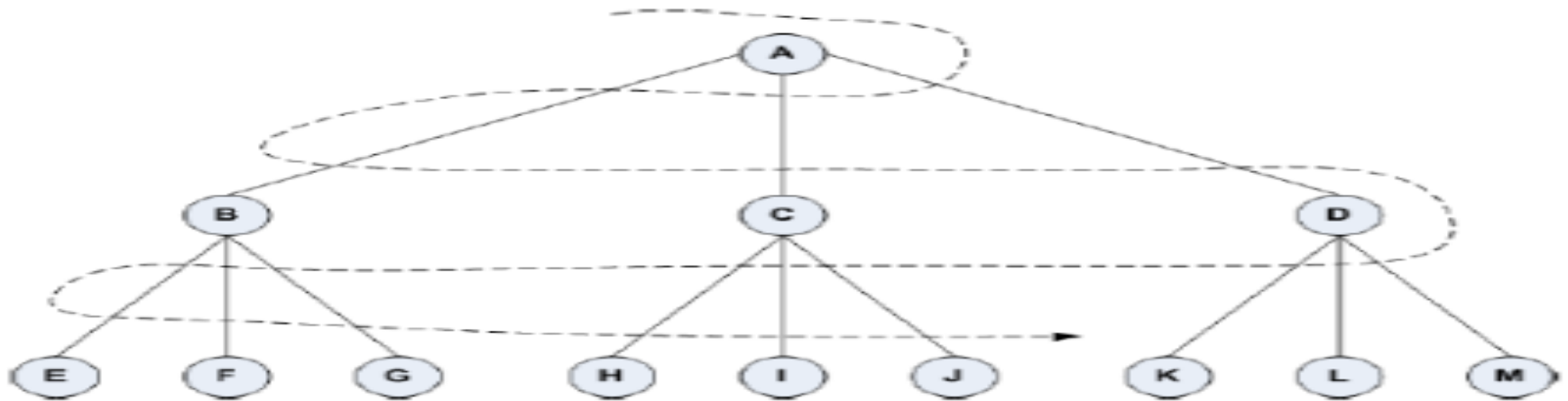
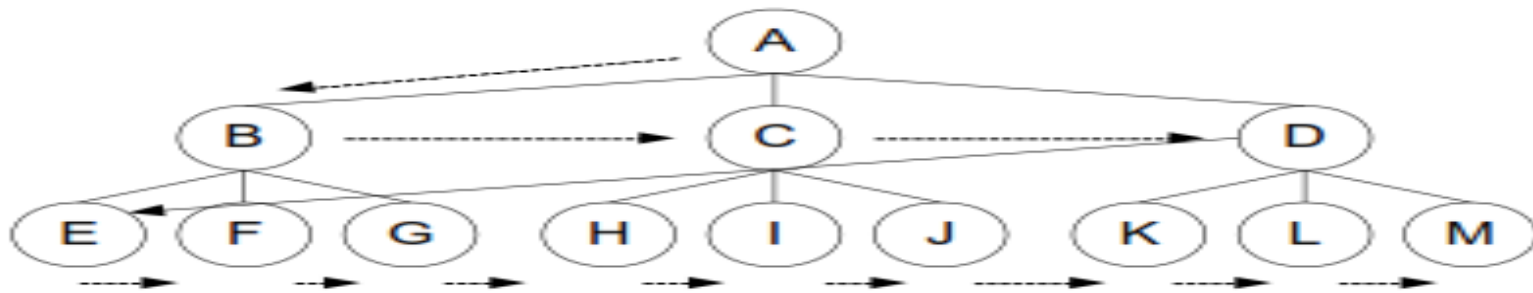


Figure 1.6: Diagram pohon dari BFS.



BFS melakukan searching pada semua node yang berada pada level yang sama terlebih dahulu, sebelum melanjutkan proses searching pada node di level berikutnya.

## Keterangan :

- ▶ Pada metode *breadth-first search*, semua node pada level  $n$  akan dikunjungi terlebih dahulu sebelum mengunjungi node-node pada level  $n+1$ .
- ▶ Pencarian dimulai dari node akar terus ke level ke-1 dari kiri ke kanan, kemudian berpindah ke level berikutnya, demikian pula dari kiri ke kanan hingga ditemukannya solusi.

## Keuntungan BFS :

- ▶ Tidak akan menemui jalan buntu
- ▶ Jika ada satu solusi, maka *breadth-first search* akan menemukannya. Dan, jika
- ▶ ada lebih dari satu solusi, maka solusi minimum akan ditemukan.

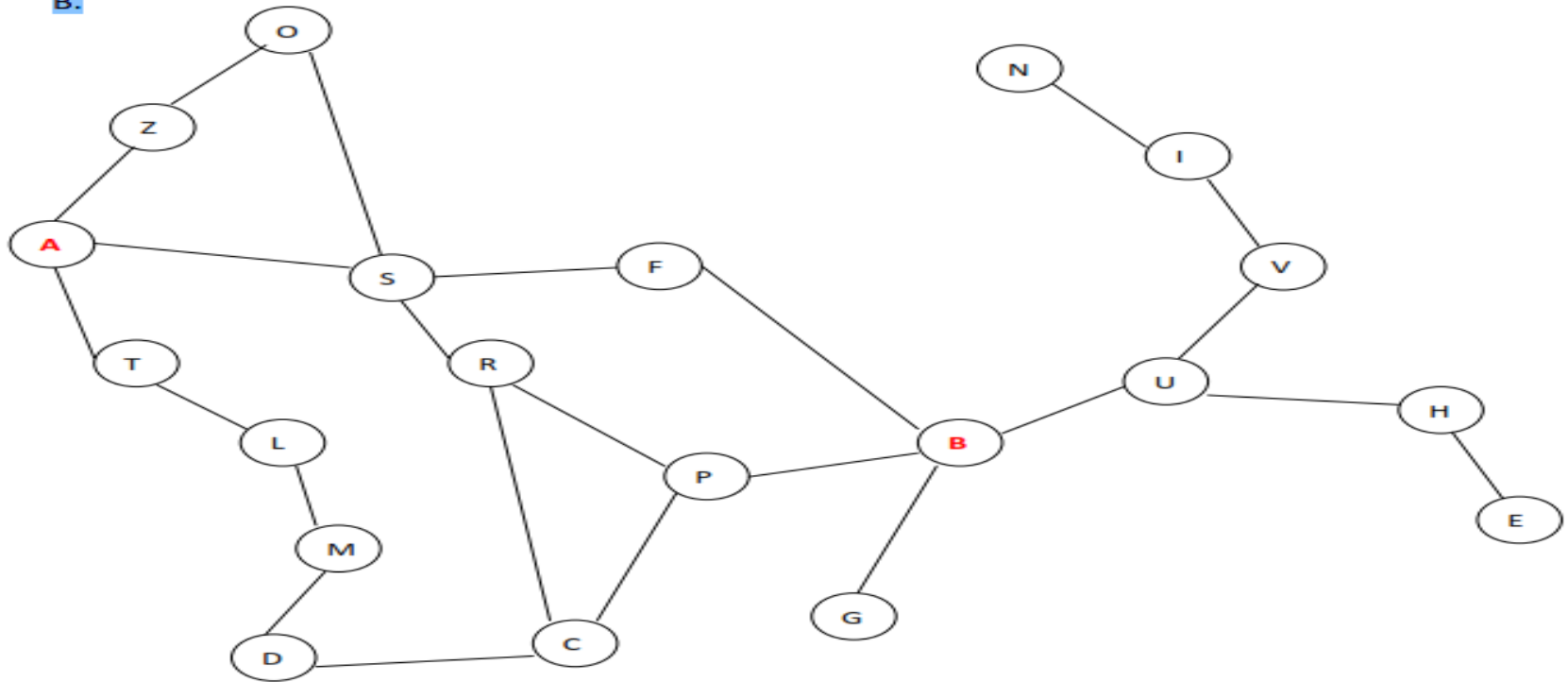
## Kelemahan BFS:

- ▶ Membutuhkan memori yang cukup banyak, karena menyimpan semua node dalam satu pohon
- ▶ Membutuhkan waktu yang cukup lama, karena akan menguji  $n$  level untuk mendapatkan solusi pada level yang ke- $(n+1)$ .

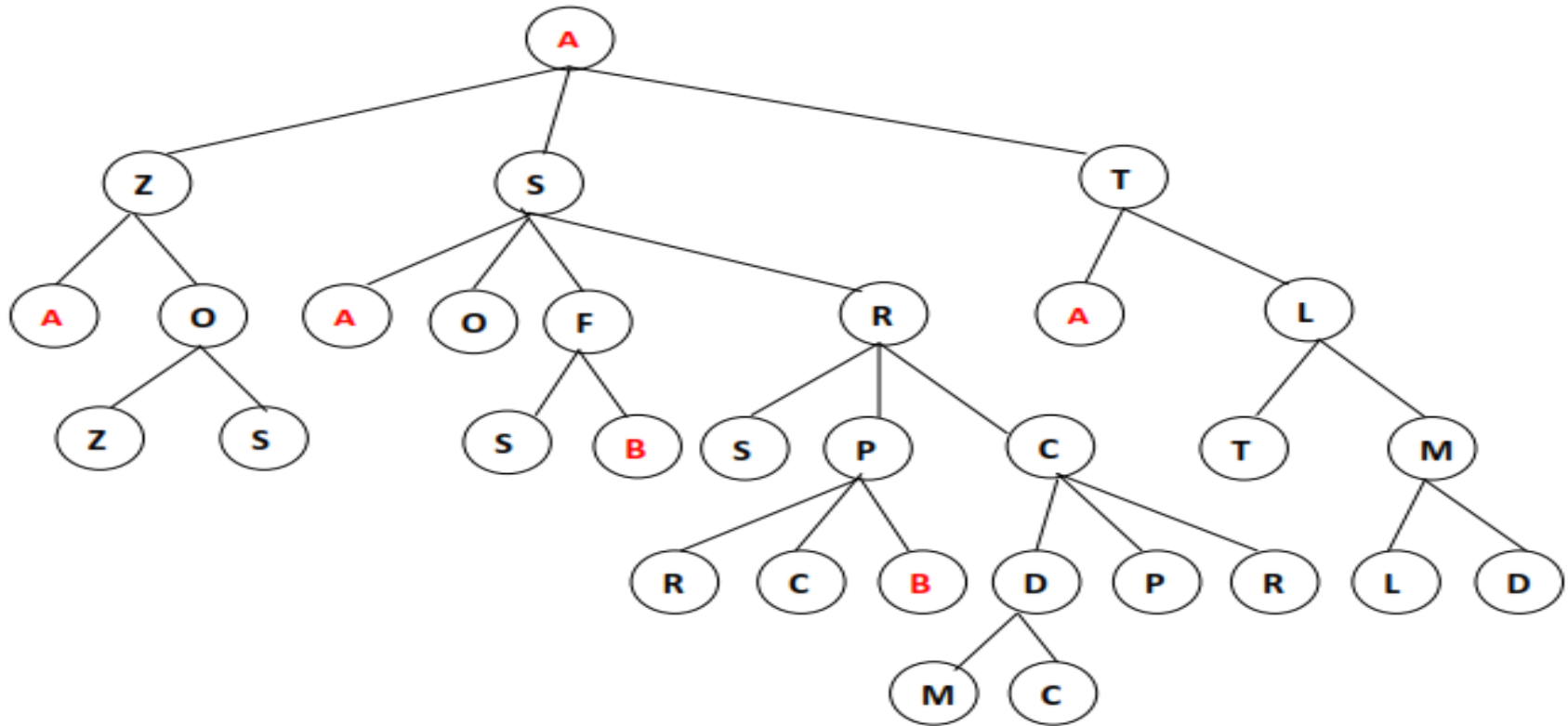
## Contoh soal BFS!

- ▶ Diketahui sebuah peta perjalanan dimana ingin mencari rute perjalanan dari kota A ke kota B.

B.



## Bagaimana metode pencarian dengan menggunakan BFS ??



Hasil pencarian menggunakan BFS mendapat solusi di level 3 dan level 4

## DEPTH FIRST SEARCH (DFS)

- ▶ *Depth-first search (DFS) adalah proses searching sistematis buta yang melakukan ekspansi sebuah path (jalur) menuju penyelesaian masalah sebelum melakukan eksplorasi terhadap path yang lain.*
- ▶ Proses searching mengikuti sebuah path tunggal sampai menemukan goal atau dead end.
- ▶ Apabila proses searching menemukan dead-end, DFS akan melakukan penelusuran balik ke node terakhir untuk melihat apakah node tersebut memiliki path cabang yang belum dieksplorasi.

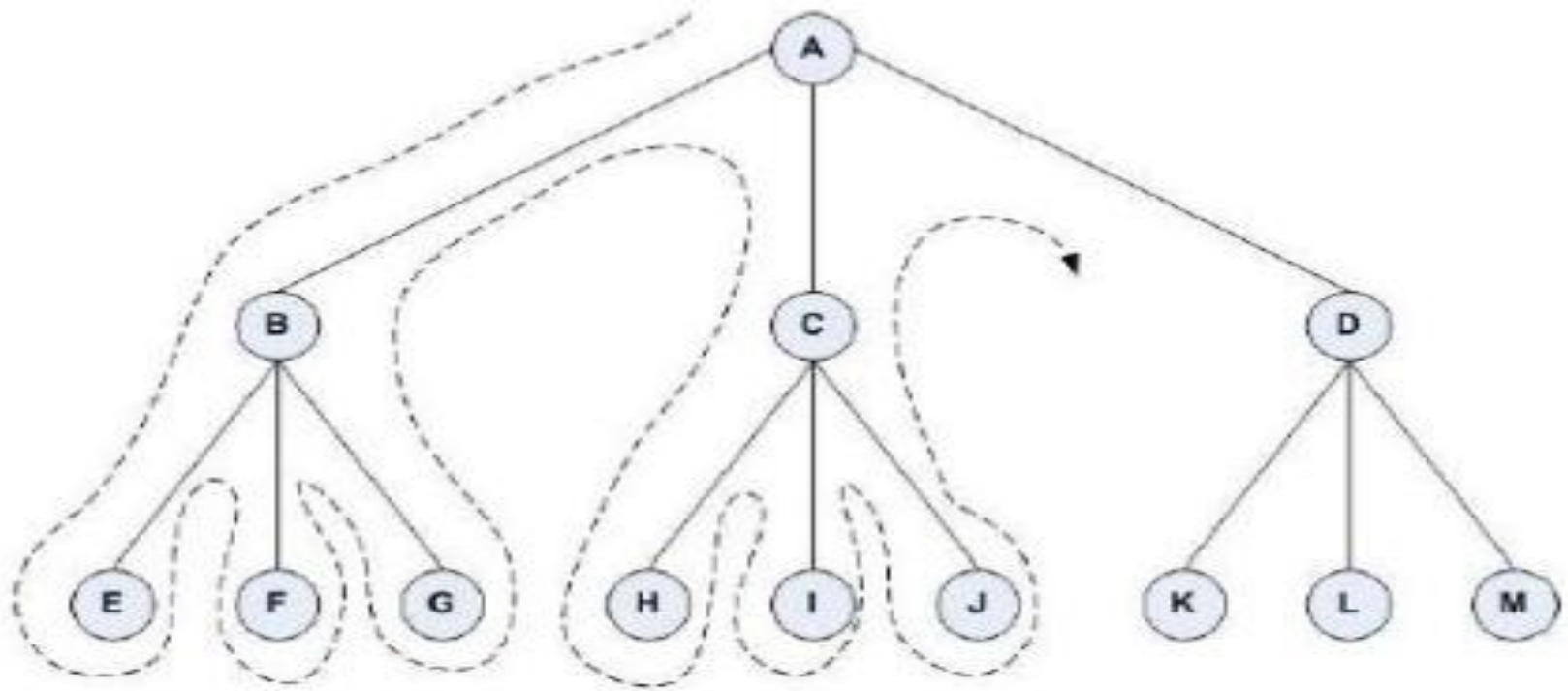
## Kelebihan DFS :

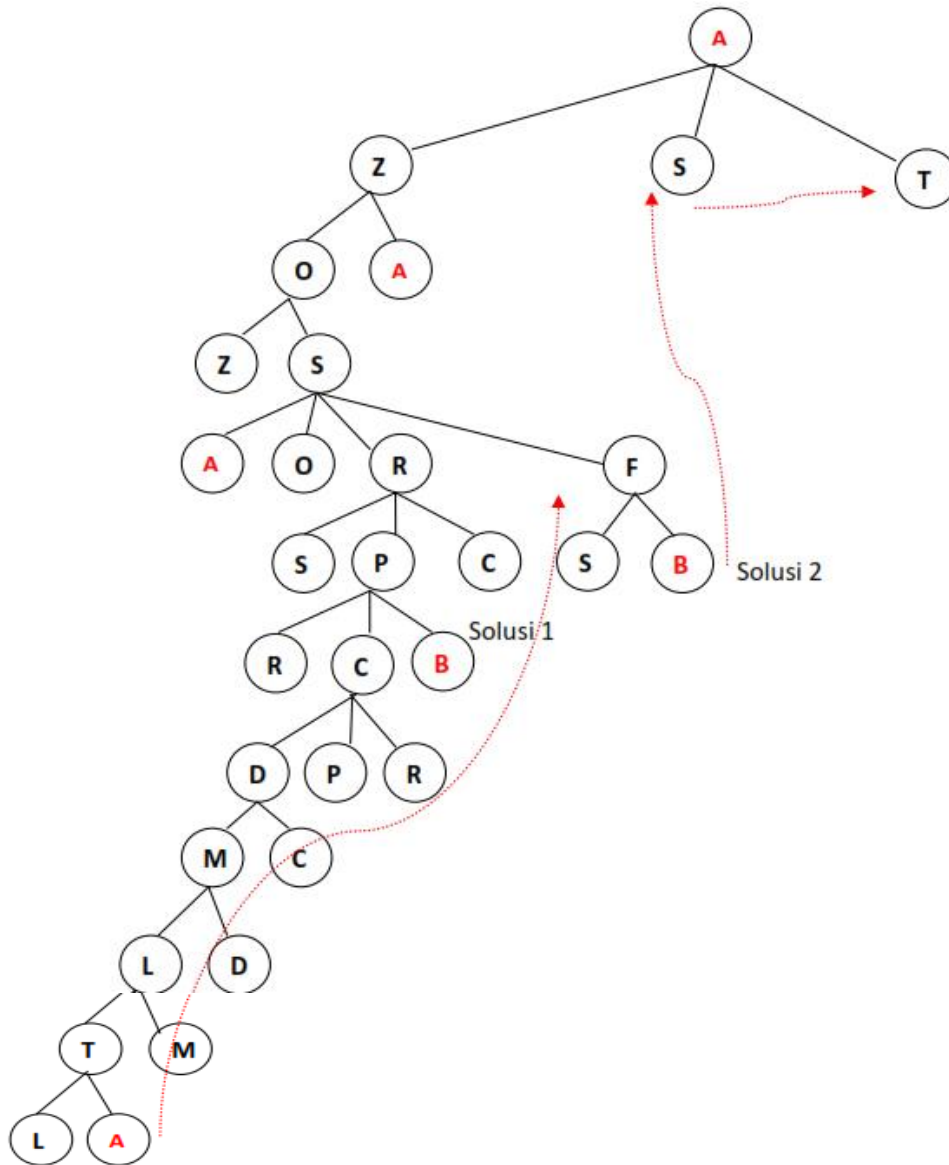
- ▶ Pemakaian memori hanya sedikit, berbeda jauh dengan BFS yang harus menyimpan semua node yang pernah dibangkitkan.
- ▶ Jika solusi yang dicari berada pada level yang dalam dan paling kiri, maka DFS akan menemukannya secara cepat.

## Kelemahan DFS :

- ▶ Jika pohon yang dibangkitkan mempunyai level yang dalam (tak terhingga), maka tidak ada jaminan untuk menemukan solusi (**Tidak *Complete***).
- ▶ Jika terdapat lebih dari satu solusi yang sama tetapi berada pada level yang berbeda, maka pada DFS tidak ada jaminan untuk menemukan solusi yang paling baik (**Tidak *Optimal***).







Contoh penerapan DFS,  
masih dari soal yang sama  
dengan BFS di atas.

# Pencarian Heuristik (Heuristic Search)

- ▶ Heuristik adalah sebuah teknik yang mengembangkan efisiensi dalam proses pencarian, namun dengan kemungkinan mengorbankan kelengkapan (*completeness*).
- ▶ Fungsi heuristik digunakan untuk mengevaluasi keadaan-keadaan problema individual dan menentukan seberapa jauh hal tersebut dapat digunakan untuk mendapatkan solusi yang diinginkan.

Ada 4 metode *Heuristic Searching*:

- ▶ *A. Generate and Test.*
- ▶ *B. Hill Climbing.*
- ▶ *C. Best First Search.*
- ▶ *D. Simulated Annealing*

## PEMBANGKITAN dan PENGUJIAN (*Generate and Test*)

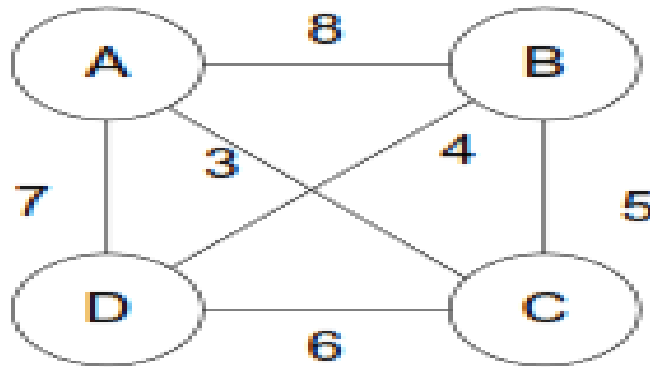
- ▶ Metode ini merupakan penggabungan antara *depth-first search* dengan *pelacakan* mundur (*backtracking*), yaitu *bergerak ke belakang menuju pada suatu keadaan awal*.

### Algoritma :

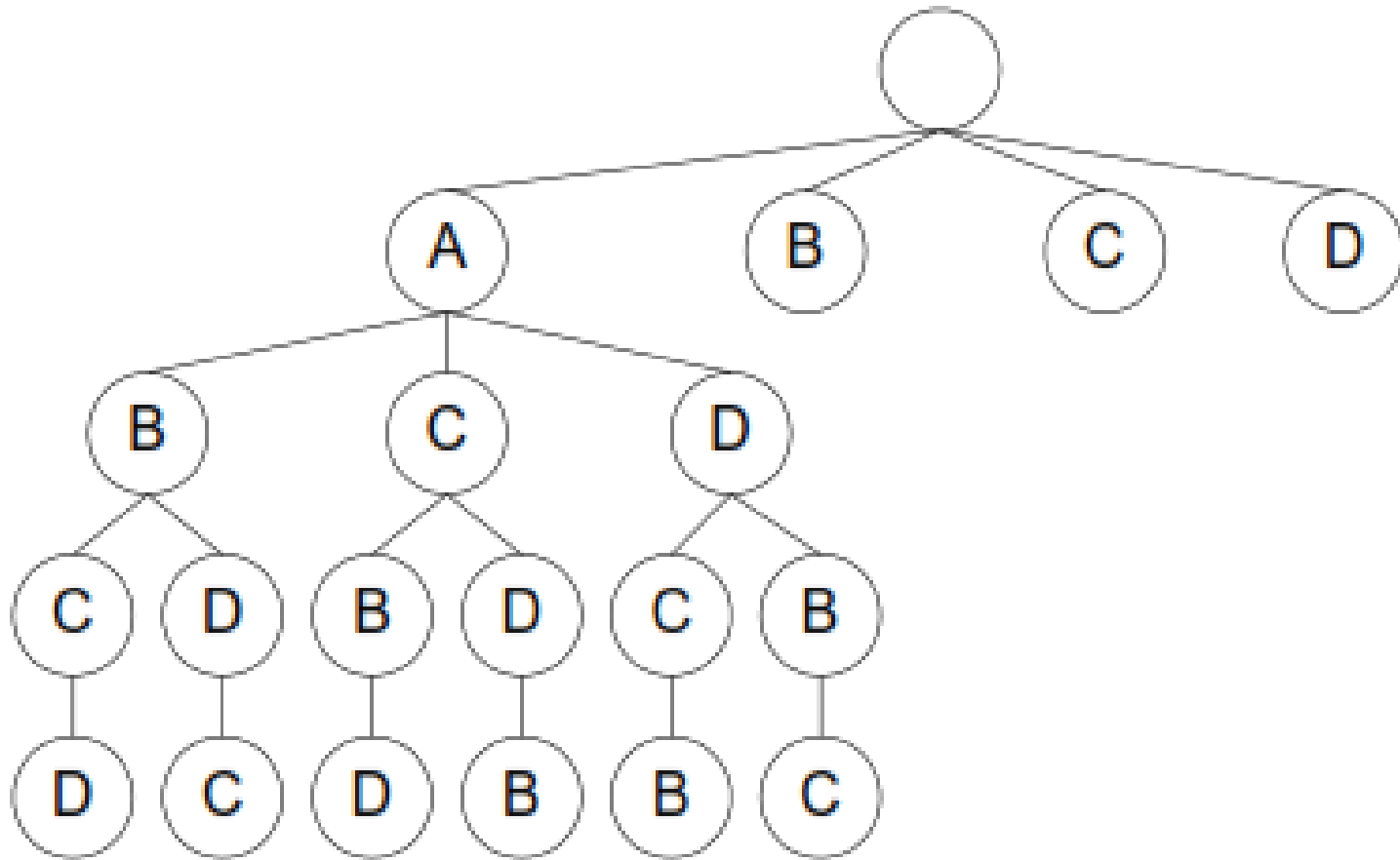
- ▶ Bangkitkan suatu kemungkinan solusi (membangkitkan suatu titik tertentu atau lintasan tertentu dari keadaan awal).
- ▶ Uji untuk melihat apakah node tersebut benar-benar merupakan solusinya dengan cara membandingkan node tersebut atau node akhir dari suatu lintasan yang dipilih dengan kumpulan tujuan yang diharapkan.
- ▶ Jika solusi ditemukan, keluar. Jika tidak, ulangi kembali langkah pertama.

## Contoh : Pada Traveling Salesman Problem (TSP)

- ▶ Seorang salesman ingin mengunjungi  $n$  kota. Jarak antara tiap–tiap kota sudah diketahui.
- ▶ Kita ingin mengetahui ruter terpendek dimana setaip kota hanya boleh dikkunjungi tepat 1 kali. Misalkan ada 4 kota dengan jarak antara tiap–tiap kota seperti gambar di bawah ini :



## Penyelesaian dengan metode *Generate and Test* :



### Alur pencarian dengan *Generate and Test*

Pencarian ke-	Lintasan	Panjang Lintasan	Lintasan Terpilih	Panjang Lintasan Terpilih
1	ABCD	19	ABCD	19
2	ABDC	18	ABDC	18
3	ACBD	12	ACBD	12
4	ACDB	13	ACBD	12
5	ADBC	16	ACBD	12
6	ADCB	18	ACBD	12
7	BACD	17	ACBD	12
8	BADC	21	ACBD	12
9	BCAD	15	ACBD	12
10	BCDA	18	ACBD	12
11	BDAC	14	ACBD	12
12	BDCA	13	ACBD	12
13	CABD	15	ACBD	12
14	CADB	14	ACBD	12
15	CBAD	20	ACBD	12

Pencarian ke-	Lintasan	Panjang Lintasan	Lintasan Terpilih	Panjang Lintasan Terpilih
16	CBDA	16	ACBD	12
17	CDAB	21	ACBD	12
18	CDBA	18	ACBD	12
19	DABC	20	ACBD	12
20	DACB	15	ACBD	12
21	DBAC	15	ACBD	12
22	DBCA	12	ACBD atau DBCA	12
23	DCAB	17	ACBD atau DBCA	12
24	DCBA	19	ACBD atau DBCA	12



## PENDAKIAN BUKIT (*Hill Climbing*)

- ▶ Metode ini hampir sama dengan metode pembangkitan dan pengujian, hanya saja proses pengujian dilakukan dengan menggunakan fungsi heuristic.
- ▶ Pembangkitan keadaan berikutnya tergantung pada feedback dari prosedur pengetesan. Tes yang berupa fungsi heuristic ini akan menunjukkan seberapa baiknya nilai terkaan yang diambil terhadap keadaan-keadaan lainnya yang mungkin.

## Algoritma *Simple Hill Climbing*

- ▶ Cari operator yang belum pernah digunakan; gunakan operator ini untuk mendapatkan keadaan yang baru.
- ▶ Kerjakan langkah–langkah berikut sampai solusinya ditemukan atau sampai tidak ada operator baru yang akan diaplikasikan pada keadaan sekarang : Cari operator yang belum digunakan; gunakan operator ini untuk mendapatkan keadaan yang baru.

a) Evaluasi keadaan baru tersebut :

- (i) Jika keadaan baru merupakan tujuan, keluar
- (ii) Jika bukan tujuan, namun nilainya lebih baik daripada keadaan sekarang, maka jadikan keadaan baru tersebut menjadi keadaan sekarang.
- (iii) Jika keadaan baru tidak lebih baik daripada keadaan sekarang, maka lanjutkan iterasi.

Pada *simple hill climbing*, ada 3 masalah yang mungkin:

- ▶ Algoritma akan berhenti kalau mencapai nilai optimum local
- ▶ Urutan penggunaan operator akan sangat berpengaruh pada penemuan solusi.
- ▶ Tidak diijinkan untuk melihat satupun langkah sebelumnya

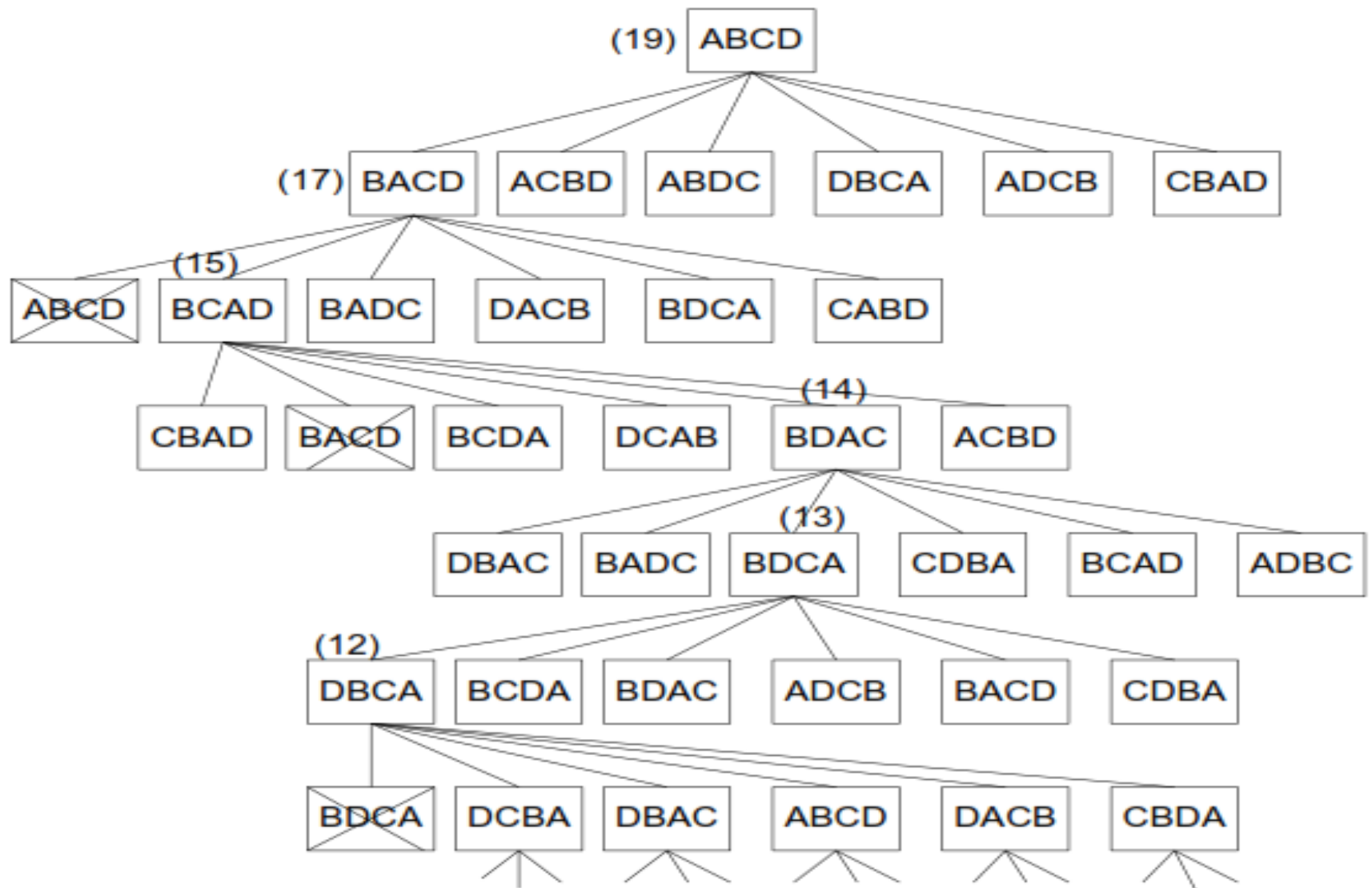
## Contoh : TSP dengan *Simple Hill Climbing*

Disini ruang keadaan berisi semua kemungkinan lintasan yang mungkin. Operator digunakan untuk menukar posisi kota-kota yang bersebelahan. Apabila ada  $n$  kota, dan kita ingin mencari kombinasi lintasan dengan menukar posisi urutan 2 kota, maka kita akan mendapatkan sebanyak :

$$\frac{n!}{2! (n-2)!}$$

kombinasi (lihat gambar di bawah).

Fungsi heuristic yang digunakan adalah panjang lintasan yang terjadi.



## Tugas :

Pada suatu hari ada seorang yang mempunyai seekor kambing dan srigala . Pada saat itu ia baru saja panen sayuran. Karena membutuhkan uang, petani tersebut hendak menjual kambing, srigala dan sayurannya ke pasar. Untuk sampai di pasar ia harus menyeberangi sebuah sungai. Permasalahannya adalah di sungai itu hanya tersedia satu perahu yang hanya bisa memuat petani dan satupenumpang lain (kambing, serigala, atau sayuran). Jika ditinggalkan oleh petani tersebut, maka sayuran akan di makan oleh kambing dan kambing akan di makan oleh srigala. Bagaimana caranya agar petani, kambing, srigala dan sayuran dapat selamat sampai di seberang sungai ?

# DAFTAR PUSTAKA

- Suyanto, “**Artificial Intelligence, Searching, Reasoning, Planning dan Learning**”, Informatika Bandung
- T.Sutojo, Edy Mulyanto, Dr. Vincent Suhartono, “**Kecerdasan Buatan**”, Penerbit Andi Offset.
- Sri Kusumadewi, “**Artificial Intelligence (teknik dan aplikasinya)**”, Penerbit Graha Ilmu
- Stuart Russel, Peter Norvig, “**Artificial Intelligence, A Modern Approach**”, Prentice Hall