KECERDASAN BUATAN

MASALAH, RUANG KEADAAN DAN PENCARIAN

ERWIEN TJIPTA WIJAYA, ST., M.KOM

KERANGKA MASALAH

- Masalah
- Ruang Keadaan
- Pencarian

DEFINISI MASALAH

- Sistem yang menggunakan kecerdasan buatan akan menghasilkan output berupa solusi dari suatu masalah, dimana penyelesaian masalah tersebut berdasarkan kumpulan pengetahuan yang ada.

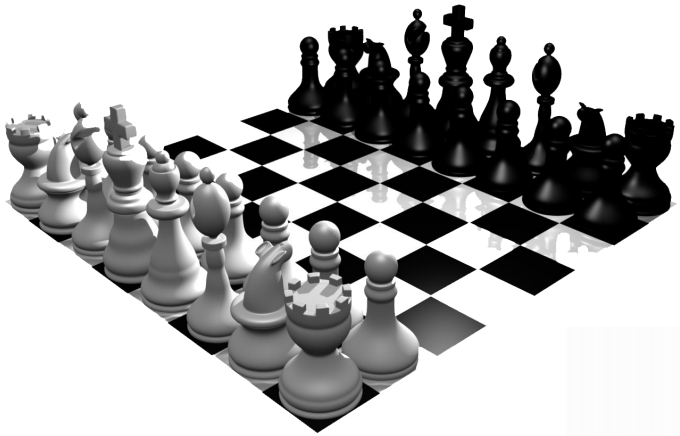
MEMBANGUN SISTEM YANG DAPAT MENYELESAIKAN MASALAH

- Secara umum untuk membangun suatu sistem yang mampu menyelesaikan masalah, perlu dipertimbangkan 4 hal :

1. Mendefinisikan masalah dengan tepat, pendefinisian ini mencakup spesifikasi yang tepat mengenai keadaan awal dan solusi yang diharapkan
2. Menganalisa masalah tersebut, serta mencari beberapa teknik penyelesaian masalah yang sesuai.
3. Merepresentasikan pengetahuan yang perlu untuk menyelesaikan masalah tersebut.
4. Memilih teknik penyelesaian masalah yang terbaik.

DEFINISI RUANG KEADAAN

- Ruang keadaan (***state space***) : suatu ruang yang berisi keadaan yang mungkin. contoh : pada permainan catur kita menempatkan diri pada keadaan awal, kemudian bergerak dari suatu keadaan ke keadaan yang lain sesuai dengan aturan permainan dan mengakhiri permainan jika salah satu telah mencapai tujuan / Goal. (misal : skak mat dimana raja sudah tidak bisa bergerak lagi)



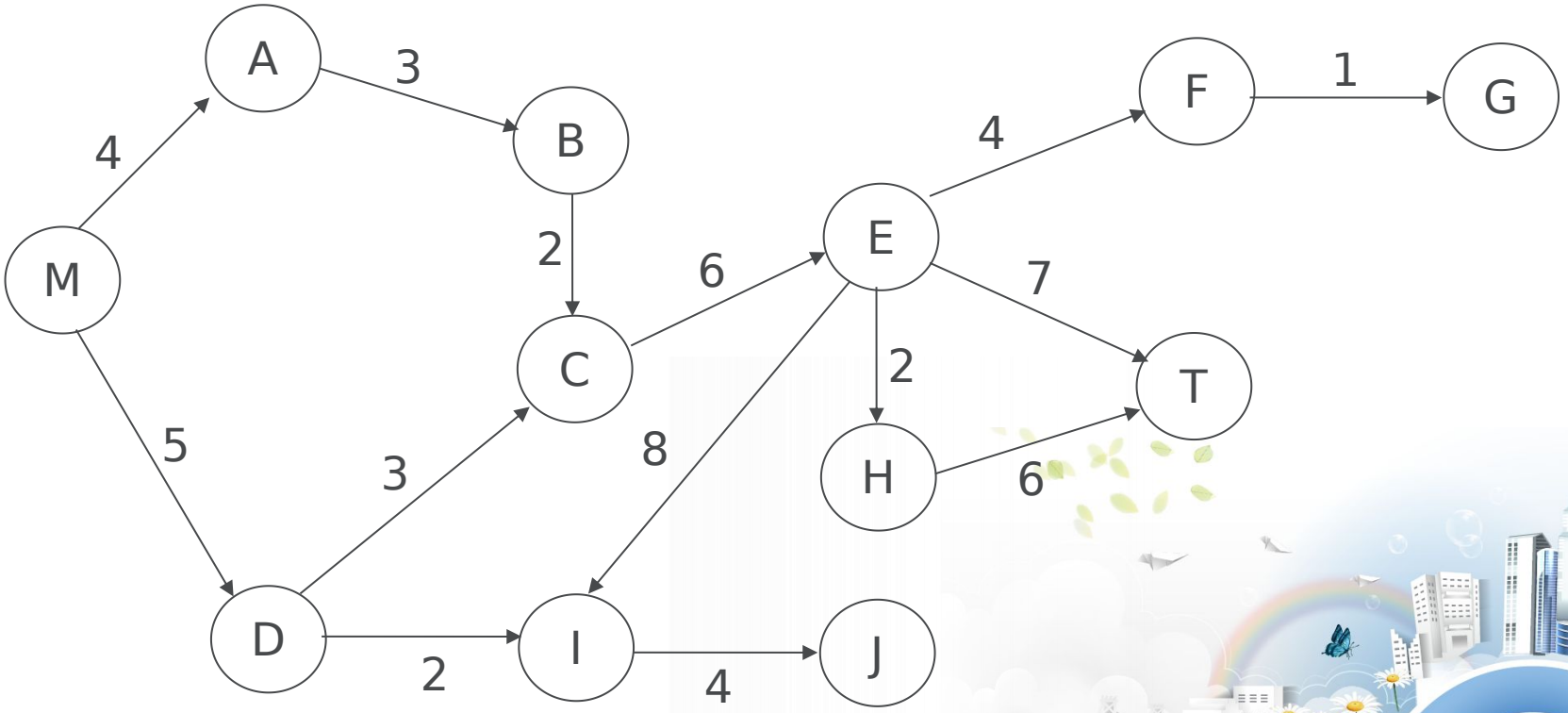
MENDESKRIPSIKAN MASALAH

- Secara umum untuk mendeskripsikan masalah dengan baik, harus :

1. Mendefinisikan suatu ruang keadaan.
2. Menetapkan satu atau lebih keadaan awal.
3. Menetapkan satu atau lebih tujuan (Goal).
4. Menetapkan kumpulan aturan (Knowledge).

CARA MEREPRESENTASIKAN RUANG KEADAAN

- ***Graph Keadaan*** : Graph terdiri - dari node - node yang menunjukkan keadaan baru yang akan dicapai dengan menggunakan operator. node - node dalam graph keadaan saling dihubungkan dengan menggunakan ***arc*** (busur) yang diberi panah untuk menunjukan arah dari suatu keadaan ke keadaan berikutnya.



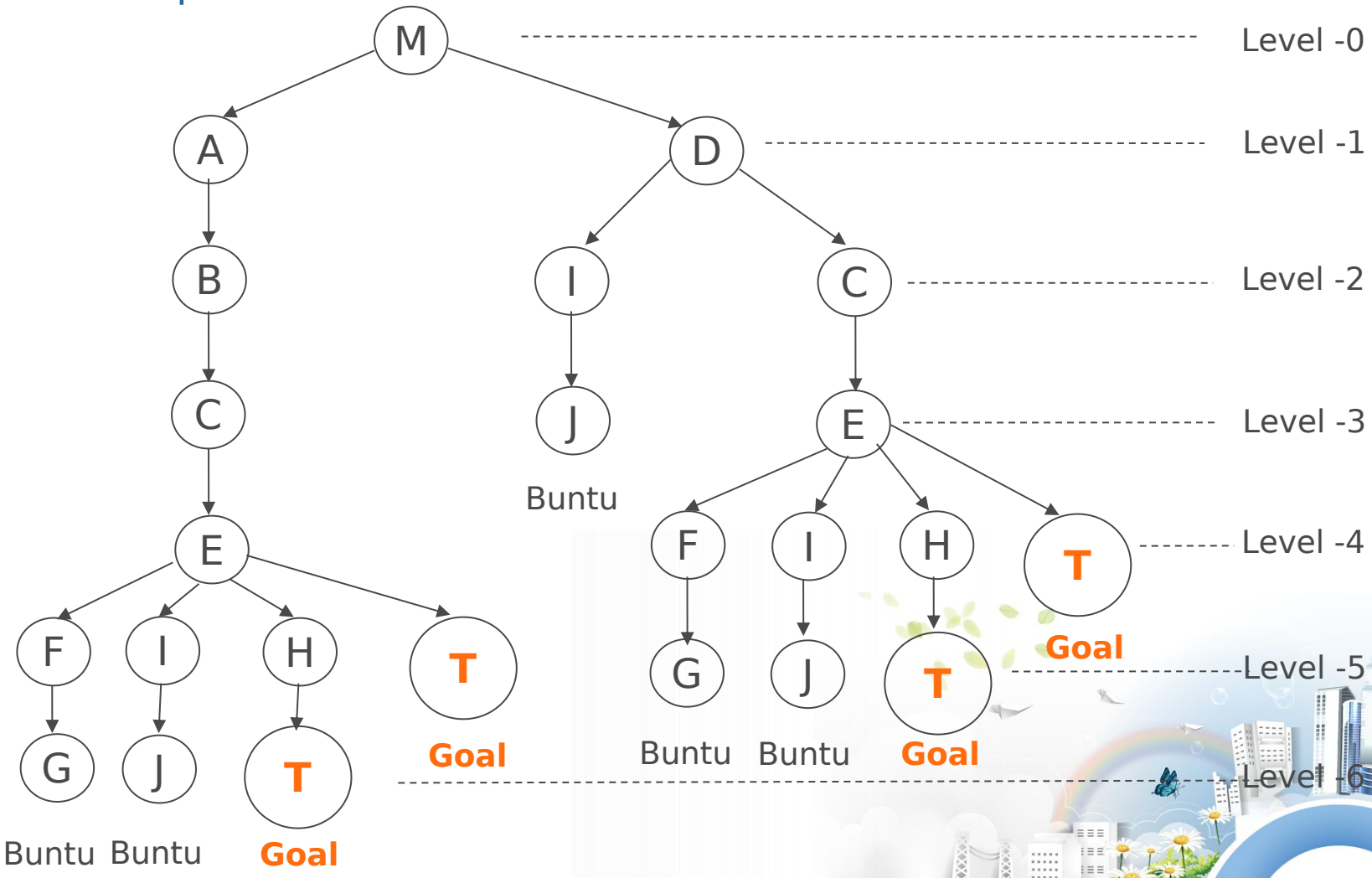
CARA MEREPRESENTASIKAN RUANG KEADAAN

- ***Graph keadaan*** : contoh **M** merupakan keadaan awal sedangkan **T** keadaan yang akan dicapai (***Goal***), ada 4 jalur yang dapat dilalui yaitu :
    1. M - A - B - C - E - T
    2. M - A - B - C - E - H - T
    3. M - D - C - E - T (Jalur penyelesaian yang efektif)
    4. M - D - C - E - H - T

- ***Graph Keadaan*** memiliki kelemahan apabila dibuat dalam software, berikut kelemahan - kelemahan yang tidak dapat diselesaikan dengan software, perhatikan jalur berikut ini :
    1. M - D - I - J (Jalur jalan buntu, Goal tidak tercapai)
    2. M - D - C - E - I - D - C (Jalur mengalami looping)
    3. M - D - C - E - F - G (Jalur jalan buntu, Goal tidak tercapai)
    4. M - A - B - C - E - I - D - C (Jalur mengalami looping)
    5. M - A - B - C - E - F - G (Jalur jalan buntu, Goal tidak tercapai)

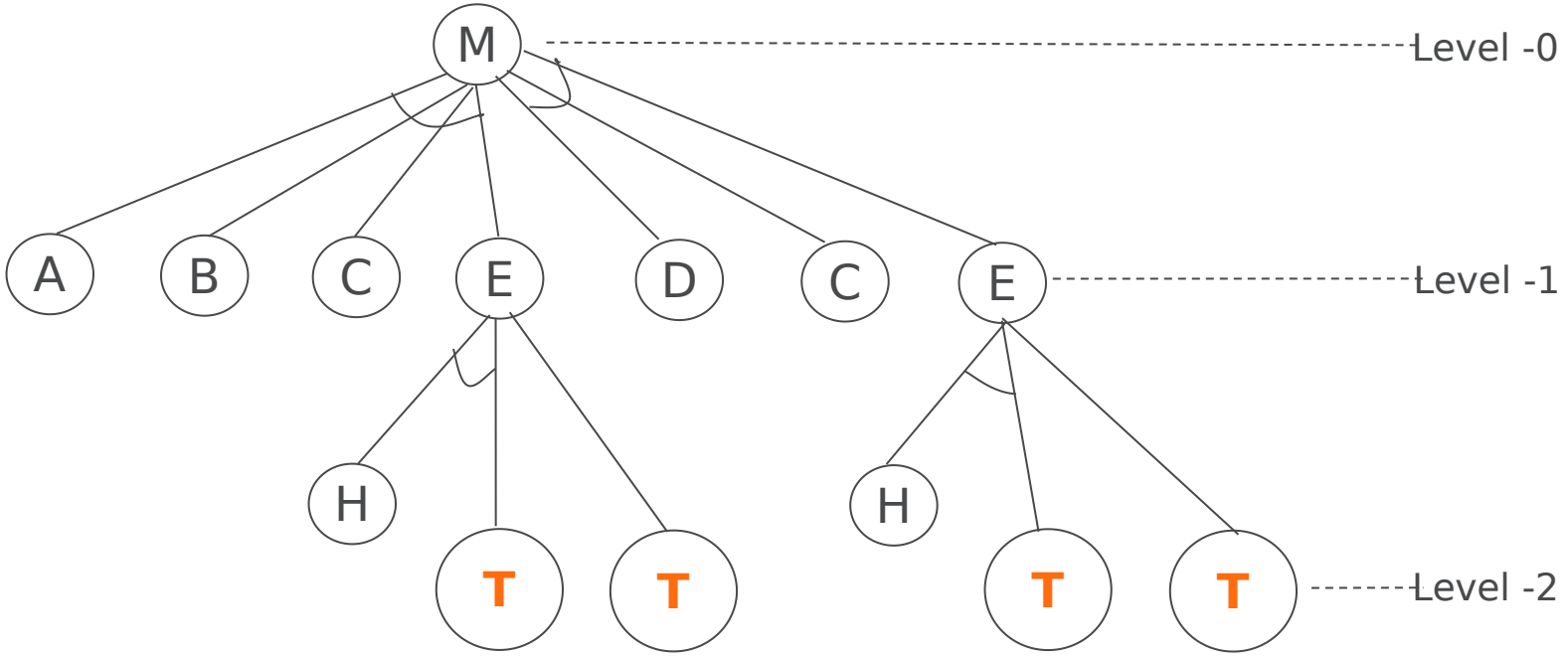
CARA MEREPRESENTASIKAN RUANG KEADAAN

- ***Pohon Pelacakan*** : untuk menghindari kemungkinan adanya proses pelacakan suatu node secara berulang, maka digunakan struktur pohon.



CARA MEREPRESENTASIKAN RUANG KEADAAN

- ***Pohon AND/OR*** :



CONTOH KASUS

- Seorang petani akan menyeberangkan seekor kambing, seekor serigala, dan sayur -sayuran dengan sebuah perahu yang melalui sungai.
- Perahu hanya bisa memuat petani dan 1 penumpang (Kambing, serigala atau sayur-sayuran) saja. Jika ditinggalkan oleh petanitersebut, maka sayur - sayuran akan dimakan oleh kambing, dan kambing akan dimakan oleh serigala.
- Penyelesaian :

1. Identifikasi ruang keadaan : Permasalahan ini dapat dilambangkan dengan (jumlah kambing, jumlah serigala, jumlah sayuran, jumlah perahu). sebagai contoh : daerah asal (0,1,1,1). berarti pada daerah asal tidak ada kambing, ada serigala, ada sayuranm dan ada perahu.

CONTOH KASUS

2. Keadaan awal & tujuan.

- Keadaan awal, pada kedua daerah :
  - Daerah asal : (1,1,1,1)
  - Daerah tujuan : (0,0,0,0)
- Tujuan, pada kedua daerah :
  - Daerah asal : (0,0,0,0)
  - Daerah tujuan : (1,1,1,1)

3. Aturan - aturan

No	Aturan
1	Kambing menyeberang
2	Sayuran menyeberang
3	Serigala menyeberang
4	Kambing kembali
5	Sayuran kembali
6	Serigala kembali
7	Perahu kembali

CONTOH KASUS

- Tabel solusi => (Kambing, Serigala, Sayur, Perahu)

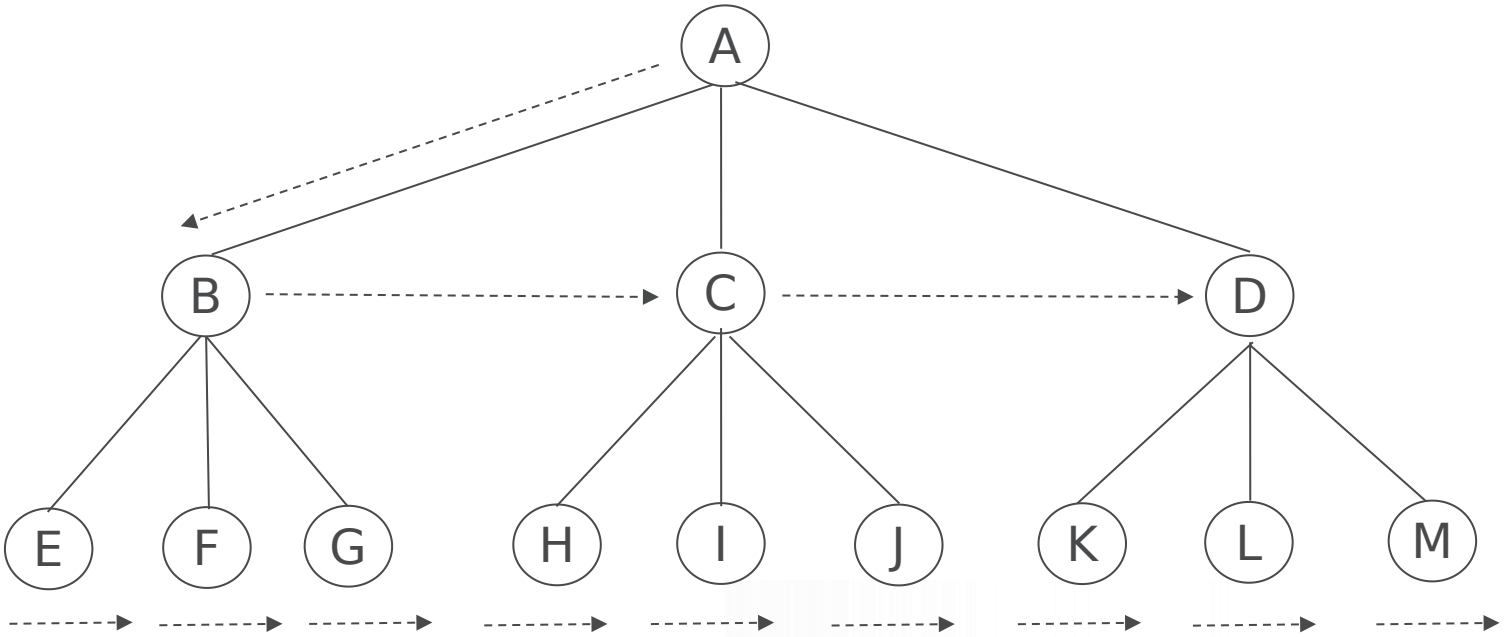
DAERAH ASAL	DAERAH TUJUAN	ATURAN
(1,1,1,1)	(0,0,0,0)	1
(0,1,1,0)	(1,0,0,1)	7
(0,1,1,1)	(1,0,0,0)	3
(0,0,1,0)	(1,1,0,1)	4
(1,0,1,1)	(0,1,0,0)	2
(1,0,0,0)	(0,1,1,1)	7
(1,0,0,1)	(0,1,1,0)	1
(0,0,0,0)	(1,1,1,1)	**SOLUSI**

METODE PENCARIAN DAN PELACAKAN

- Penentu keberhasilan sistem berdasarkan kecerdasan adalah kesuksesan dalam pencarian dan pencocokan.
- Ada 2 teknik pencarian dan pelacakan yaitu ***pencarian buta*** (*Blind search*) dan ***pencarian terbimbing*** (*Heuristic search*)

PENCARIAN BUTA (BLIND SEARCH)

- ***Breadth-First Search***



BREADTH-FIRST SEARCH

- **Algoritma** :

1. Buat suatu variabel *node_list* dan tetapkan sebagai keadaan awal.
2. Kerjakan langkah - langkah berikut ini sampai tujuan tercapai atau *node_list* dalam keadaan kosong:
    - Hapus elemen pertama dari *Node_list*, jika *node_list* kosong maka keluar.
    - Pada setiap langkah yang aturannya cocok dengan *Node*_list, maka kerjakan:
        a. Aplikasikan aturan tersebut untuk membentuk suatu keadaan baru.
        b. Jika keadaan awal adalah tujuan yang diharapkan, maka sukses dan keluar
        c. Jika tidak demikan, tambahkan keadaan awal yang baru tersebut pada akhir *Node_list*

BREADTH-FIRST SEARCH

- **Keuntungan** :

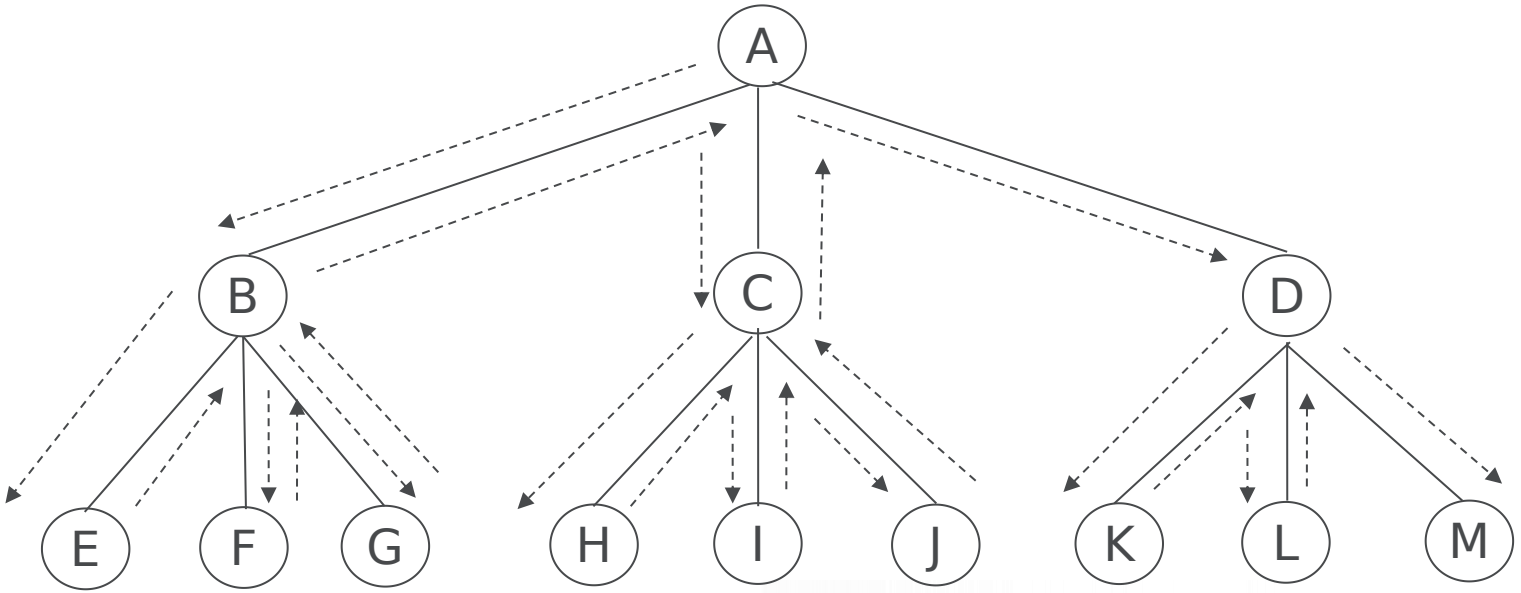
1. Tidak akan menemui jalan buntu.
2. Jika ada satu solusi,maka breadth-first search akan menemukannya. dan jika ada lebih dari 1 solusi, maka solusi minimum akan ditemukan.

- **Kelemahan** :

1. Membutuhkan memori yang besar, karena menyimpan node dalam 1 pohon
2. Membutuhkan waktu yang lama, karena akan menguji n level untuk mendapatkan solusi pada level yang ke-(n+1)

PENCARIAN BUTA (BLIND SEARCH)

- ***Depth-First Search***



DEPTH-FIRST SEARCH

- **Algoritma** :

1. Jika keadaan awal merupakan tujuan, keluar (sukses).
2. Jika keadan tidak demikan, kerjakan langkah - langkah berikut ini sampai tercapai keadaan sukses atau gagal:
    - Bangkitkan succesor dari keadaan awal, jika tidak ada succesor, maka akan terjadi kegagalan
    - Panggil depth-firsth search sebagai keadaan awal.
    - Jika sukses berikan tanda sukses, jika tidak ulangi langkah kedua.

DEPTH-FIRST SEARCH

- **Keuntungan** :

1. Membutuhkan memori relatif kecil, karena hanya node-node pada lintasan yang aktif saja yang disimpan.
2. Secara kebetulan, metode depth-firsth search akan menemukan solusi tanpa harus menguji lebih banyak lagi dalam ruang keadaan.

- **Kelemahan** :

1. Memungkinkan tidak ditemukannnya tujuan yang diharapkan
2. hanya akan mendapatkan 1 solusi pada setiap pencarian

TERIMA KASIH