Masalah, Ruang Masalah dan Pencarian

- ✓ **Definisi Masalah dan Ruang Masalah**
- ✓ **Metode Pencarian Buta**

Breadth First Search

Depth First Search

Referensi
Luger & Stubblefield - bab 3
Sri Kusumadewi - bab 2
Rich & Knight – bab 3

Definisi Masalah dan Ruang Masalah

Untuk **membangun sistem** yang mampu menyelesaikan masalah **menggunakan KB** :

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

Untuk **Mendefinisikan Suatu Masalah** :

- Definisikan/buat '*state space*' atau ruang masalah
- Tentukan keadaan awal (*initial state*)
- Tentukan keadaan akhir/tujuan (*goal state*)
- Tentukan operatornya/aturannya

Contoh 1 : "**Permainan Catur**"

Yang harus ditentukan adalah :

1. Posisi awal pada papan catur
2. Aturan-aturan untuk melakukan gerakan secara legal
3. Tujuan (*goal*) yang ingin dicapai adalah posisi pada papan catur yang menunjukkan kemenangan seseorang terhadap lawannya.

Contoh2 : **A water jug problem**

- *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 gollon air!
- Solusi: Buat asumsi dengan:
  - X : ember berkapasitas 4 gallon
  - Y : ember berkapasitas 3 gallon
- *Production Rules* :

  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*)

Untuk kasus water jug, production rules-nya :

- (X,Y), if (X < 4) → (4,Y)
- (X,Y), if (Y < 3) → (X,3)
- (X,Y), if X > 0 → (X-d,Y)
- (X,Y), if (Y > 0) → (X,Y-d)
- (X,Y), if X > 0 → (0,Y)
- (X,Y), if Y > 0 → (X,0)
- (X,Y) if X+Y >= 4 and Y > 0 → (4, Y-(4-X))
- (X,Y) if X+Y >= 3 and X > 0 → (X-(3-Y),3)
- (X,Y) if (X+Y) <= 4 and Y > 0 → (X+Y,0)
- (X,Y) if X+Y <= 3 and X >0 → (0,X+Y)
- (0,2) → (2,0)
- (X,2) → (0,2)

Salah satu solusinya :

X	Y	*Rules* yang digunakan
0	0	
		2
0	3	
		9
3	0	
		2
3	3	
		7
4	2	
		5 atau 12
0	2	
		9 atau 11
2	0	
		solusi

Contoh 3 : Masalah “**Petani, Kambing, Serigala dan Sayuran”**

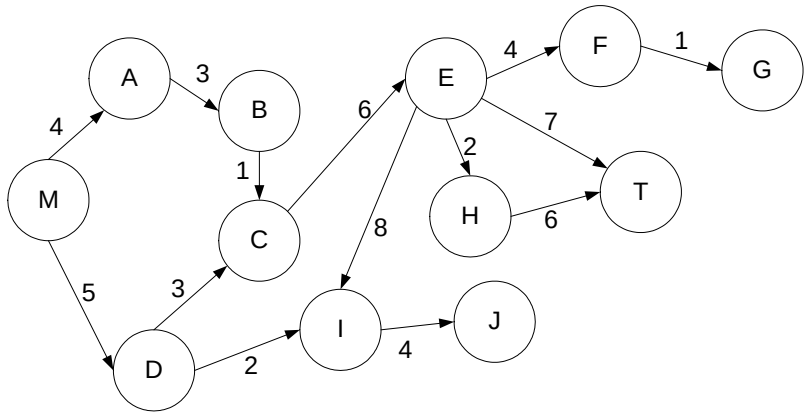
Seorang petani akan menyeberangkan seekor kambing, seekor serigala dan sayuran dengan sebuah boat yang melalui sungai. Boat hanya bisa memuat petani dan satu penumpang lain (kambing, serigala atau sayuran). Jika ditinggalkan oleh petani tersebut, maka sayuran akan dimakan oleh kambing dan kambing akan dimakan oleh serigala.

Bagaimana caranya agar petani, kambing, serigala dan sayuran dapat selamat sampai di seberang sungai ?

Beberapa cara **Merepresentasikan Ruang Masalah** :

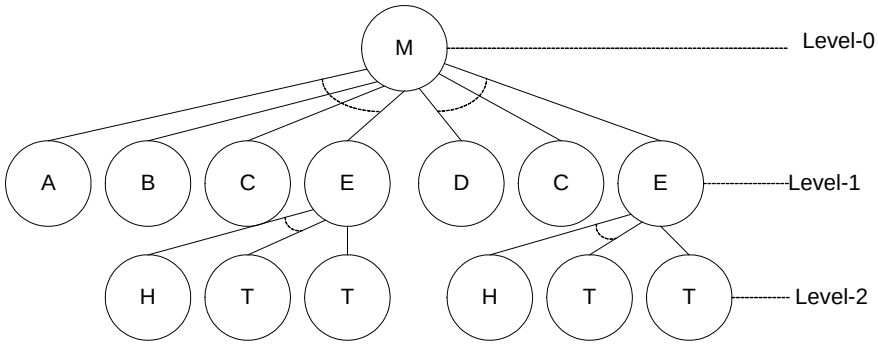
1. Graph Keadaan

Contoh :

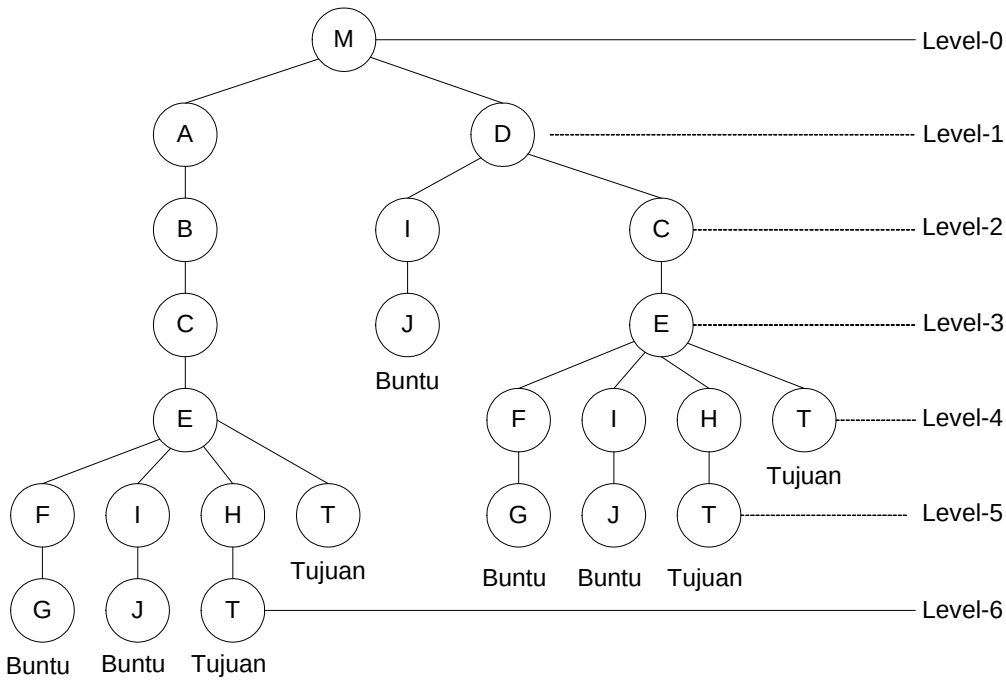


Graph berarah dengan satu tujuan (T)	Graph berarah yang menemui jalan buntu	Graph dengan siklus
• M-A-B-C-E-T • M-A-B-C-E-H-T • M-D-C-E-T • M-D-C-E-H-T	• M-A-B-C-E-F-G • M-A-B-C-E-I-J • M-D-C-E-F-G • M-D-C-E-I-J • M-D-I-J	D-E-C-E-I-D

2. Pohon Pelacakan



3. Pohon AND/OR



KARAKTERISTIK MASALAH/PROBLEM

Untuk memilih metode yang paling baik untuk memecahkan suatu masalah tertentu, diperlukan suatu analisa masalah. Dalam menganalisa suatu masalah kita perlu mengetahui beberapa karakte –ristik masalah, diantaranya adalah:

1. Apakah masalah dapat dipilah-pilah (*decompose- able*) menjadi sejumlah sub-masalah independent yang lebih kecil atau lebih mudah ?
2. Dapatkah langkah-langkah penyelesaian yang terbukti tidak tepat diabaikan ?
3. Apakah ruang lingkup atau semesta pembicaraan masalah dapat diprakirakan ?
4. Apakah solusi masalah yang baik telah dibanding- kan dengan semua solusi yang dimungkinkan ?
5. Apakah basis pengetahuan yang digunakan untuk memecahkan masalah bersifat konsisten ?
6. Apakah benar-benar dibutuhkan sejumlah besar informasi untuk memecahkan masalah yang sedang dihadapi, atau pengetahuan hanya penting untuk membatasi proses pencarian (*searching*) ?
7. Apakah sebuah komputer sendirian dapat diberi masalah dan kemudian menyajikan solusi secara sederhana, atau akankah solusi dari suatu masalah membutuhkan interaksi antara komputer dan manusia ?

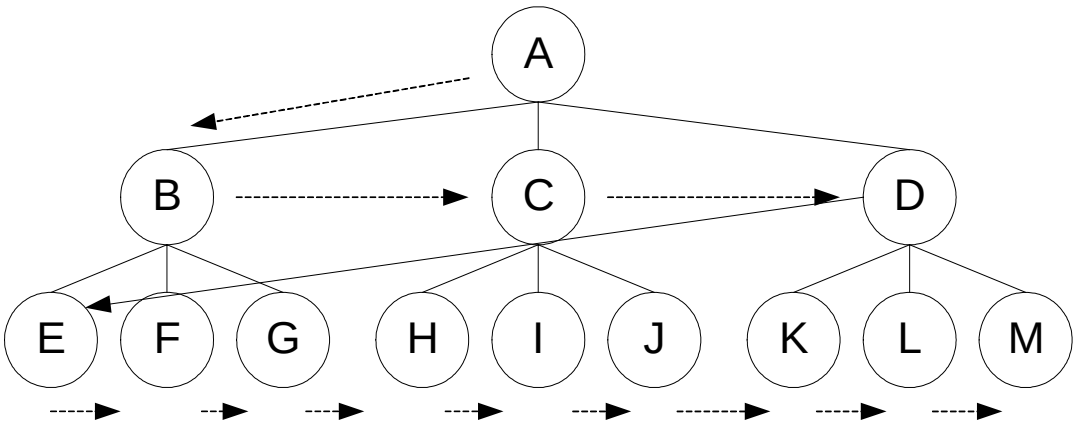
TEKNIKPENCARIAN/ PELACAKAN(SEARCHING)

- Pada umumnya manusia mempertimbangkan sejumlah alternatif strategi dalam menyelesaikan suatu problema.
- Dalam permainan catur misalnya, seorang pemain mempertimbangkan sejumlah kemungkinan tentang langkah-langkah berikutnya, memilih yang terbaik menurut kriteria tertentu seperti kemungkinan respon lawannya.
- Aspek tingkahlaku cerdas yang mendasari teknik penyelesaian problema seperti dalam permainan catur tersebut dinamakan proses pencarian ruang keadaan (*space state search*).
- *Exhaustive search* – adalah proses pencarian terhadap seluruh ruang keadaan serangkaian langkah yang paling dimungkinkan untuk menghasilkan kemenangan.
- Walaupun metode ini dapat diterapkan pada setiap ruang keadaan, namum ukuran ruang keadaan yang sangat besar membuat pendekatan ini secara praktis tidak dimungkinkan (dalam permainan catur terdapat 10120 keadaan)
- Bila kasus ini diimplementasikan ke dalam sisten komputer, maka akan membutuhkan memori yang sangat besar, dan waktu pencarian yang sangat lama. Dengan kata lain metode *exhaustive search* ini tidak efisien dan tidak efektif, sehingga tidak praktis untuk diimplementasikan.
- Untuk mengatasi kendala tersebut di atas, ada beberapa cara yang dapat dilakukan, diantaranya: pertama teknik pencarian parsial (Blind Search) dan yang kedua teknik pencarian heuristic (Heuristik Search).

Pencarian Buta (Blind Search)

A. PENCARIAN MELEBAR PERTAMA (*Breadth-First Search*)

- 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 (lihat gambar berikut).

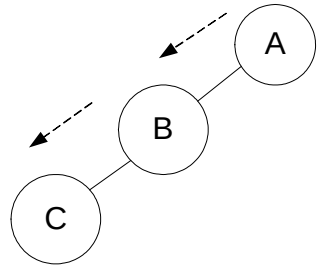


Prosedur breadth_first_search
Inisialisasi : open = [start]; closed [ ]
While open = [ ] do
Begin
Hapuskan keadaan paling kiri dari keadaan open,
sebutlah keadaan itu dengan X;
Jika X merupakan tujuan then return (sukses);
Buatlah semua child dari X;
Ambillah X dan masukkan pada closed;
Eliminasilah setiap child X yang telah berada
pada open atau closed, yang akan menyebabkan
loop dalam search;
Ambillah turunan di ujung kanan open sesuai urutan
penemuan-nya;
End.

- Keuntungan :
  - 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 :
  - 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).

B. PENCARIAN KEDALAM PERTAMA (*Depth-First Search*)

Pada *Depth-First Search*, proses pencarian akan dilakukanpada semua anaknya sebelum dilakukan pencarian ke node-node yang selevel. Pencarian dimulai dari node akar ke level yang lebih tinggi. Proses ini diulangi terus hingga ditemukannya solusi.



Prosedur depth_first_search
Inisialisasi: open = [Start]; closed = []
While open x [] do
Begin
Hapuskan keadaan berikutnya dari sebelah kiri open, sebutlah keadaan itu dengan X;
Jika X merupakan tujuan then return(sukses);
Buatlah semua child yang dimungkinkan dari X;
Ambilah X dan masukkan pada closed;
Eliminasilah setiap child X yang telah berada pada open atau closed, yang akan menyebabkan loop dalam search;
Ambilah child X yang tersisa di ujung kanan open sesuai urutan penemuannya;
End.

- Keuntungan :
  - Membutuhkan memori yang relative kecil, karena hanya node-node pada lintasan yang aktif saja yang disimpan.
  - Secara kebetulan, metode *depth-first search* akan menemukan solusi tanpa harus menguji lebih banyak lagi dalam ruang keadaan.
- Kelemahan :
  - Memungkinkan tidak ditemukannya tujuan yang diharapakan
  - Hanya akan menemukan 1 solusi pada setiap pencarian.