TEKNIK PENYELESAIAN MASALAH BERDASARKAN AI

1. **Definisikan masalah dengan tepat**
2. **Analisa masalahnya**
3. **Representasikan '*task knowledge*'**
4. **Pilih dan gunakan representasi dan teknik reasoning**

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**

Sistem Produksi/*Production System* terdiri dari:

- **Sekumpulan Aturan (*a set of rules*)**
- **Knowledge Base /Data Base**
- **Sebuah strategi pengontrol (*Control Strategy*)**
- **Arutan yang dipakai (*a rule applier*)**

Contoh kasus: *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 untuk masalah di atas (*water jug problem*) adalah:

1. **(X,Y) → (4,Y)**
   if (X < 4)
2. **(X,Y) → (X,3)**
   if (Y < 3)
3. **(X,Y) → (X-d,Y)**
   if X > 0
4. **(X,Y) → (X,Y-d)**
   if (Y > 0)
5. **(X,Y) → (0,Y)**
   if X > 0
6. **(X,Y) → (X,0)**
   if Y > 0
7. **(X,Y) → (4, Y-(4-X))**
   if X+Y >= 4 and Y > 0
8. **(X,Y) → (X-(3-Y),3)**
   if X+Y >= 3 and X > 0
9. **(X,Y) → (X+Y,0)**
   if (X+Y) <= 4 and Y > 0
10. **(X,Y) → (0,X+Y)**
    if X+Y <= 3 and X >0
11. **(0,2) → (2,0)**
12. **(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	

Buat solusi lainnya !

KARAKTERISTIK MASALAH/PROBLEMA

- **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/PENELUSURAN (*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 serangakaian 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 10^{120} 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 heuristik.**

PENCARIAN MELEBAR PERTAMA

(*Breadth-First Search*).

- **Prosedur pencarian melebar pertama (*breadth-first search*) merupakan prosedur yang menjamin diperolehnya sebuah solusi jika solusi itu memang ada, dimana tersedia sejumlah percabangan pohon (*tree*) yang berhingga**

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);

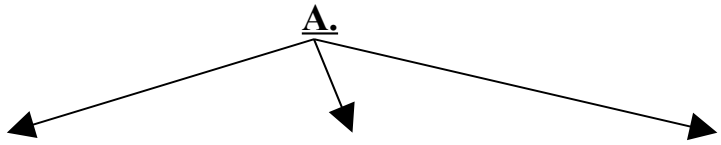
Buatlah semua child dari X;

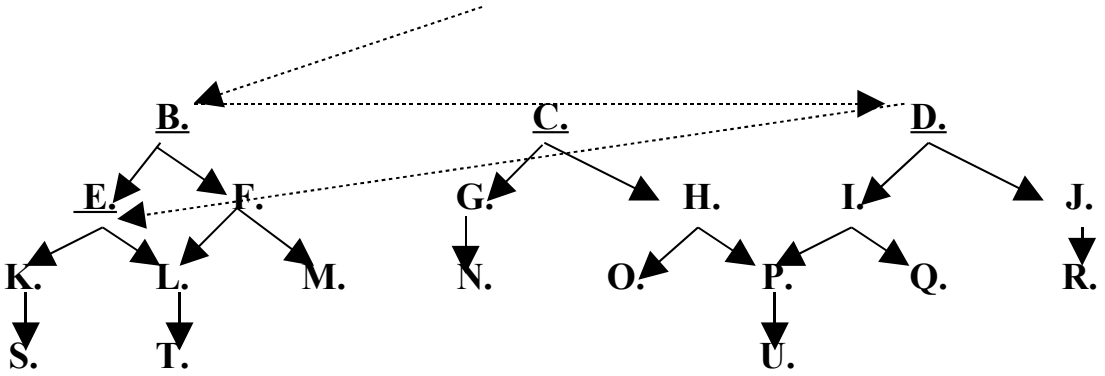
Ambillah X dan masukkan pada closed;

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;





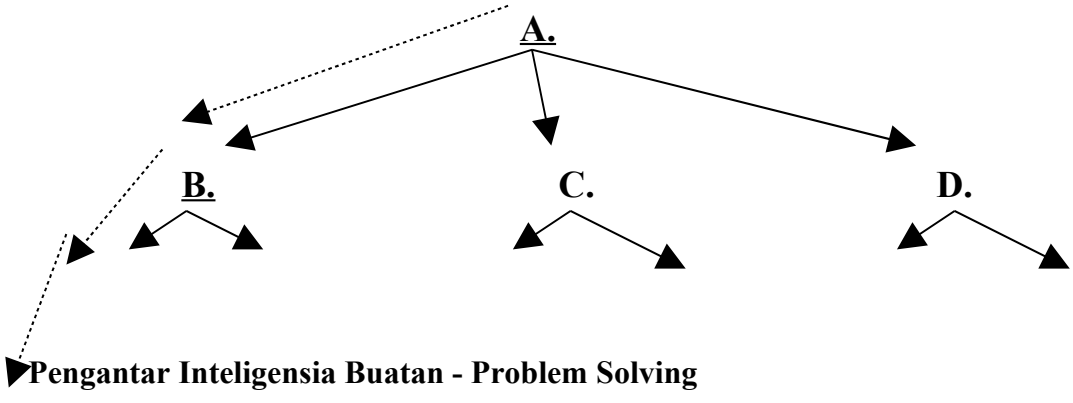
Gambar 1 Sebuah Graf setelah 6 iterasi breadth_first_search.

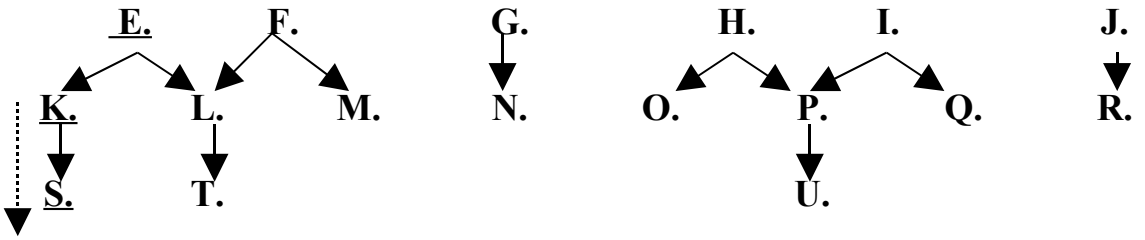
- **Karena proses pencarian *breadth-first* mengamati setiap simpul di setiap tingkat graf sebelum bergerak menuju ruang yang lebih dalam, maka mula-mula semua keadaan akan dicapai lewat lintasan yang terpendek dari keadaan awal. Karena itu, proses pencarian ini menjamin ditemukannya lintasan terpendek dari keadaan awal ke keadaan tujuan.**

PENCARIAN KEDALAM PERTAMA (Depth-First Search)

Pencarian atau Penelusuran Kedalam Pertama melakukan penelusuran dari Root menuju node paling kiri kemudian meneruskan kedalam (level berikutnya) sampai ditemukan daun (*leap*), Jika daun (*leap*) tersebut merupakan *goal*-nya maka berhenti (sukses), jika bukan *goal* maka proses dilanjutkan ke *leap* yang lain (melebar) atau *backtracking* ke node di level atasnya sampai ditemukanya *goal*.

```
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.
```





Gambar 2. Sebuah Graf. Setelah 6 iterasi depth-first search.

Key Points

- **Karena proses pencarian Breadth-First selalu memeriksa setiap simpul (node) pada tingkat n sebelumnya memproses tingkat n + 1, maka pencarian tersebut selalu mendapatkan lintasan terpendek menuju simpul tujuan**

- **Proses pencarian Depth-First akan dengan cepat mencapai kedalaman ruang pencarian. Jika diketahui bahwa lintasan solusi problema akan panjang, maka pencarian depth_first tidak akan memboroskan waktu untuk melakukan pencarian sejumlah besar keadaan 'dangkal' dalam graf problema.**
- **Pencarian depth-first jauh lebih efisien untuk ruang pencarian dengan banyak percabangan, karena tidak perlu harus mengevaluasi semua simpul pada suatu tingkat tertentu.**