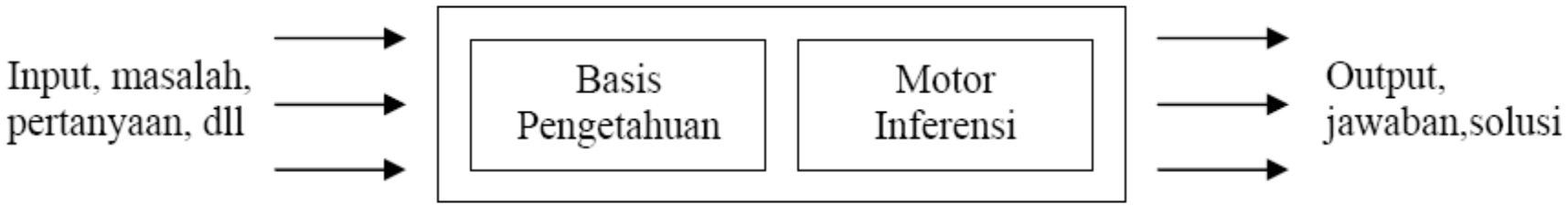
MASALAH DAN METODE PEMECAHAN MASALAH (Minggu 2)

Pendahuluan

- Sistem yang menggunakan kecerdasan buatan akan memberikan output berupa solusi dari suatu masalah berdasarkan kumpulan pengetahuan yang ada.



Gambar sistem yang menggunakan kecerdasan buatan

Pendahuluan (Lanjt)

- Pada gambar, input yg diberikan pada sistem yg menggunakan kecerdasan buatan adalah berupa masalah. Sistem harus dilengkapi dengan sekumpulan pengetahuan yang ada pada basis pengetahuan. Sistem harus memiliki motor inferensi agar mampu mengambil kesimpulan berdasarkan fakta atau pengetahuan. Output yang diberikan berupa solusi masalah sebagai hasil dari inferensi.

Pendahuluan (Lanjt)

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. *Menganalisis 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*

MENDEFINISIKAN MASALAH SEBAGAI SUATU RUANG KEADAAN

Misalkan permasalahan yang dihadapi adalah permainan catur, maka harus ditentukan :

1. posisi awal pada papan catur posisi awal setiap permainan catur selalu sama, yaitu semua bidak diletakkan di atas papan catur dalam 2 sisi, yaitu kubu putih dan kubu hitam.
2. aturan – aturan untuk melakukan gerakan aturan – aturan ini sangat berguna untuk menentukan gerakan suatu bidak, yaitu melangkah dari satu keadaan ke keadaan lain.

Misalkan untuk mempermudah menunjukkan

posisi bidak, setiap kotak ditunjukkan dalam huruf (a,b,c,d,e,f,g,h) pada arah horisontal dan angka (1,2,3,4,5,6,7,8) pada arah vertikal. Suatu aturan untuk menggerakkan bidak dari posisi (e,2) ke (e,4) dapat ditunjukkan dengan aturan :

if bidak putih pada kotak(e,2),

and kotak(e,3) kosong,

and kotak(e,4) kosong

then gerakkan bidak dari (e,2) ke (e,4)

3. tujuan (goal) tujuan yang ingin dicapai adalah posisi pada papan catur yang menunjukkan kemenangan seseorang terhadap lawannya. Kemenangan ini ditandai dengan posisi raja yang sudah tidak dapat bergerak lagi.

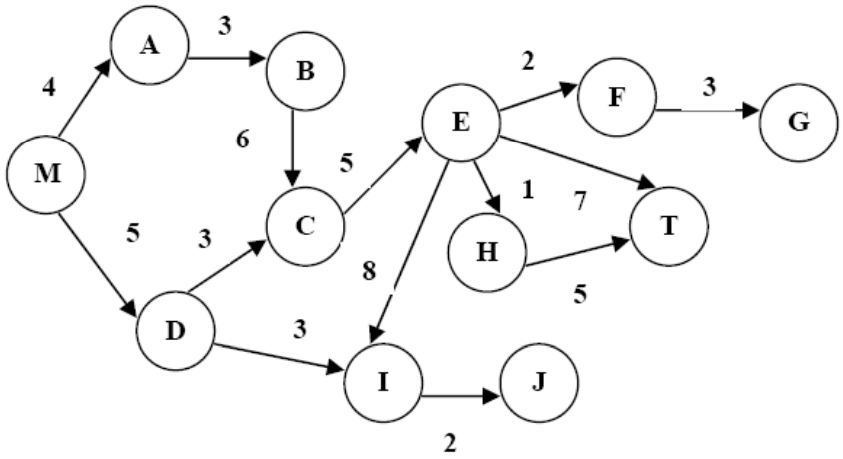
- Contoh tersebut menunjukkan representasi masalah dalam Ruang Keadaan (State Space), yaitu suatu ruang yang berisi semua keadaan yang mungkin. Kita dapat memulai bermain catur dengan menempatkan diri pada keadaan awal, kemudian bergerak dari satu keadaan ke keadaan yang lain sesuai dengan aturan yang ada, dan mengakhiri permainan jika salah satu telah mencapai tujuan.
- **Jadi untuk *mendeskripsikan masalah dengan baik harus :***

1. *Mendefinisikan suatu ruang keadaan (state space)*
2. *Menetapkan satu atau lebih keadaan awal (initial state)*
3. *Menetapkan satu atau lebih tujuan (goal state)*
4. *Menetapkan kumpulan aturan*

Cara Merepresentasikan Ruang Keadaan

- *GRAPH KEADAAN*

  Graph terdiri dari node-node yang menunjukkan keadaan yaitu keadaan awal dan keadaan baru yang akan dicapai dengan menggunakan operator. Node-node dalam graph keadaan saling dihubungkan dengan menggunakan arc (busur) yang diberi panah untuk menunjukkan arah dari suatu keadaan ke keadaan berikutnya.



Graph keadaan dengan node M menunjukkan keadaan awal, node T adalah tujuan. Ada 4 lintasan dari M ke T :

- M-A-B-C-E-T
- M-A-B-C-E-H-T
- M-D-C-E-T
- M-D-C-E-H-T

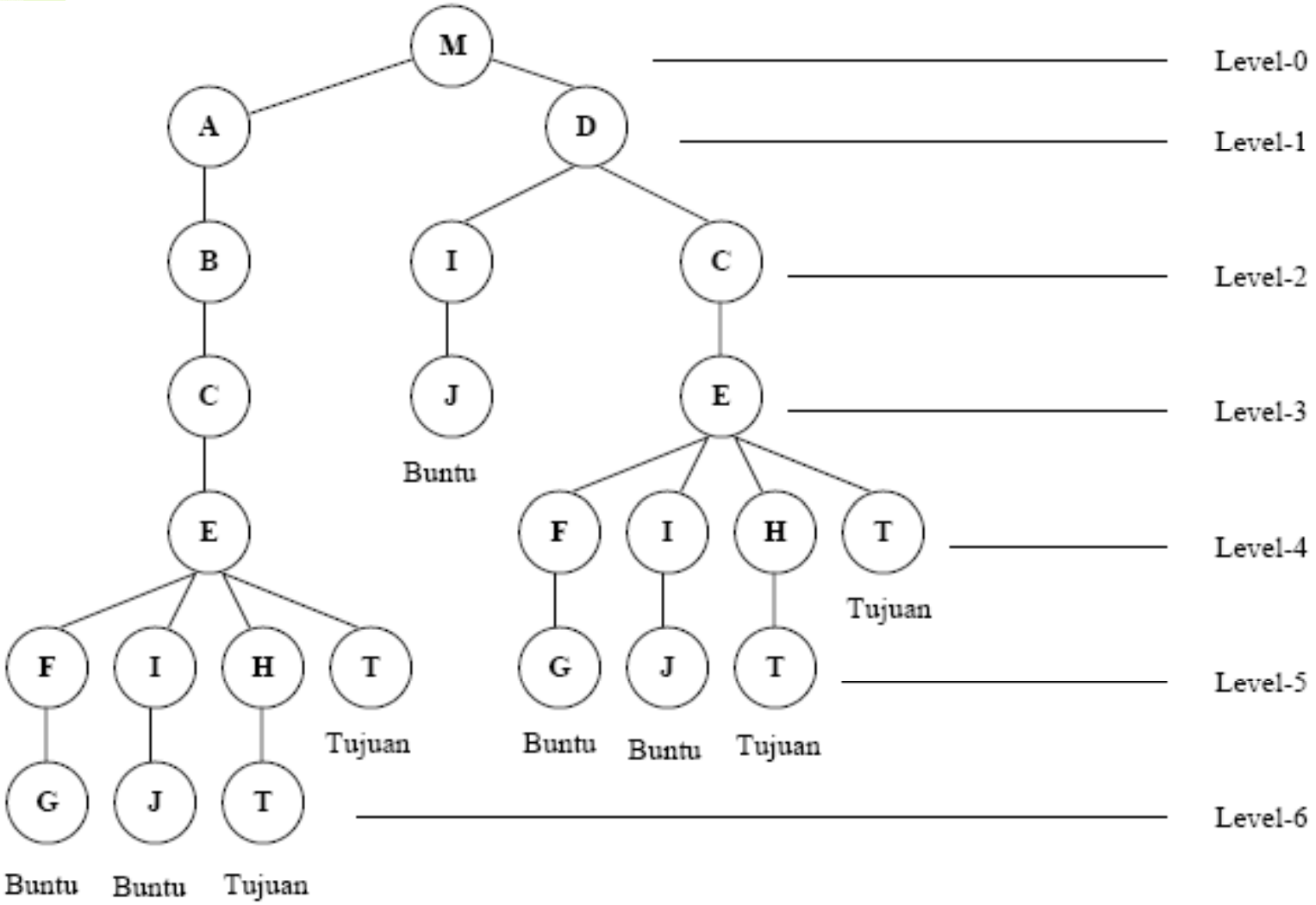
Lintasan buntu atau lintasan yang tidak sampai ke tujuan :

- 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

POHON PELACAKAN / PENCARIAN

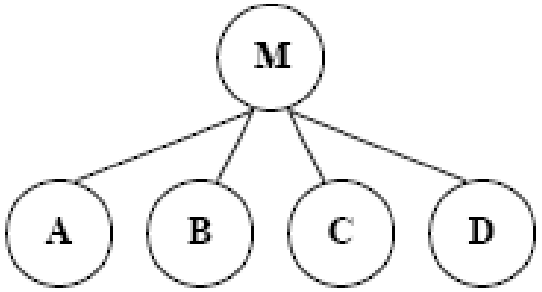
- Struktur pohon digunakan untuk menggambarkan keadaan secara hirarkis. Node yg terletak pada level-o disebut ’akar’.
- Node akar : menunjukkan keadaan awal & memiliki beberapa percabangan yang terdiri atas beberapa node yg disebut ’anak’ .
- Node-node yg tidak memiliki anak disebut ’daun’ menunjukkan akhir dari suatu pencarian, dapat berupa tujuan yang diharapkan (goal) atau jalan buntu (dead end).

- Gambar berikut menunjukkan pohon pencarian untuk graph keadaan dengan 6 level.

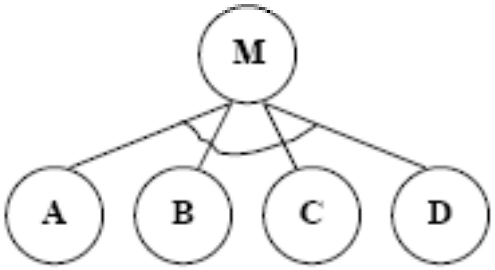


POHON AND/OR

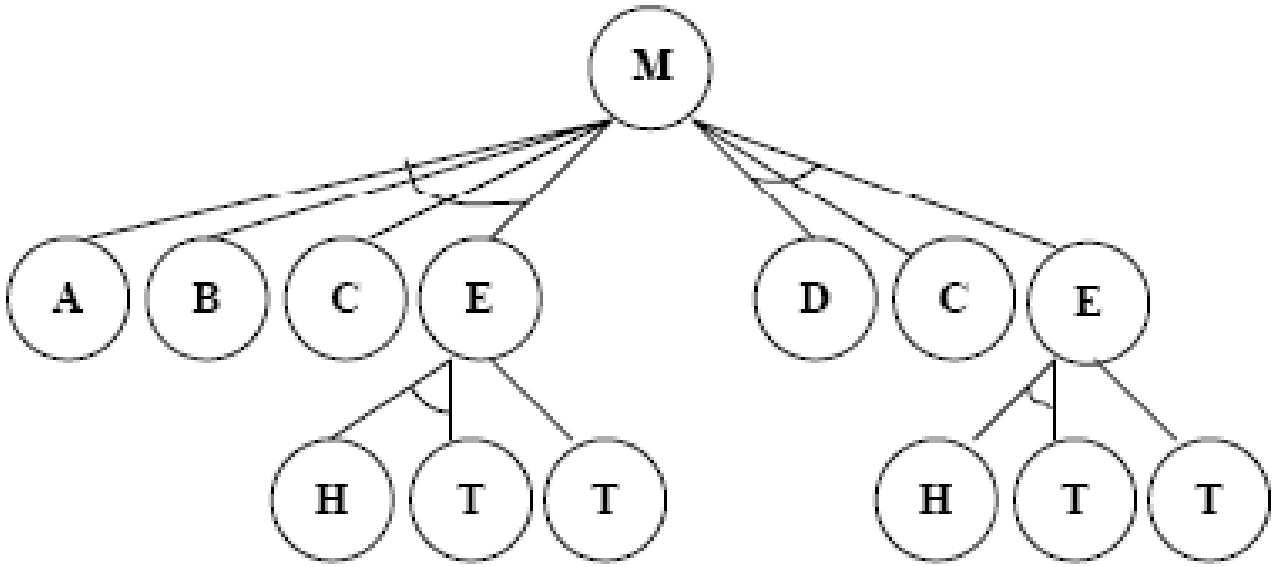
- Masalah M dicari solusinya dengan 4 kemungkinan yaitu A OR B OR C OR D.



- Masalah M hanya dapat diselesaikan dengan A AND B AND C AND D



- Contoh : Dengan menggunakan pohon AND/OR tujuan yang dicapai pada pohon di Gambar sebelumnya bisa dipersingkat hanya sampai level-2 saja.



Contoh 1 : Masalah EMBER

- Ada 2 ember masing-masing berkapasitas 4 galon (ember A) dan 3 galon (ember B). Ada pompa air yg akan digunakan untuk mengisi air pada ember tersebut. Bagaimana dapat mengisi tepat 2 galon air ke dalam ember berkapasitas 4 galon?
- Penyelesaian :

1. Identifikasi ruang keadaan (state space)
   Permasalahan ini dapat digambarkan sebagai himpunan pasangan bilangan bulat :

   x = jumlah air yg diisikan ke ember 4 galon (ember A)

   y = jumlah air yg diisikan ke ember 3 galon (ember B)

   Ruang keadaan = (x,y) sedemikian hingga $x \in \{0,1,2,3,4\}$ dan $y \in \{0,1,2,3\}$

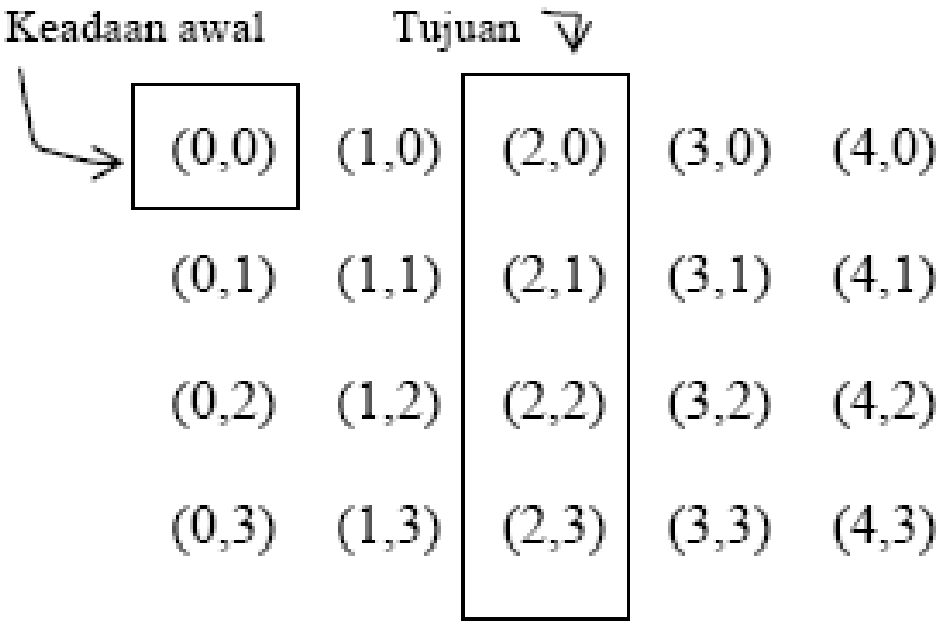
2. Keadaan awal & tujuan

Keadaan awal : kedua ember kosong = (0,0)

Tujuan : ember 4 galon berisi 2 galon air = (2,n) dengan sembarang n

3. Keadaan ember

Keadaan ember bisa digambarkan sebagai berikut :



4. Aturan-aturan

Diasumsikan kita dapat mengisi ember air itu dari pompa air, membuang air dari ember ke luar, menuangkan air dari ember yang satu ke ember yang lain.

Kita buat beberapa aturan-aturan yang dapat digambarkan sebagai berikut :

Aturan ke-	Jika	Maka
1	(x,y) $x < 4$	(4,y) Isi ember A
2	(x,y) $y < 3$	(x,3) Isi ember B
3	(x,y) $x > 0$	(x – d,y) Tuang sebagian air keluar dari ember A
4	(x,y) $y > 0$	(x,y – d) Tuang sebagian air keluar dari ember B
5	(x,y) $x > 0$	(0,y) Kosongkan ember A dengan membuang airnya
6	(x,y) $y > 0$	(x,0) Kosongkan ember B dengan membuang airnya
7	(x,y) $x+y \geq 4$ dan $y > 0$	(4,y – (4 – x)) Tuang air dari ember B ke ember A sampai ember A penuh
8	(x,y) $x+y \geq 3$ dan $x > 0$	(x – (3 – y),3) Tuang air dari ember A ke ember B sampai ember B penuh
9	(x,y) $x+y \leq 4$ dan $y > 0$	(x+y,0) Tuang seluruh air dari ember B ke ember A
10	(x,y) $x+y \leq 3$ dan $x > 0$	(0,x+y) Tuang seluruh air dari ember A ke ember B
11	(0,2)	(2,0) Tuang 2 galon air dari ember B ke ember A

5. Representasi ruang keadaan dengan pohon pelacakan
Pencarian suatu solusi dapat dilukiskan dengan menggunakan pohon. Tiap-tiap node menunjukkan satu keadaan. Jalur dari parent ke child ,menunjukkan 1 operasi. Tiap node memiliki node child yg menunjukkan keadaan yg dapat dicapai oleh parent.

Solusi yg ditemukan :

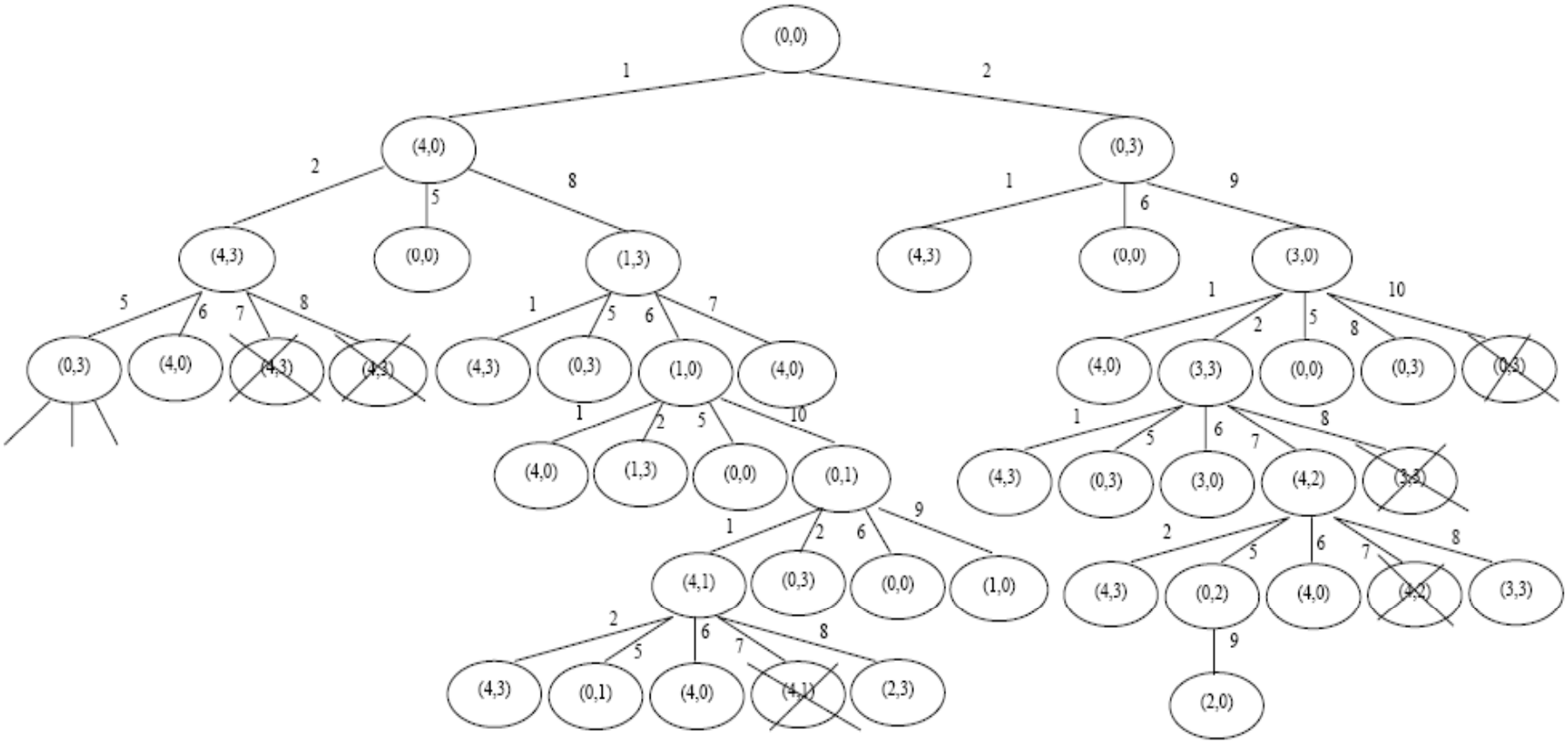
Solusi 1

Isi ember A	Isi ember B	Aturan yg dipakai
0	0	1
4	0	8
1	3	6
1	0	10
0	1	1
4	1	8
2	3	Solusi

Solusi 2

Isi ember A	Isi ember B	Aturan yg dipakai
0	0	2
0	3	9
3	0	2
3	3	7
4	2	5
0	2	9
2	0	Solusi

Representasi Ruang Keadaan untuk Kasus EMBER



Contoh 2 : Masalah PETANI, KAMBING, SERIGALA, SAYURAN, PERAHU

Seorang petani akan menyeberangkan seekor kambing,seekor serigala,sayuran dengan sebuah perahu yg melalui sungai. Perahu hanya bisa memuat petani & satu penumpang yg lain (kambing, serigala, atau sayuran). Jika ditinggalkan petani tersebut, maka sayuran dimakan kambing dan kambing akan dimakan serigala.

Penyelesaian :

1. Identifikasi ruang keadaan

   Permasalahan ini dapat dilambangkan dengan (jumlah kambing,jumlah serigala,jumlah sayuran,jumlah perahu).

Contoh : daerah asal (0,1,1,1) = daerah asal tidak ada kambing,ada serigala, ada sayuran,ada perahu

2. Keadaan awal & tujuan

Keadaan awal, pada kedua daerah :

daerah asal = (1,1,1,1)

daerah seberang = (0,0,0,0)

Keadaan tujuan, pada kedua daerah :

daerah asal = (0,0,0,0)

daerah seberang = (1,1,1,1)

3. Aturan-aturan

Aturan ke-	Aturan
1	Kambing menyeberang
2	Sayuran menyeberang
3	Serigala menyeberang
4	Kambing kembali
5	Sayuran kembali
6	Serigala kembali
7	Perahu kembali

4. Solusi yg ditemukan

Daerah asal	Daerah seberang	Aturan yg dipakai
(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 PELACAKAN / PENCARIAN

- Hal penting dalam menentukan keberhasilan sistem cerdas adalah kesuksesan dalam pencarian. Pencarian = suatu proses mencari solusi dari suatu permasalahan melalui sekumpulan kemungkinan ruang keadaan (state space). Ruang keadaan = merupakan suatu ruang yang berisi semua keadaan yang mungkin.
- Untuk mengukur perfomansi metode pencarian, terdapat empat kriteria yang dapat digunakan :
  - Completeness : apakah metode tsb menjamin penemuan solusi jika solusinya memang ada?
  - Time complexity : berapa lama waktu yang diperlukan?
  - Space complexity : berapa banyak memori yg diperlukan
  - Optimality : apakah metode tsb menjamin menemukan solusi yg terbaik jika terdapat beberapa solusi berbeda?

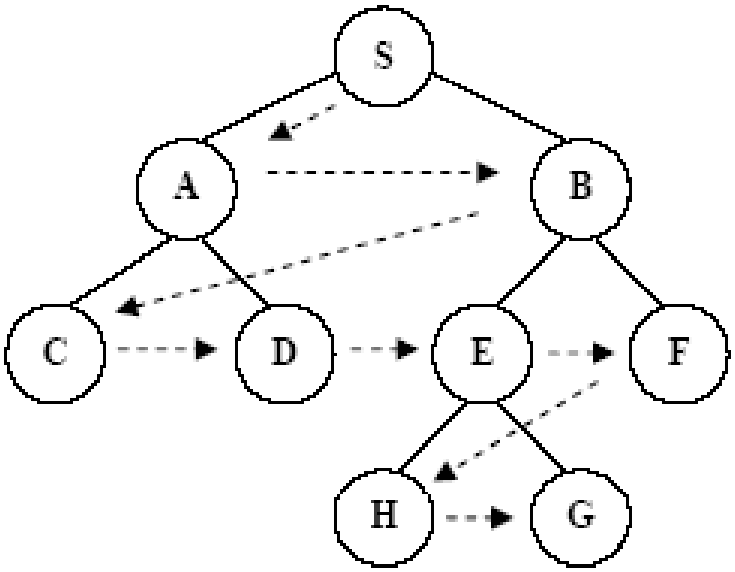
Teknik pencarian

A. Pencarian buta (blind search) : tidak ada informasi awal yang digunakan dalam proses pencarian
   1. Pencarian melebar pertama (Breadth – First Search)
   2. Pencarian mendalam pertama (Depth – First Search)

B. Pencarian terbimbing (heuristic search) : adanya informasi awal yang digunakan dalam proses pencarian
   1. Pendakian Bukit (Hill Climbing)
   2. Pencarian Terbaik Pertama (Best First Search)

A. Pencarian Buta (blind search)

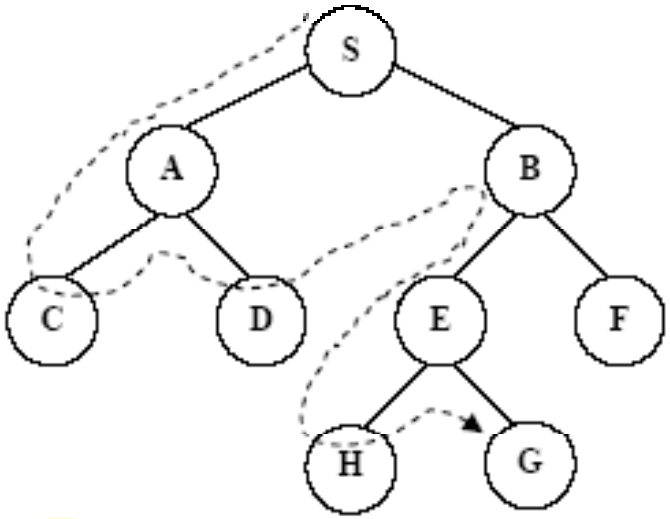
1. Breadth – First Search Semua node pada level n akan dikunjungi terlebih dahulu sebelum mengunjungi node-node pada level n+1. Pencarian dimulai dari node akar terus ke level 1 dari kiri ke kanan, kemudian berpindah ke level berikutnya dari kiri ke kanan hingga solusi ditemukan.



- Keuntungan :
  - tidak akan menemui jalan buntu, menjamin ditemukannya solusi (jika solusinya memang ada) dan solusi yang ditemukan pasti yang paling baik
  - jika ada 1 solusi, maka breadth – first search akan menemukannya,jika ada lebih dari 1 solusi, maka solusi minimum akan ditemukan.
  - Kesimpulan : complete dan optimal
- Kelemahan :
  - membutuhkan memori yang banyak, karena harus menyimpan semua simpul yang pernah dibangkitkan. Hal ini harus dilakukan agar BFS dapat melakukan penelusuran simpul-simpul sampai di level bawah
  - membutuhkan waktu yang cukup lama

2. Depth – First Search

- Pencarian dilakukan pada suatu simpul dalam setiap level dari yang paling kiri.
- Jika pada level yang paling dalam tidak ditemukan solusi, maka pencarian dilanjutkan pada simpul sebelah kanan dan simpul yang kiri dapat dihapus dari memori.
- Jika pada level yang paling dalam tidak ditemukan solusi, maka pencarian dilanjutkan pada level sebelumnya. Demikian seterusnya sampai ditemukan solusi.

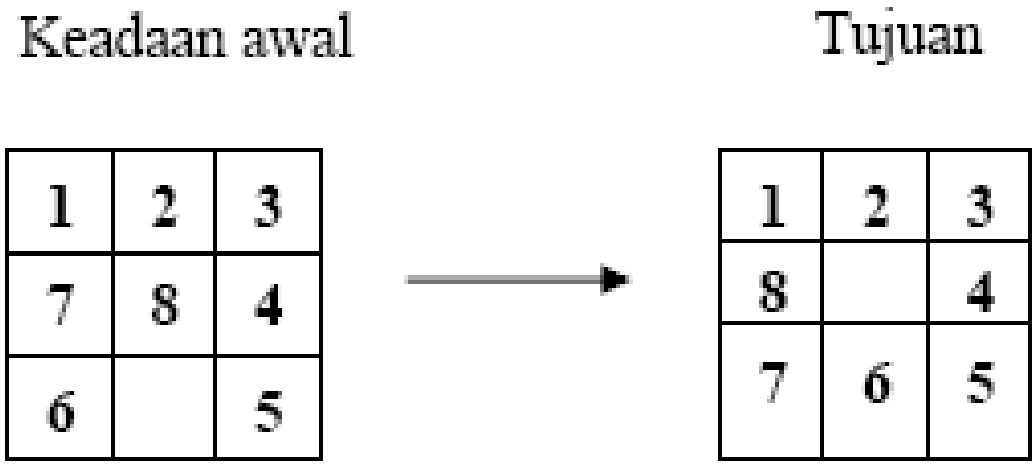


- Keuntungan :
  - membutuhkan memori relatif kecil, karena hanya node-node pada lintasan yang aktif saja yang disimpan
  - Secara kebetulan, akan menemukan solusi tanpa harus menguji lebih banyak lagi dalam ruang keadaan, jadi jika solusi yang dicari berada pada level yang dalam dan paling kiri, maka → **waktu cepat**
- Kelemahan :
  - Memungkinkan tidak ditemukannya tujuan yang diharapkan, karena jika pohon yang dibangkitkan mempunyai level yang sangat dalam (tak terhingga) → **tidak complete** karena tidak ada jaminan menemukan solusi
  - Hanya mendapat 1 solusi pada setiap pencarian, karena jika terdapat lebih dari satu solusi yang sama tetapi berada pada level yang berbeda, maka DFS tidak menjamin untuk menemukan solusi yang paling baik → **tidak optimal**.

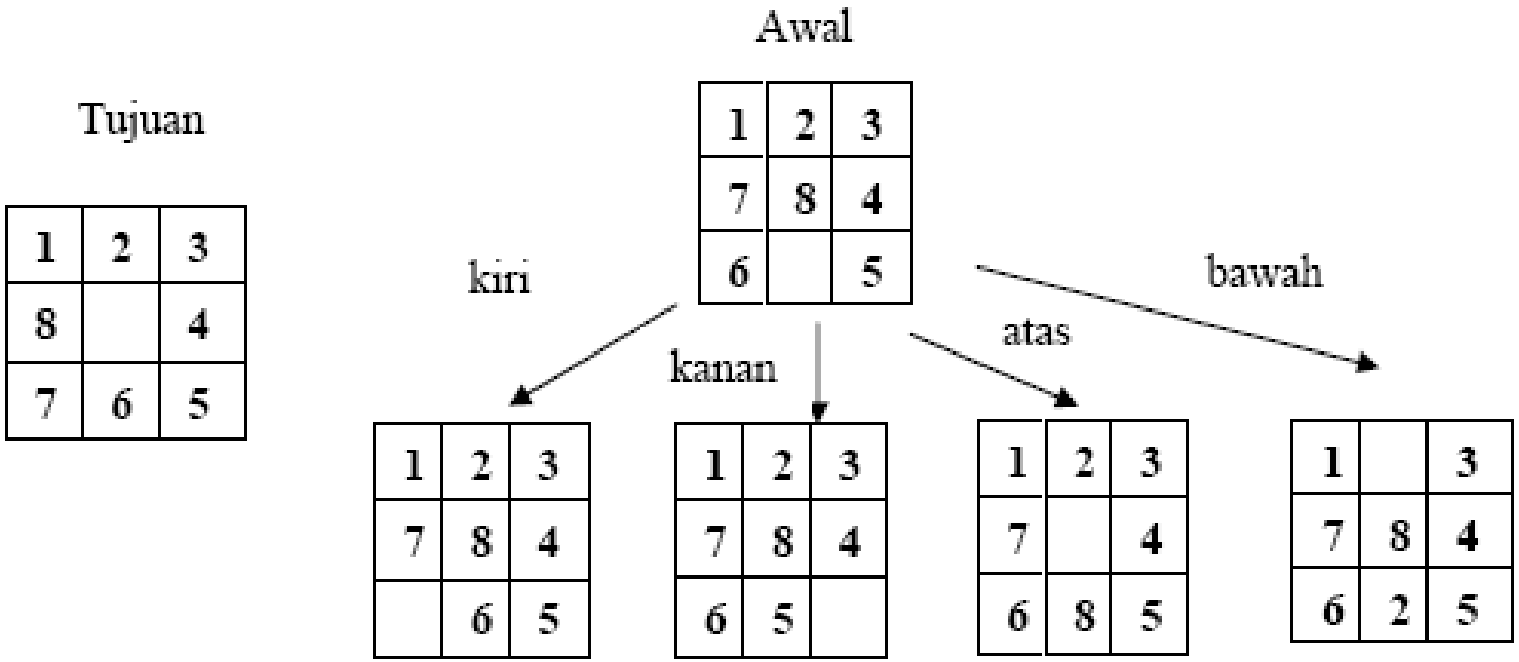
B. Heuristic Search

- Pencarian buta tidak selalu dapat diterapkan dengan baik, hal ini disebabkan waktu aksesnya yang cukup lama & besarnya memori yang diperlukan. Untuk masalah dengan ruang masalah yang besar, teknik pencarian buta bukan metode yang baik karena keterbatasan kecepatan komputer dan memori.
- Metode heuristic search diharapkan bisa menyelesaikan permasalahan yang lebih besar. Metode heuristic search menggunakan suatu fungsi yang menghitung biaya perkiraan (estimasi) dari suatu simpul tertentu menuju ke simpul tujuan → disebut fungsi heuristic Aplikasi yang menggunakan fungsi heuristic : Google, Deep Blue Chess Machine

- Misal kasus 8-puzzle. Ada 4 operator yang dapat digunakan untuk menggerakkan dari satu keadaan ke keadaan yang baru
  1. Ubin kosong digeser ke kiri
  2. Ubin kosong digeser ke kanan
  3. Ubin kosong digeser ke bawah
  4. Ubin kosong digeser ke atas

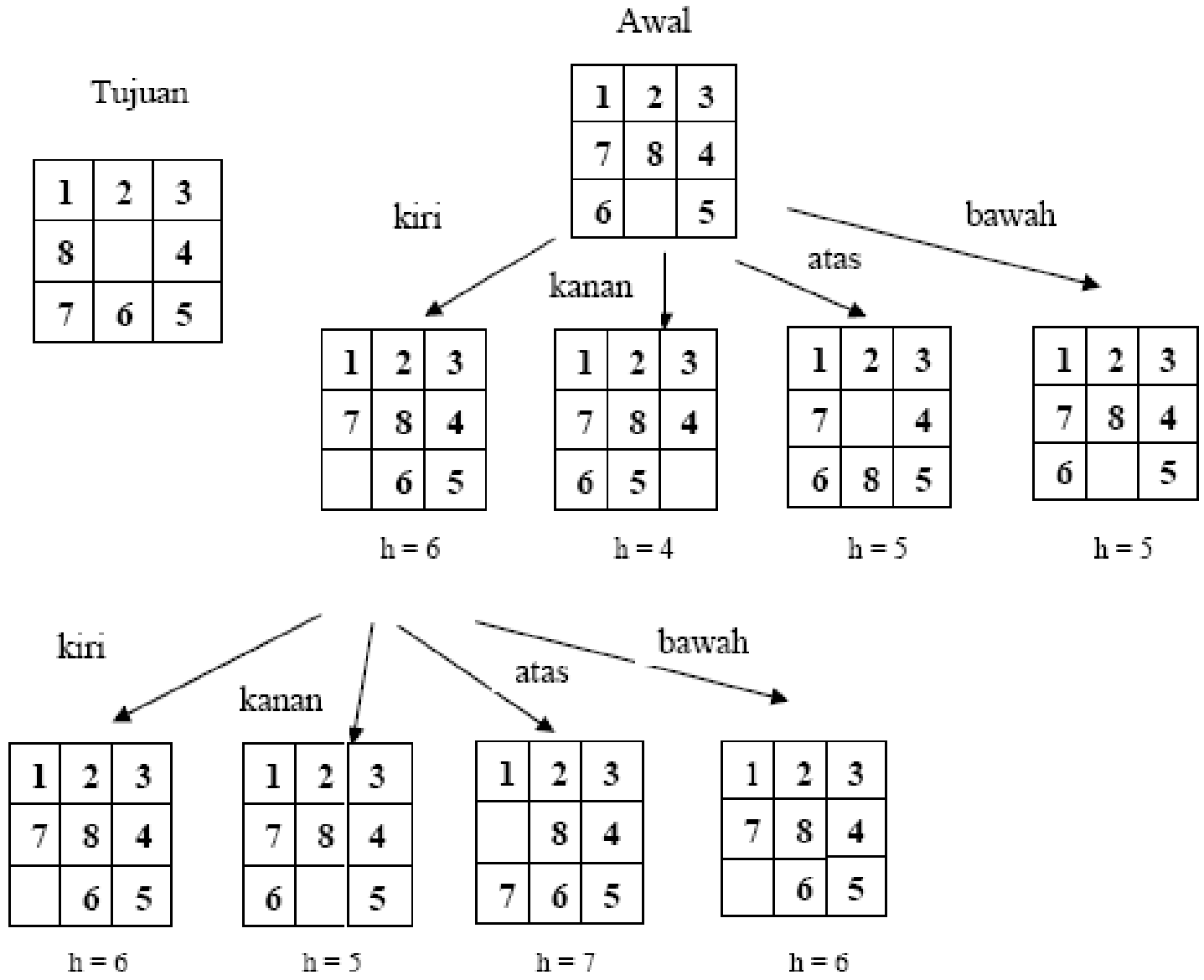


Langkah awal

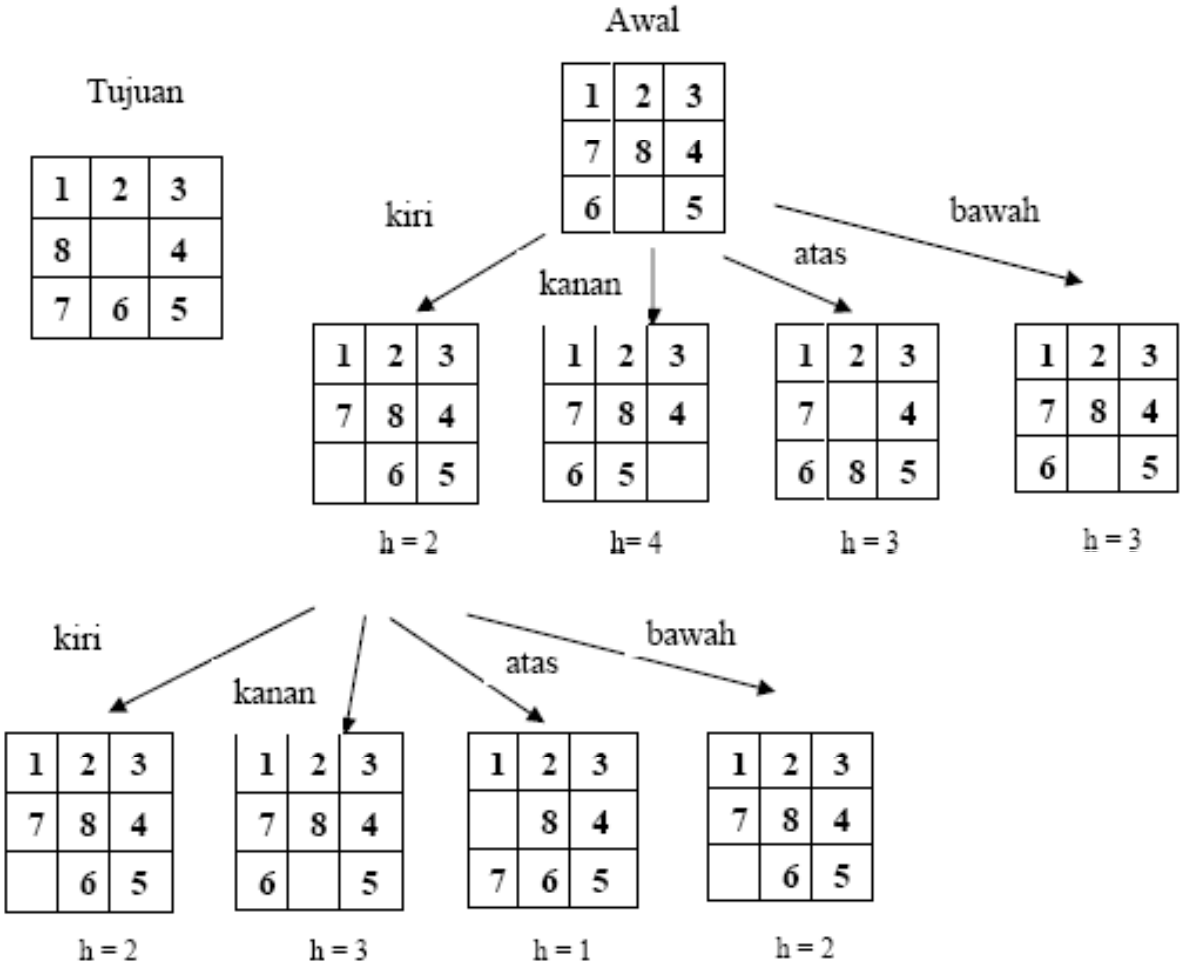


Pada pencarian heuristik perlu diberikan informasi khusus, yaitu :

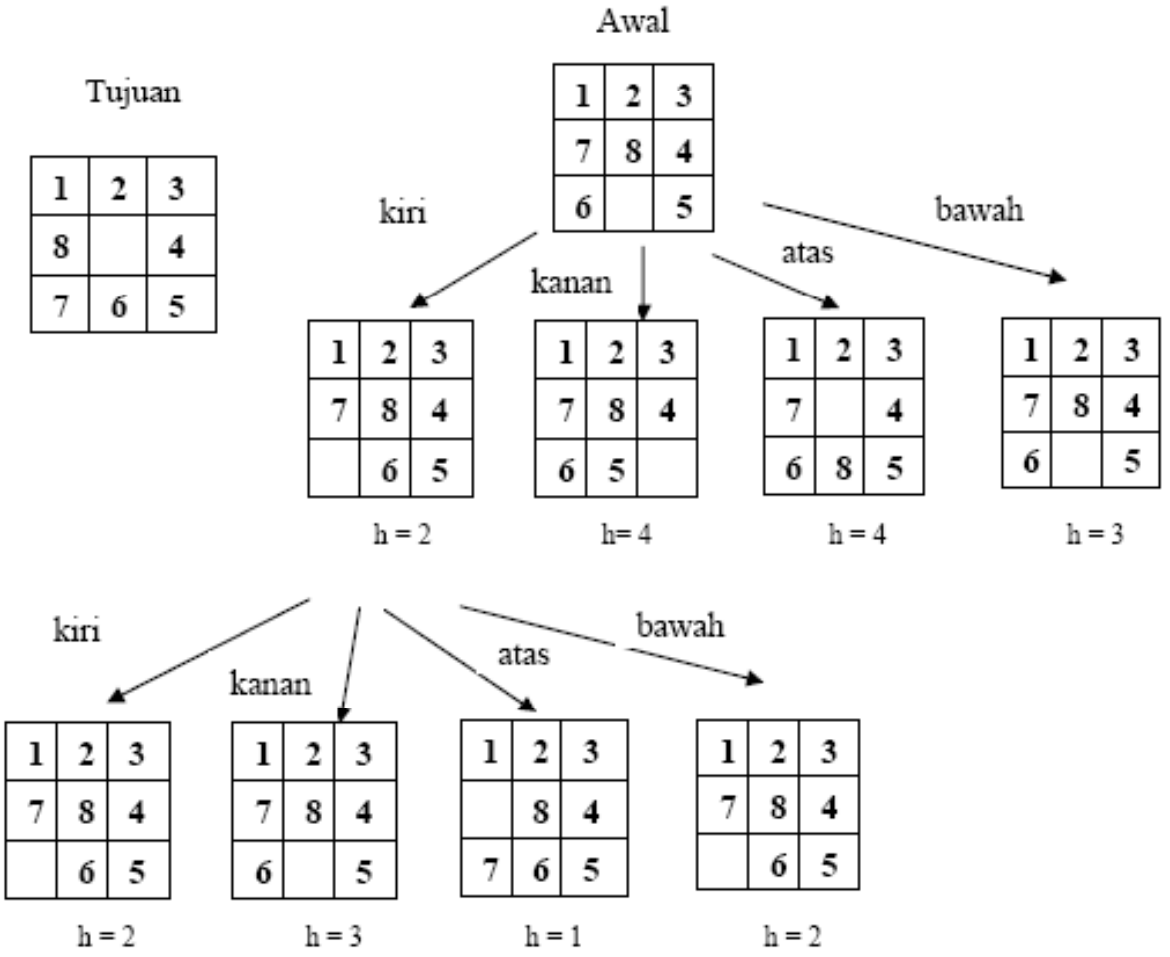
- Untuk jumlah ubin yang menempati posisi yang benar Jumlah yang lebih tinggi adalah yang lebih diharapkan (lebih baik)


Tujuan
1 2 3
8 4
7 6 5
Awal
1 2 3
7 8 4
6 5
kiri
kanan
atas
bawah
1 2 3
7 8 4
6 5
h = 6
1 2 3
7 8 4
6 5
h = 4
1 2 3
7 4
6 8 5
h = 5
1 2 3
7 8 4
6 5
h = 5
kiri
kanan
atas
bawah
1 2 3
7 8 4
6 5
h = 6
1 2 3
7 8 4
6 5
h = 5
1 2 3
8 4
7 6 5
h = 7
1 2 3
7 8 4
6 5
h = 6

- Untuk jumlah ubin yang menempati posisi yang salah Jumlah yang lebih kecil adalah yang diharapkan (lebih baik)



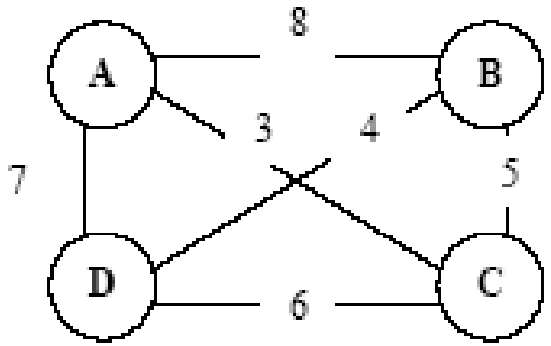
- Menghitung total gerakan yang diperlukan untuk mencapai tujuan Jumlah yang lebih kecil adalah yang diharapkan (lebih baik)



1. Hill Climbing

- Contoh : Traveling Salesman Problem (TSP)
  Seorang salesman ingin mengunjungi n kota. Jarak antara tiap-tiap kota sudah diketahui. Kita ingin mengetahui rute terpendek dimana setiap kota hanya boleh dikunjungi tepat 1 kali. Misal ada 4 kota dengan jarak antara tiap-tiap kota seperti berikut ini :

Solusi – solusi yang mungkin dengan menyusun kota-kota dalam urutan abjad, misal :
A – B – C – D : dengan panjang lintasan (=19)
A – B – D – C : (=18)
A – C – B – D : (=12)
A – C – D – B : (=13)
dst



a. Metode Simple Hill Climbing

- Ruang keadaan berisi semua kemungkinan lintasan yang mungkin. Operator digunakan untuk menukar posisi kota-kota yang bersebelahan. Fungsi heuristik yang digunakan adalah panjang lintasan yang terjadi. Operator yang akan digunakan adalah menukar urutan posisi 2 kota dalam 1 lintasan. Bila ada n kota, dan ingin mencari kombinasi lintasan dengan menukar posisi urutan 2 kota, maka akan didapat sebanyak :

$$\frac{n!}{2!(n-2)!} = \frac{4!}{2!(4-2)!} = 6 \text{ kombinasi}$$

- Keenam kompbinasi ini akan dipakai semuanya sebagai operator, yaitu :

  Tukar 1,2 = menukar urutan posisi kota ke – 1 dengan kota ke – 2

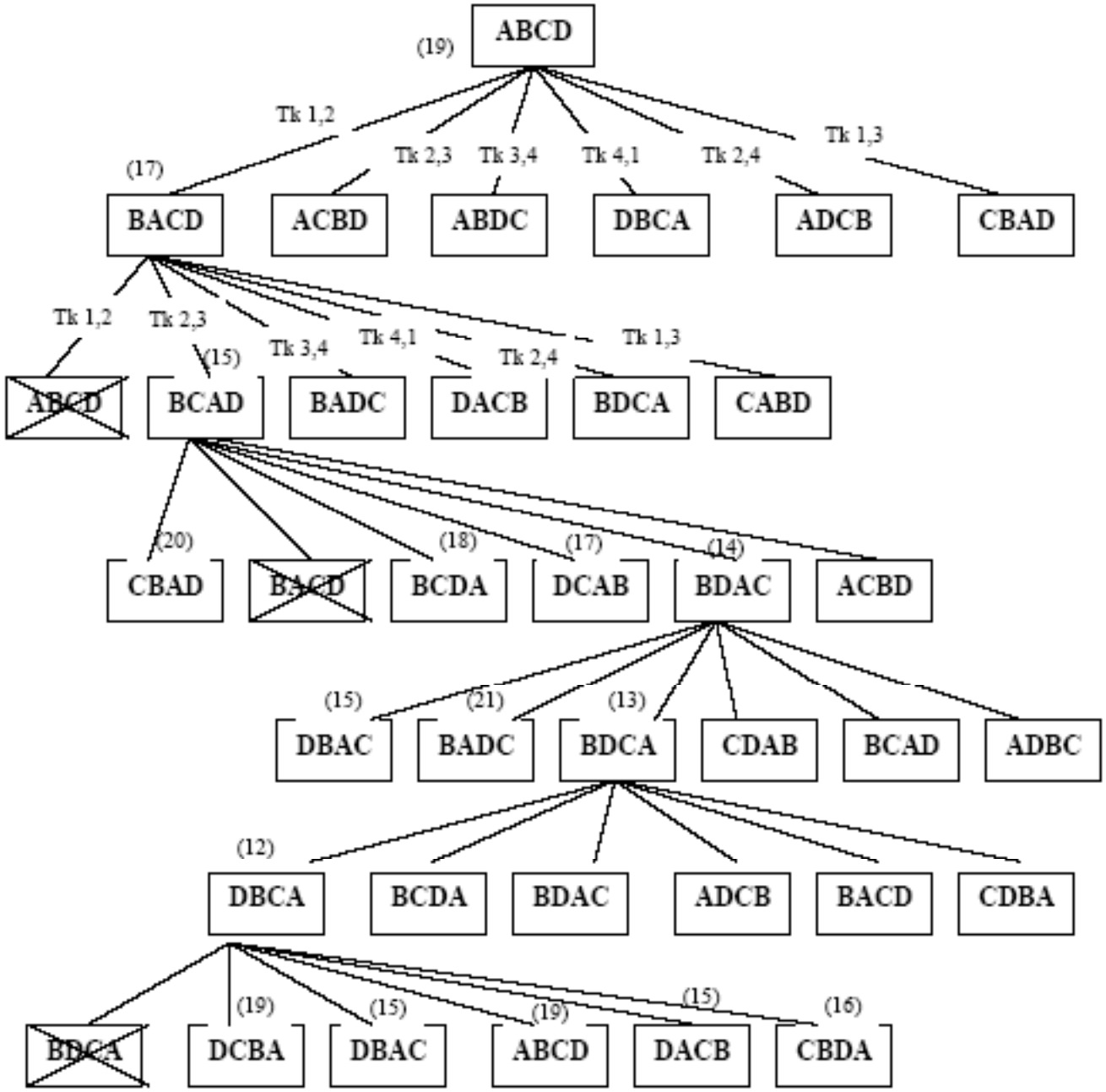
  Tukar 2,3 = menukar urutan posisi kota ke – 2 dengan kota ke – 3

  Tukar 3,4 = menukar urutan posisi kota ke – 3 dengan kota ke – 4

  Tukar 4,1 = menukar urutan posisi kota ke – 4 dengan kota ke – 1

  Tukar 2,4 = menukar urutan posisi kota ke – 2 dengan kota ke – 4

  Tukar 1,3 = menukar urutan posisi kota ke – 1 dengan kota ke – 3

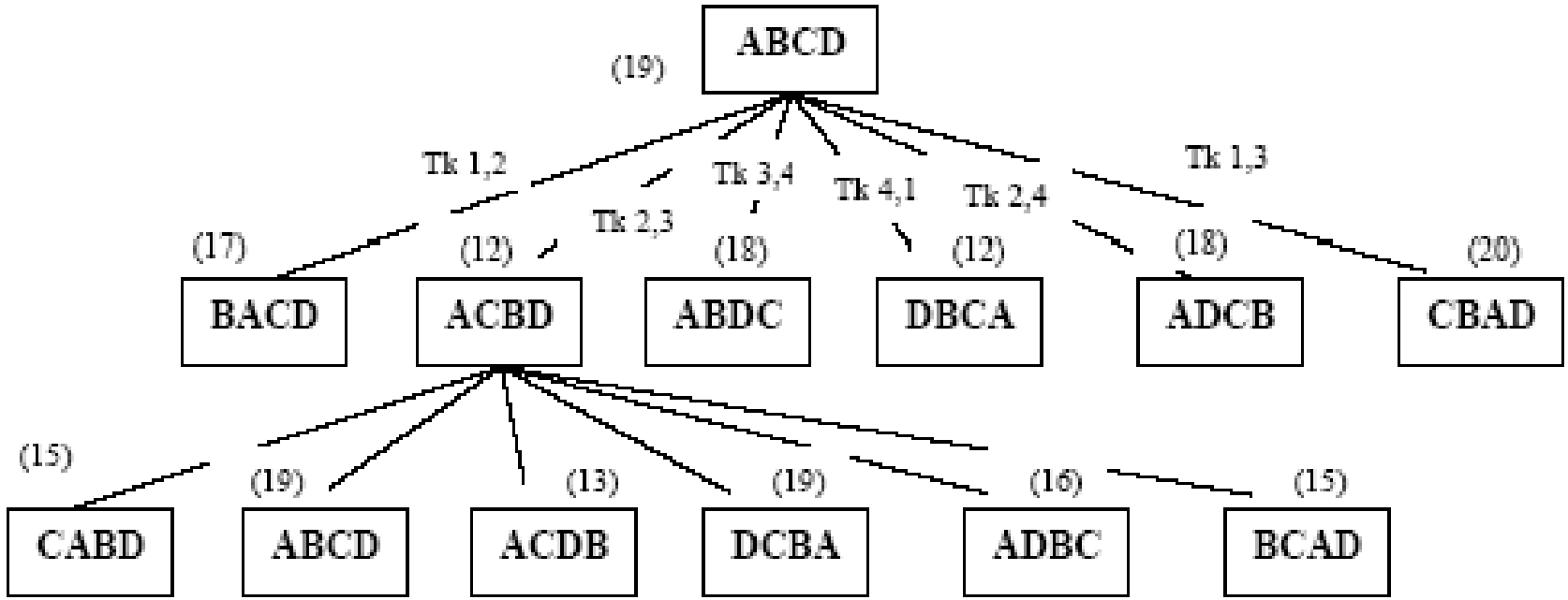

(19)
ABCD
Tk 1,2
Tk 2,3
Tk 3,4
Tk 4,1
Tk 2,4
Tk 1,3
(17)
BACD
ACBD
ABDC
DBCA
ADCB
CBAD
Tk 1,2
Tk 2,3
Tk 3,4
Tk 4,1
Tk 2,4
Tk 1,3
(15)
ABCD
BCAD
BADC
DACB
BDCA
CABD
(20)
(18)
(17)
(14)
CBAD
BACD
BCDA
DCAB
BDAC
ACBD
(15)
(21)
(13)
DBAC
BADC
BDCA
CDAB
BCAD
ADBC
(12)
DBCA
BCDA
BDAC
ADCB
BACD
CDBA
(19)
(15)
(19)
(15)
(16)
BDCA
DCBA
DBAC
ABCD
DACB
CBDA

- Keadaan awal, lintasan ABCD (=19).
- Level pertama, hill climbing mengunjungi BACD (=17), BACD (=17) < ABCD (=19), sehingga BACD menjadi pilihan selanjutnya dengan operator Tukar 1,2
- Level kedua, mengunjungi ABCD, karena operator Tukar 1,2 sudah dipakai BACD, maka pilih node lain yaitu BCAD (=15), BCAD (=15) < BACD (=17)
- Level ketiga, mengunjungi CBAD (=20), CBAD (=20) > BCAD (=15), maka pilih node lain yaitu BCDA (=18), pilih node lain yaitu DCAB (=17), pilih node lain yaitu BDAC (=14), BDAC (=14) < BCAD (=15)
- Level keempat, mengunjungi DBAC (=15), DBAC(=15) > BDAC (=14), maka pilih node lain yaitu BADC (=21), pilih node lain yaitu BDCA (=13), BDCA (=13) < BDAC (=14)
- Level kelima, mengunjungi DBCA (=12), DBCA (=12) < BDCA (=13)
- Level keenam, mengunjungi BDCA, karena operator Tukar 1,2 sudah dipakai DBCA, maka pilih node lain yaitu DCBA, pilih DBAC, pilih ABCD, pilih DACB, pilih CBDA
- Karena sudah tidak ada node yang memiliki nilai heuristik yang lebih kecil dibanding nilai heuristik DBCA, maka **node DBCA (=12) adalah lintasan terpendek (SOLUSI)**

b. Metode Steepest – Ascent Hill Climbing

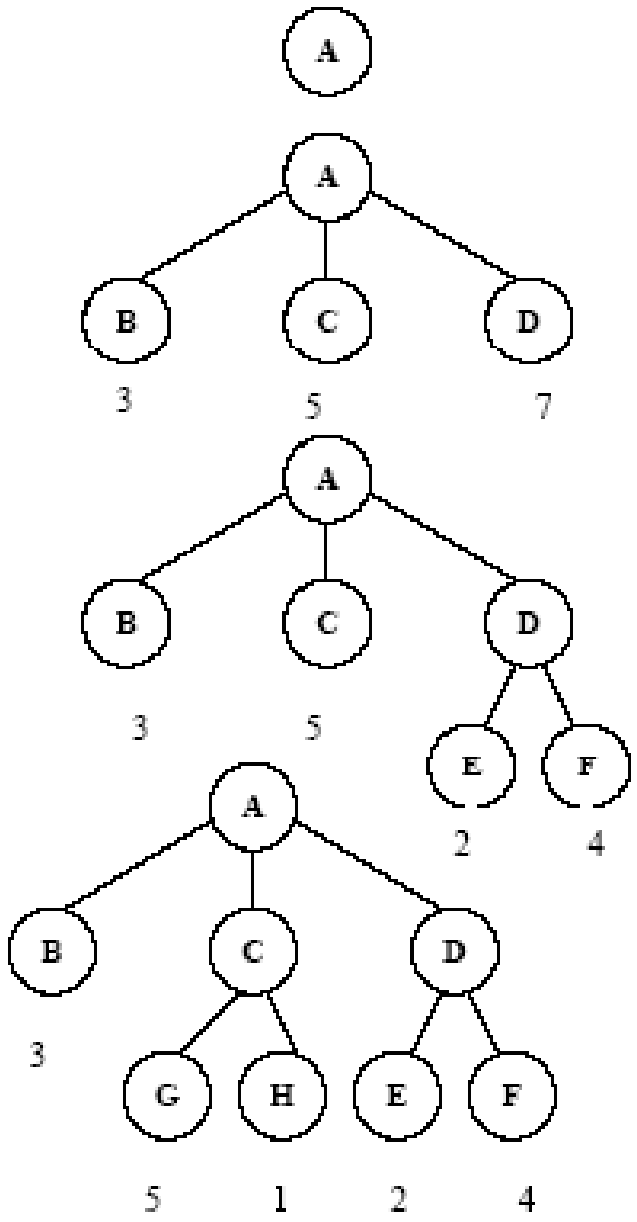
- Steepest – ascent hill climbing hampir sama dengan simple – ascent hill climbing, hanya saja gerakan pencarian tidak dimulai dari kiri, tetapi berdasarkan nilai heuristik terbaik.
- Keadaan awal, lintasan ABCD (=19).
- Level pertama, hill climbing memilih nilai heuristik terbaik yaitu ACBD (=12) sehingga ACBD menjadi pilihan selanjutnya.

- Level kedua, hill climbing memilih nilai heuristik terbaik, karena nilai heuristik lebih besar dibanding ACBD, maka hasil yang diperoleh lintasannya tetap ACBD (=12)



2. Best First Search

- Metode best first search merupakan kombinasi dari metode depth first search & breadth first search dengan mengambil kelebihan dari kedua metode tersebut. Hill climbing tidak diperbolehkan untuk kembali ke node yang lebih rendah meskipun node tersebut memiliki nilai heuristik lebih baik. Pada best first search, pencarian diperbolehkan mengunjungi node yang lebih rendah, jika ternyata node di level lebih tinggi memiliki nilai heuristik lebih buruk. Untuk mengimplementasikan metode ini, dibutuhkan 2 antrian yang berisi node-node, yaitu :
- OPEN : berisi node-node yang sudah dibangkitkan, sudah memiliki fungsi heuristik namun belum diuji. Umumnya berupa antrian berprioritas yang berisi elemen-elemen dengan nilai heuristik tertinggi.
- CLOSED : berisi node-node yang sudah diuji.



Antrian OPEN

[A]

[D C B]

[C F B E]

[G F B E H]

Diasumsikan node dengan nilai yang lebih besar memiliki nilai evaluasi yang lebih baik. Pada keadaan awal, antrian berisi A. Pengujian dilakukan di level pertama, node D memiliki nilai terbaik, sehingga menempati antrian pertama, disusul dengan C dan B. Node D memiliki cabang E dan F yang masing-masing bernilai 2 & 4. Dengan demikian C merupakan pilihan terbaik dengan menempati antrian pertama. Demikian seterusnya.