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 tersebut **menjamin penemuan solusi** jika solusinya memang ada?
- Time complexity : berapa lama **waktu** yang diperlukan?
- Space complexity : berapa banyak **memori** yang diperlukan
- Optimality : apakah metode tersebut menjamin menemukan **solusi yang terbaik** jika terdapat beberapa solusi berbeda?

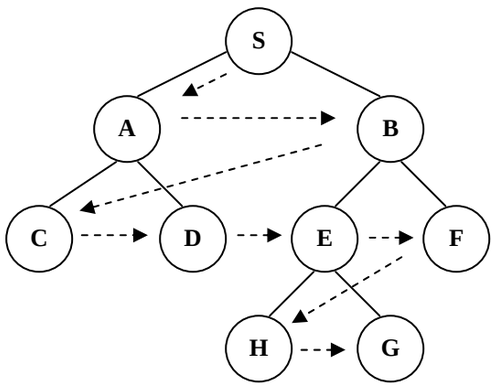
Teknik pencarian :

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 :

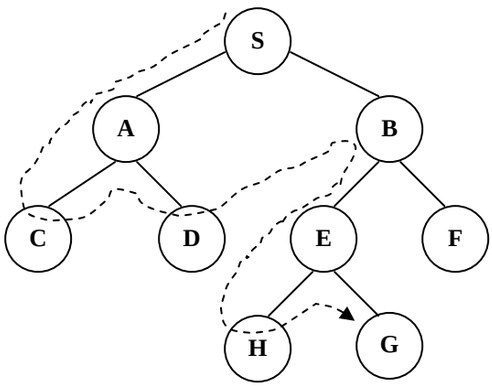
- tidak akan menemui jalan buntu, menjamin ditemukannya solusi (jika solusinya memang ada) dan solusi yang ditemukan pasti yang paling baik
- jika ada 1 solusi, maka breadth – first search akan menemukannya,jika ada lebih dari 1 solusi, maka solusi minimum akan ditemukan.
- Kesimpulan : **complete** 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 DFS akan menemukannya dengan cepat → **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

Keadaan awal

1	2	3
7	8	4
6		5

→

Tujuan

1	2	3
8		4
7	6	5

Langkah awal

Awal

1	2	3
7	8	4
6		5

kiri

1	2	3
7	8	4
	6	5

kanan

1	2	3
7	8	4
6	5	

atas

1	2	3
7		4
6	8	5

bawah

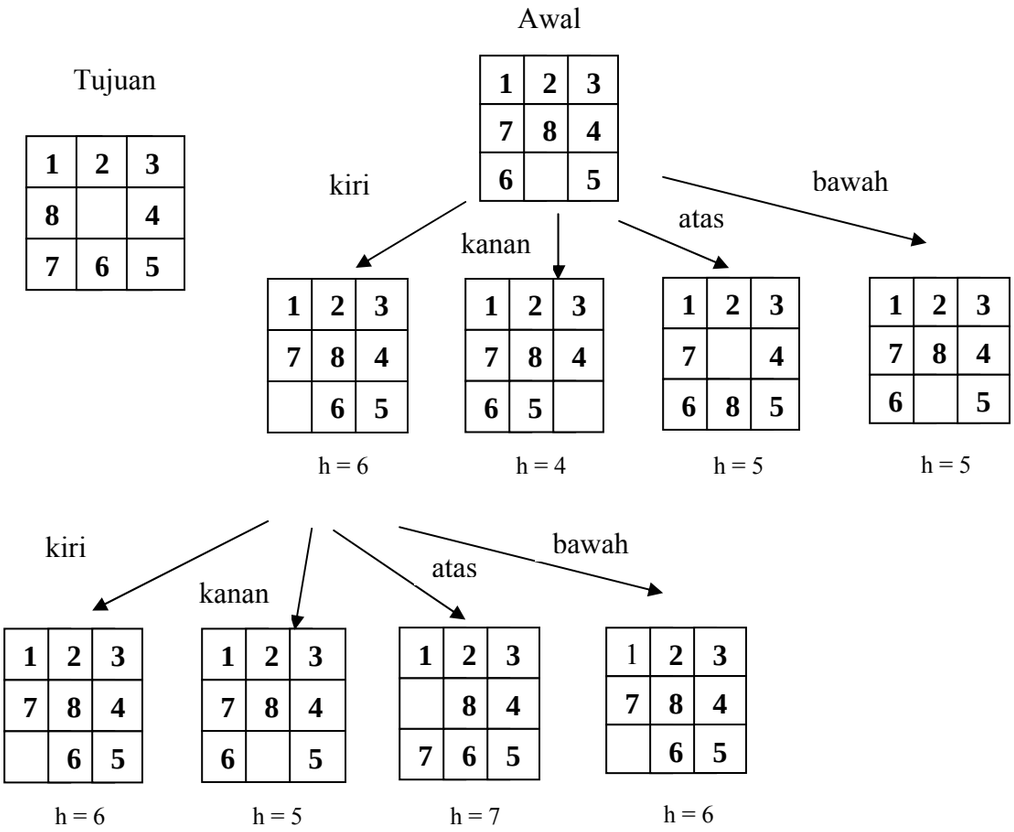
1		3
7	8	4
6	2	5

Tujuan

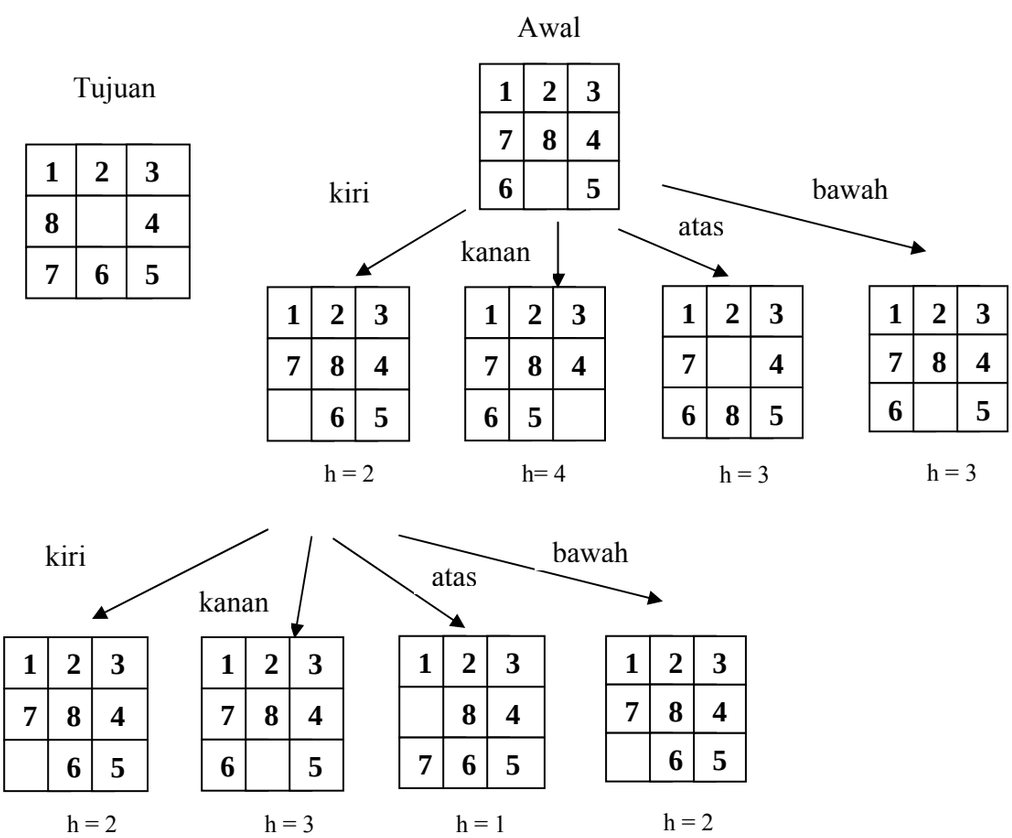
1	2	3
8		4
7	6	5

Pada pencarian heuristik perlu diberikan informasi khusus, yaitu :

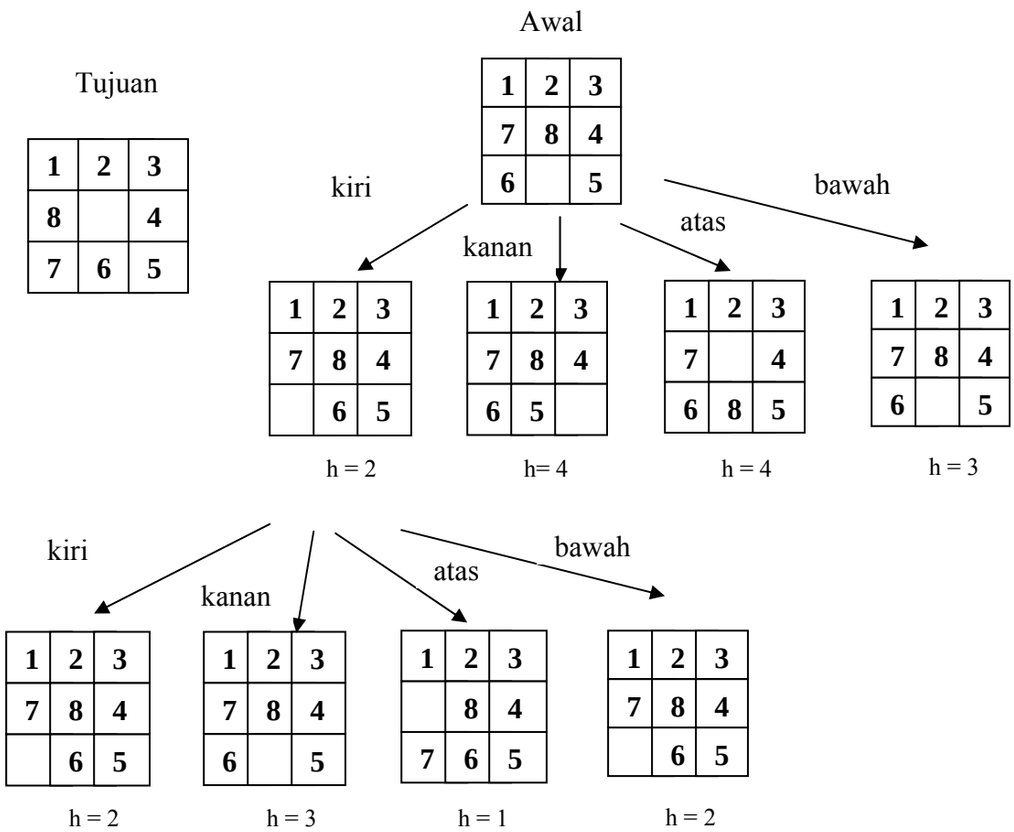
- Untuk jumlah ubin yang menempati posisi yang benar
  Jumlah yang lebih tinggi adalah yang lebih diharapkan (lebih baik)



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

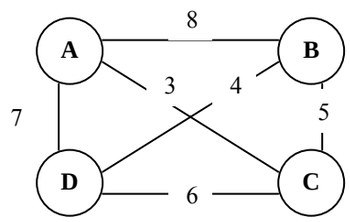


♦ Menghitung total gerakan yang diperlukan untuk mencapai tujuan
Jumlah yang lebih kecil adalah yang diharapkan (lebih baik)



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

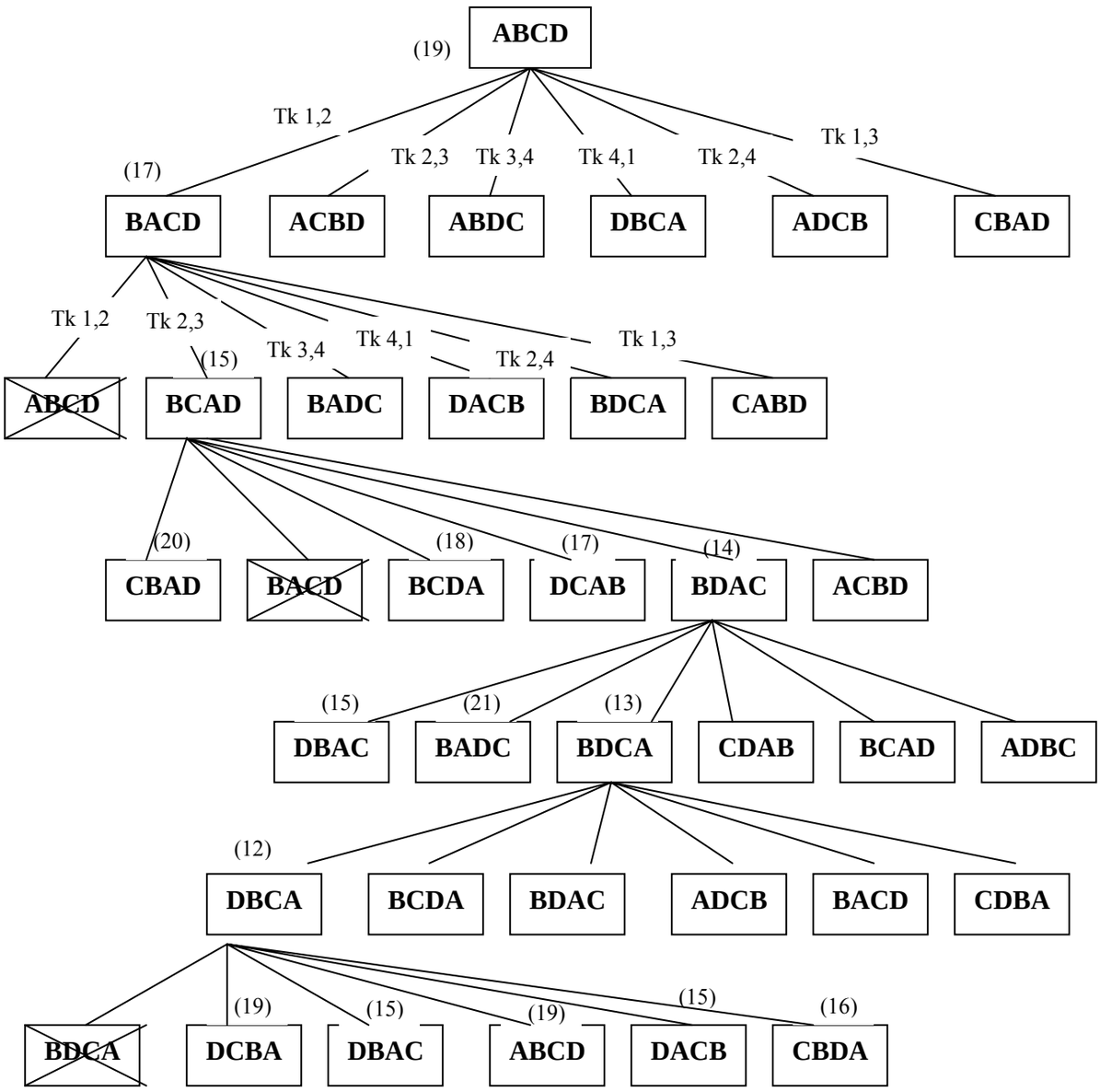
a. Metode simple hill climbing

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 :

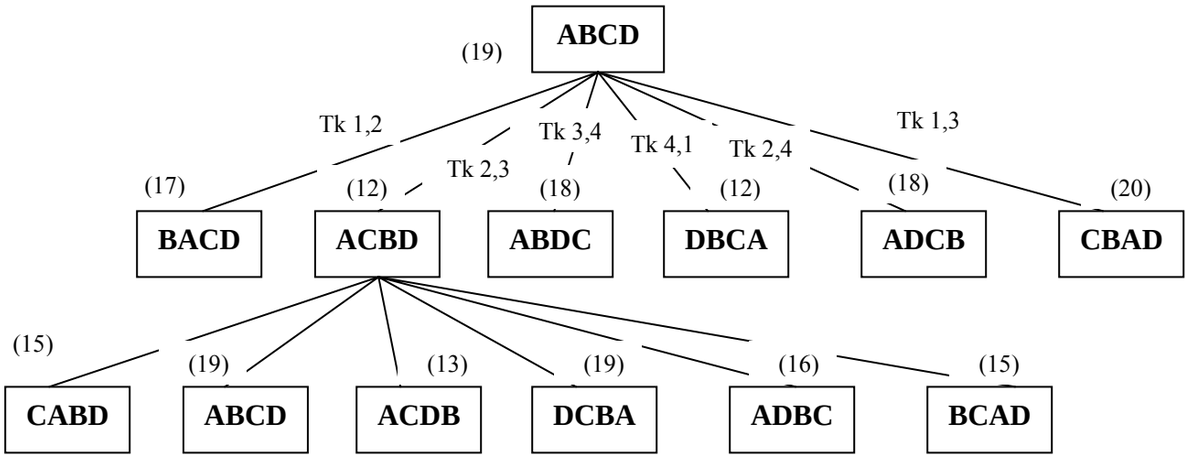
Tukar 1,2 = menukar urutan posisi kota ke – 1 dengan kota ke – 2
Tukar 2,3 = menukar urutan posisi kota ke – 2 dengan kota ke – 3
Tukar 3,4 = menukar urutan posisi kota ke – 3 dengan kota ke – 4
Tukar 4,1 = menukar urutan posisi kota ke – 4 dengan kota ke – 1
Tukar 2,4 = menukar urutan posisi kota ke – 2 dengan kota ke – 4
Tukar 1,3 = menukar urutan posisi kota ke – 1 dengan kota ke – 3



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.

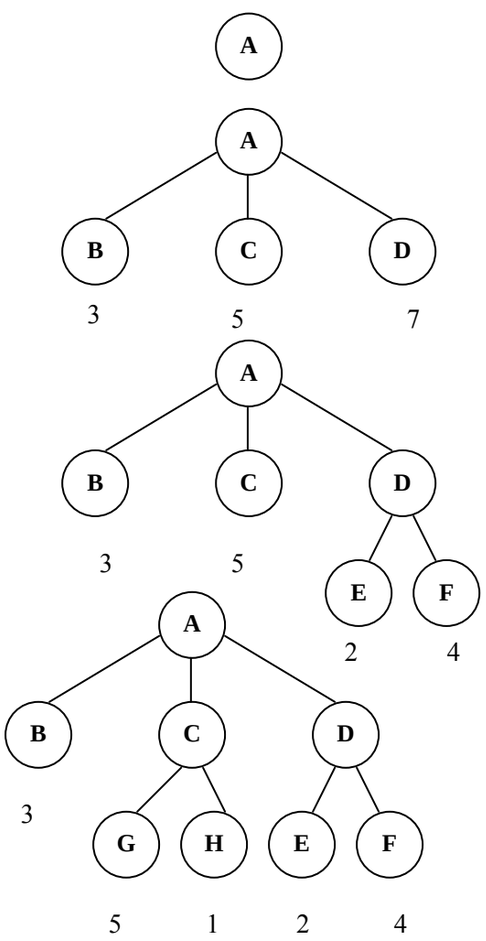
Level kedua, hill climbing memilih nilai heuristik terbaik, karena nilai heuristik lebih besar dibanding ACBD, maka hasil yang diperoleh lintasannya tetap ACBD (=12)

2. Best First Search

Metode best first search merupakan kombinasi dari metode depth first search dan breadth first search dengan mengambil kelebihan dari kedua metode tersebut. Hill climbing tidak diperbolehkan untuk kembali ke node pada lebih rendah meskipun node tersebut memiliki nilai heuristik lebih baik. Pada best first search, pencarian diperbolehkan mengunjungi node di 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.