KECERDASAN BUATAN

METODE HEURISTIK / HEURISTIC SEARCH

ERWIEN TJIPTA WIJAYA, ST., M.KOM

KERANGKA MASALAH

- Generate And Test
- Hill Climbing
- Best First Search

PENCARIAN HEURISTIK

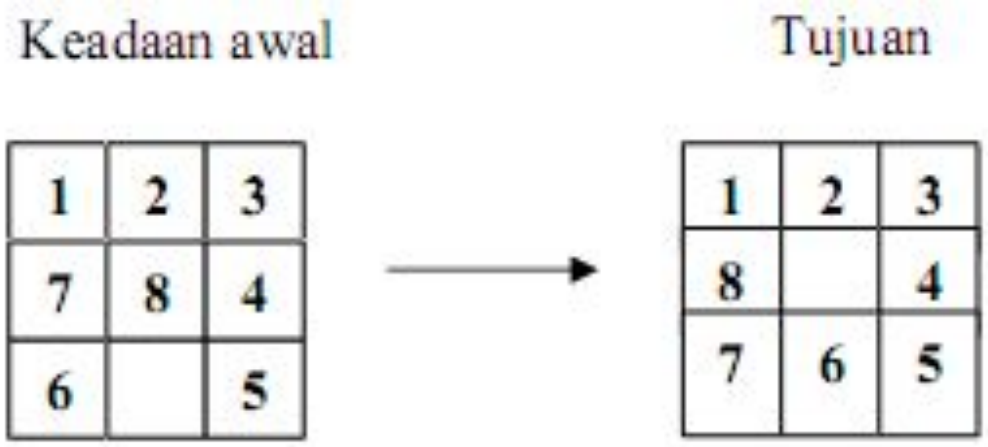
- Kelemahan blind search :

1. Waktu akses lama
2. Memori yang dibutuhkan besar
3. Ruang masalah besar – tidak cocok – karena keterbasan kecepatan komputer dan memori

- Pencarian heuristik : Menggunakan suatu fungsi yang menghitung biaya perkiraan / estimasi dari suatu simpul tertentu menuju ke simpul tujuan (disebut fungsi heuristik)

CONTOH

- Kasus 8 Puzzle :
  - Ada 4 Operator :
    1. Ubin kosong digeser ke kiri
    2. Ubin kosong digeser ke kanan
    3. Ubin kosong digeser ke bawah
    4. Ubin kosong digeser ke atas

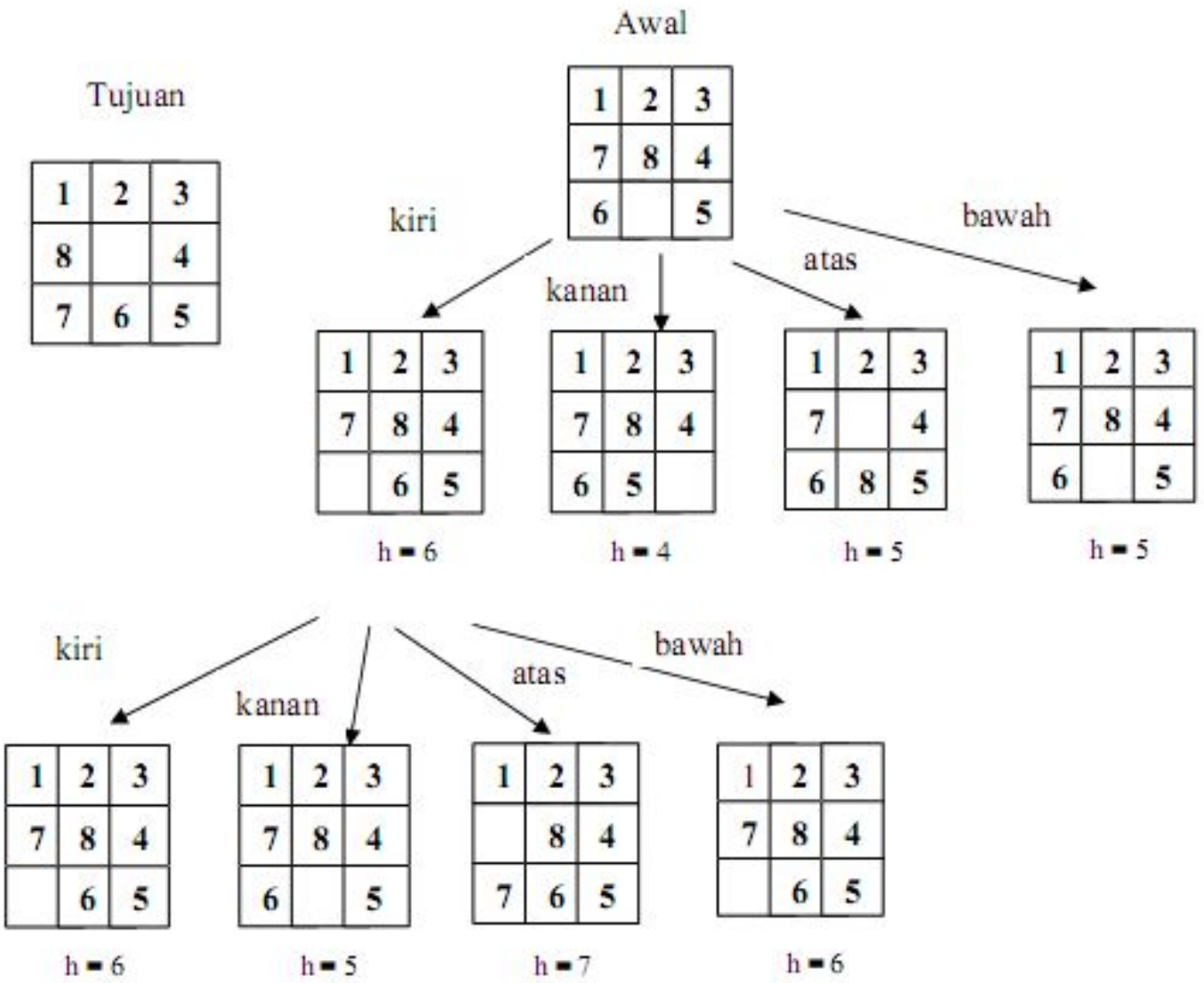


CONTOH : MEMILIH NILAI YANG BESAR

- Diberikan data informasi khusus :

1. Untuk jumlah ubin yang menempati posisi yang benar
2. Jumlah yang lebih tinggi adalah yang lebih diharapkan (lebih baik)

CONTOH

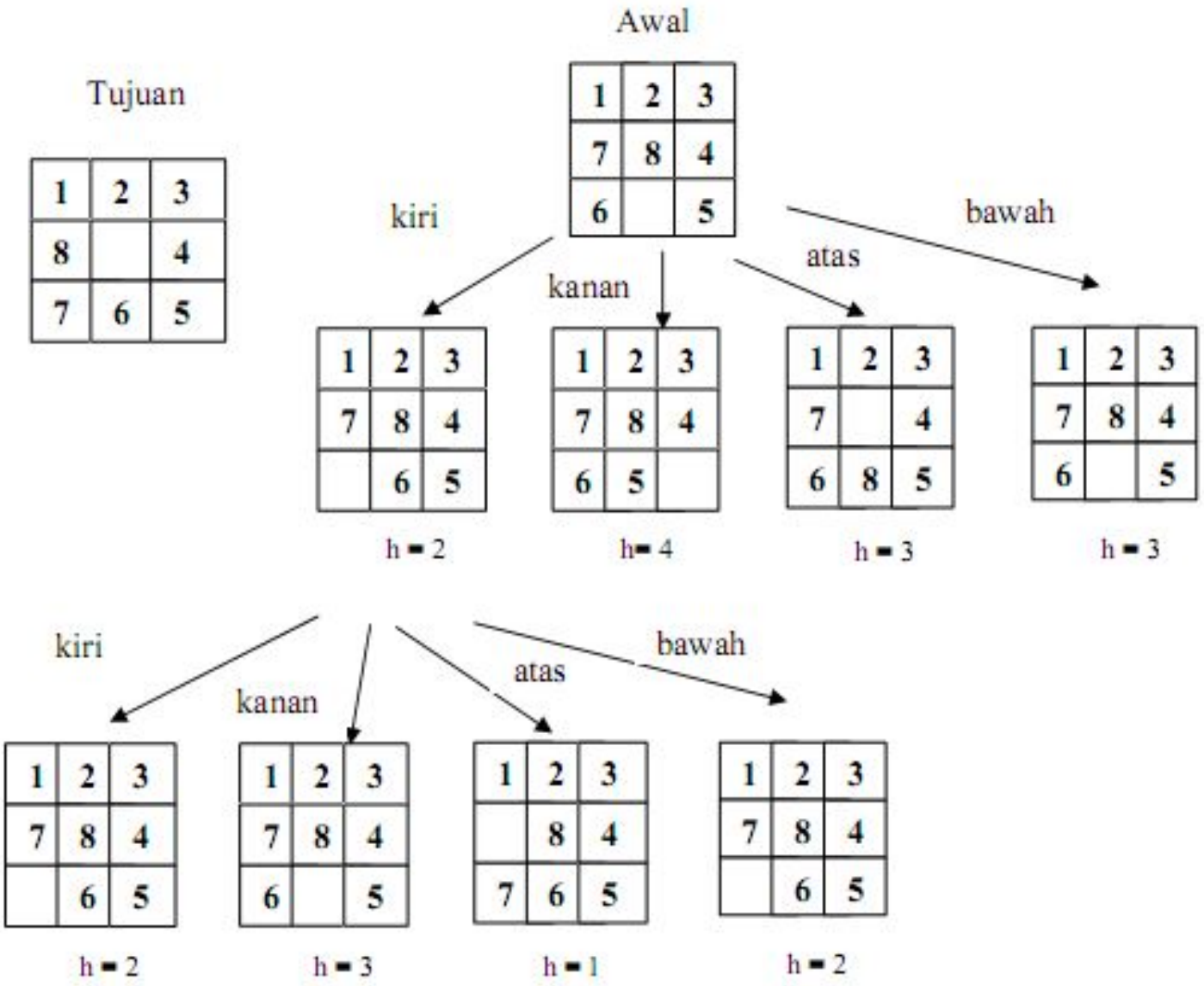

Awal
Tujuan
1 2 3
8 4
7 6 5
kiri
kanan
atas
bawah
1 2 3
7 8 4
6 5
h = 6
h = 4
h = 5
h = 5
kiri
kanan
atas
bawah
h = 6
h = 5
h = 7
h = 6

CONTOH : MEMILIH NILAI YANG KECIL

- Diberikan data informasi khusus :

1. Untuk jumlah ubin yang menempati posisi yang salah
2. Jumlah yang lebih kecil adalah yang diharapkan (lebih baik)

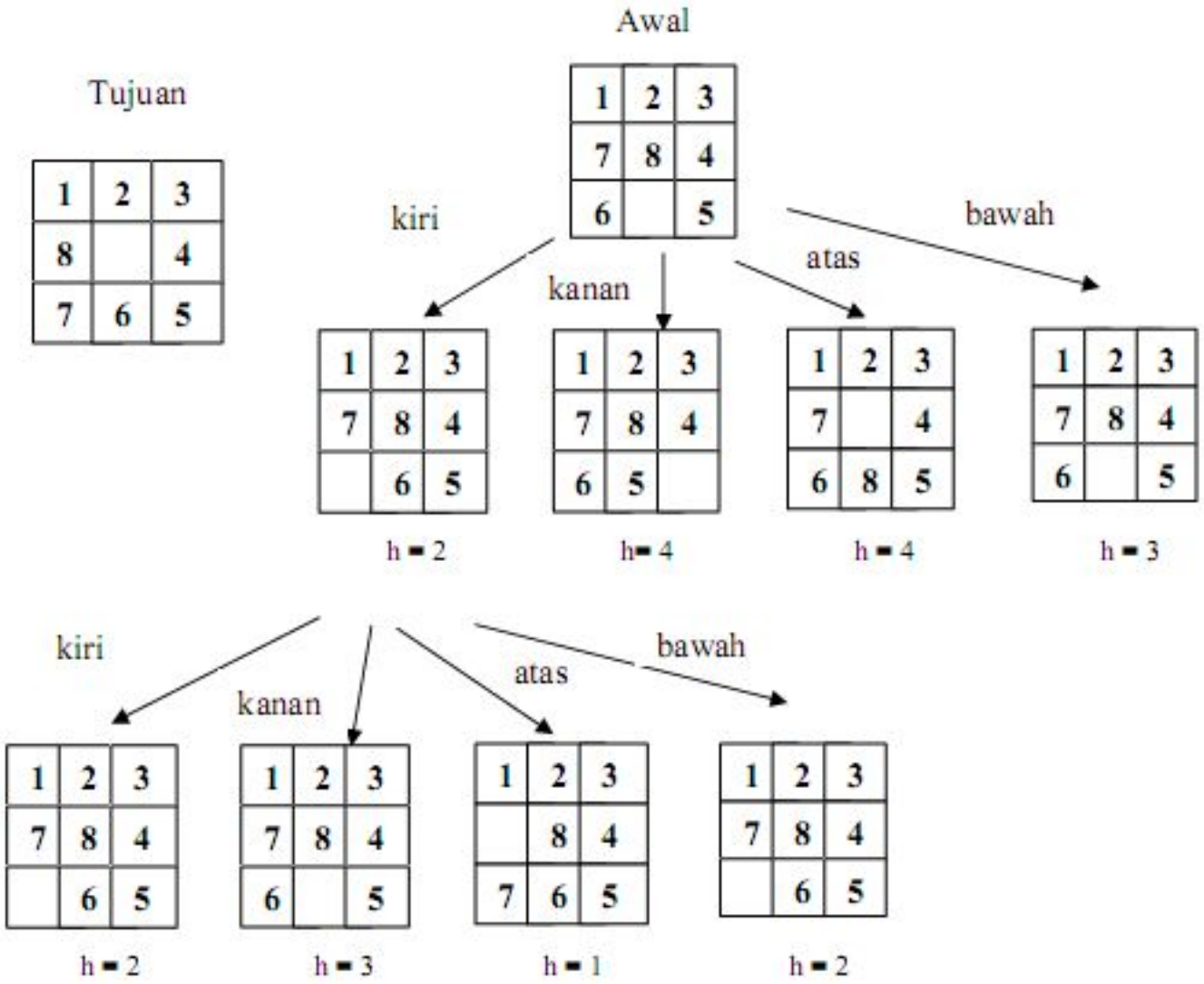
CONTOH


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

CONTOH : MEMILIH DENGAN TOTAL GERAKKAN

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

CONTOH


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

METODE - METODE HEURISTIK

- Generate And Test (Pembangkitan dan Pengujian)
- Hill Climbing

1. Simple Hill Climbing
2. Steepest-Ascent Hill Climbing

- Best-First Search

GENERATE AND TEST

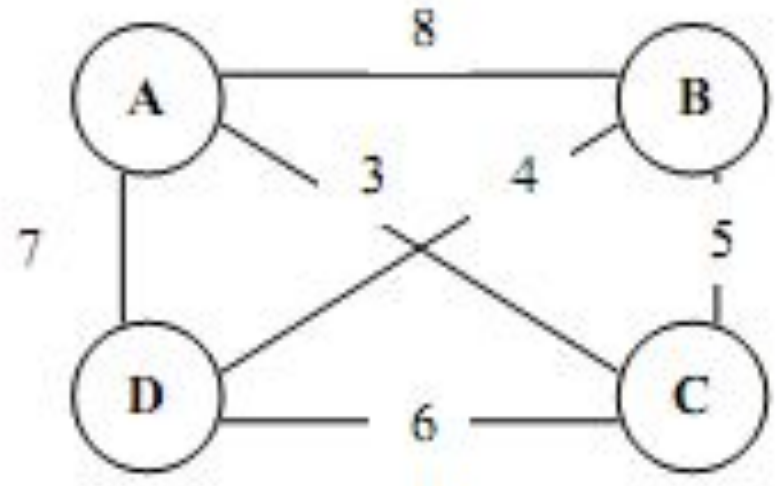
- Metode ini merupakan penggabungan antara *depth-first search* dengan pelacakan mundur (*backtracking*), yaitu bergerak ke belakang menuju pada suatu keadaan awal.

- Algoritma :

1. Bangkitkan suatu kemungkinan solusi (membangkitkan suatu titik tertentu atau lintasan tertentu dari keadaan awal).
2. Uji untuk melihat apakah node tersebut benar-benar merupakan solusinya dengan cara membandingkan node tersebut atau node akhir dari suatu lintasan yang dipilih dengan kumpulan tujuan yang diharapkan.
3. Jika solusi ditemukan, keluar. Jika tidak, ulangi kembali langkah pertama.

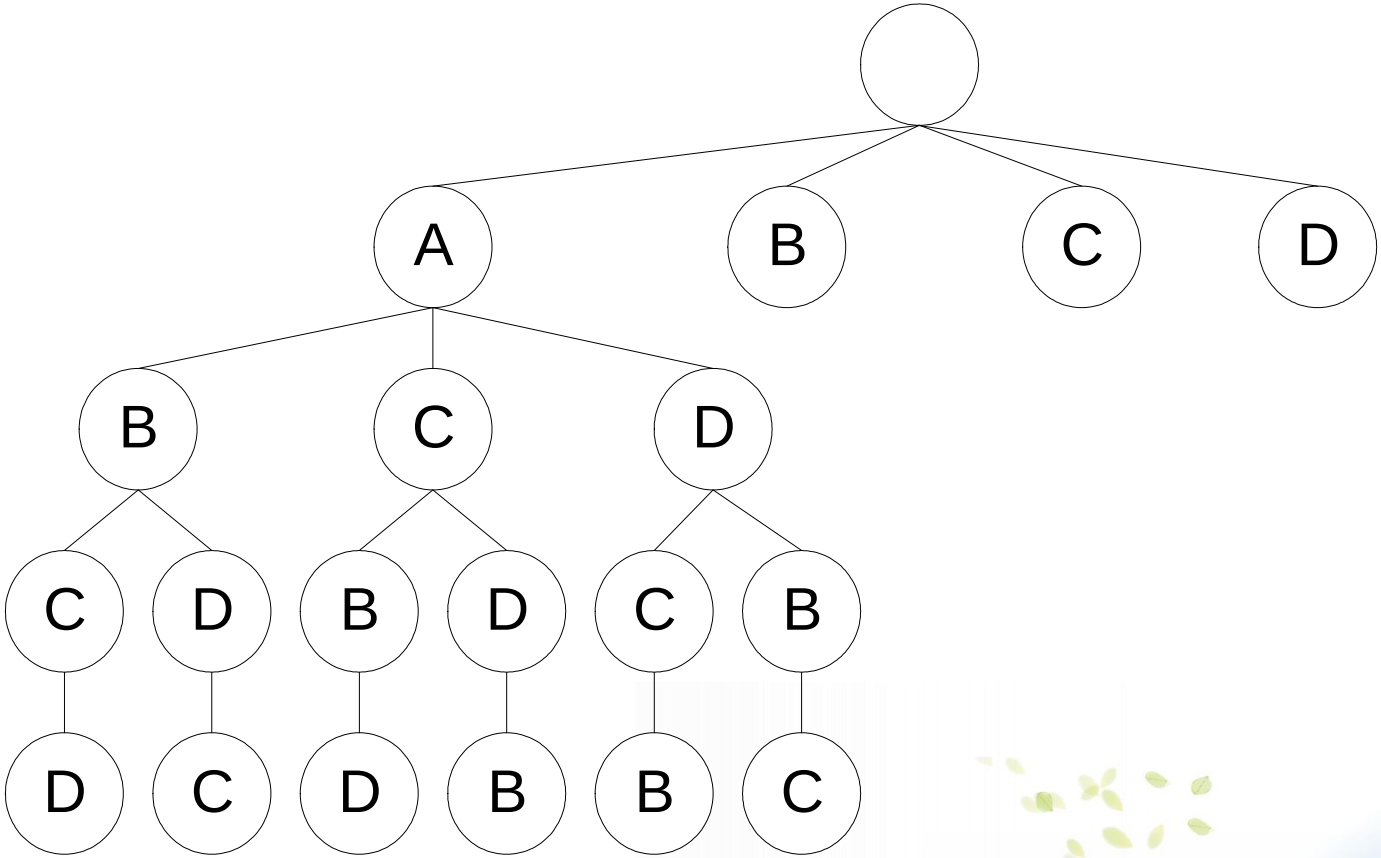
CONTOH : TRAVELING SALESMAN PROBLEM

- 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 :



PENYELESAIAN KASUS

- Penyelesaian dengan metode Generate And Test



PENYELESAIAN KASUS

PENCARIAN KE	LINTASAN	PANJANG LINTASAN	LINTASAN TERPILIH	PANJANG LINTASAN TERPILIH
1	ABCD	19	ABCD	19
2	ABDC	18	ABDC	18
3	ACBD	12	ACBD	12
4	ACDB	13	ACBD	12
5	ADBC	16	ACBD	12
6	ADCB	18	ACBD	12
7	BACD	17	ACBD	12
8	BADC	21	ACBD	12
9	BCAD	15	ACBD	12
10	BCDA	18	ACBD	12
11	BDAC	14	ACBD	12
12	BDCA	13	ACBD	12
13	CABD	15	ACBD	12
14	CADB	14	ACBD	12
15	CBAD	20	ACBD	12

PENYELESAIAN KASUS

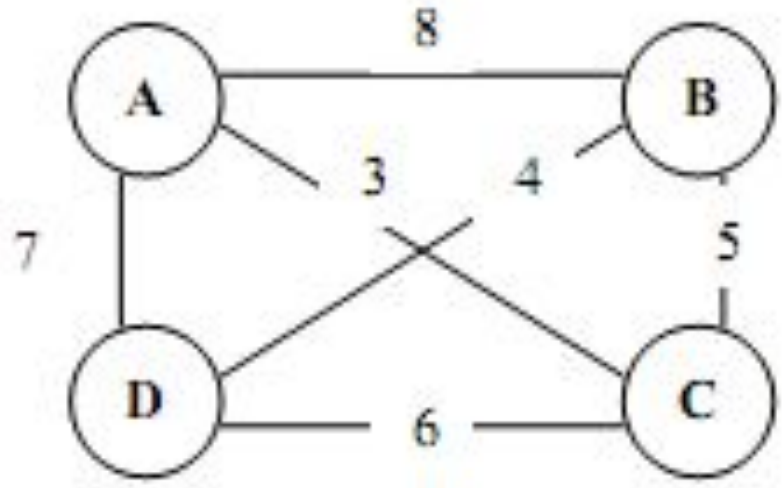
PENCARIAN KE	LINTASAN	PANJANG LINTASAN	LINTASAN TERPILIH	PANJANG LINTASAN TERPILIH
16	CBDA	16	ACBD	12
17	CDAB	21	ACBD	12
18	CDBA	18	ACBD	12
19	DABC	20	ACBD	12
20	DACB	15	ACBD	12
21	DBAC	15	ACBD	12
22	DBCA	12	ACBD / DBCA	12
23	DCAB	17	ACBD / DBCA	12
24	DCBA	19	ACBD / DBCA	12

HILL CLIMBING

- Metode ini hampir sama dengan metode generate dan test, hanya saja proses pengujian dilakukan dengan menggunakan fungsi heuristik.
- Pembangkitan keadaan berikutnya sangat tergantung pada feedback dari prosedur pengetesan.
- Tes yang berupa fungsi heuristik akan menunjukkan seberapa baiknya nilai terkaan yang diambil terhadap keadaan - keadaan lainnya yang mungkin.

CONTOH : TRAVELING SALESMAN PROBLEM

- 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 :



HILL CLIMBING

- 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)
- Dan seterusnya

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, dengan rumus sebagai berikut :

$$\frac{N!}{2!(N-2)!}$$

ALGORITMA SIMPLE HILL CLIMBING

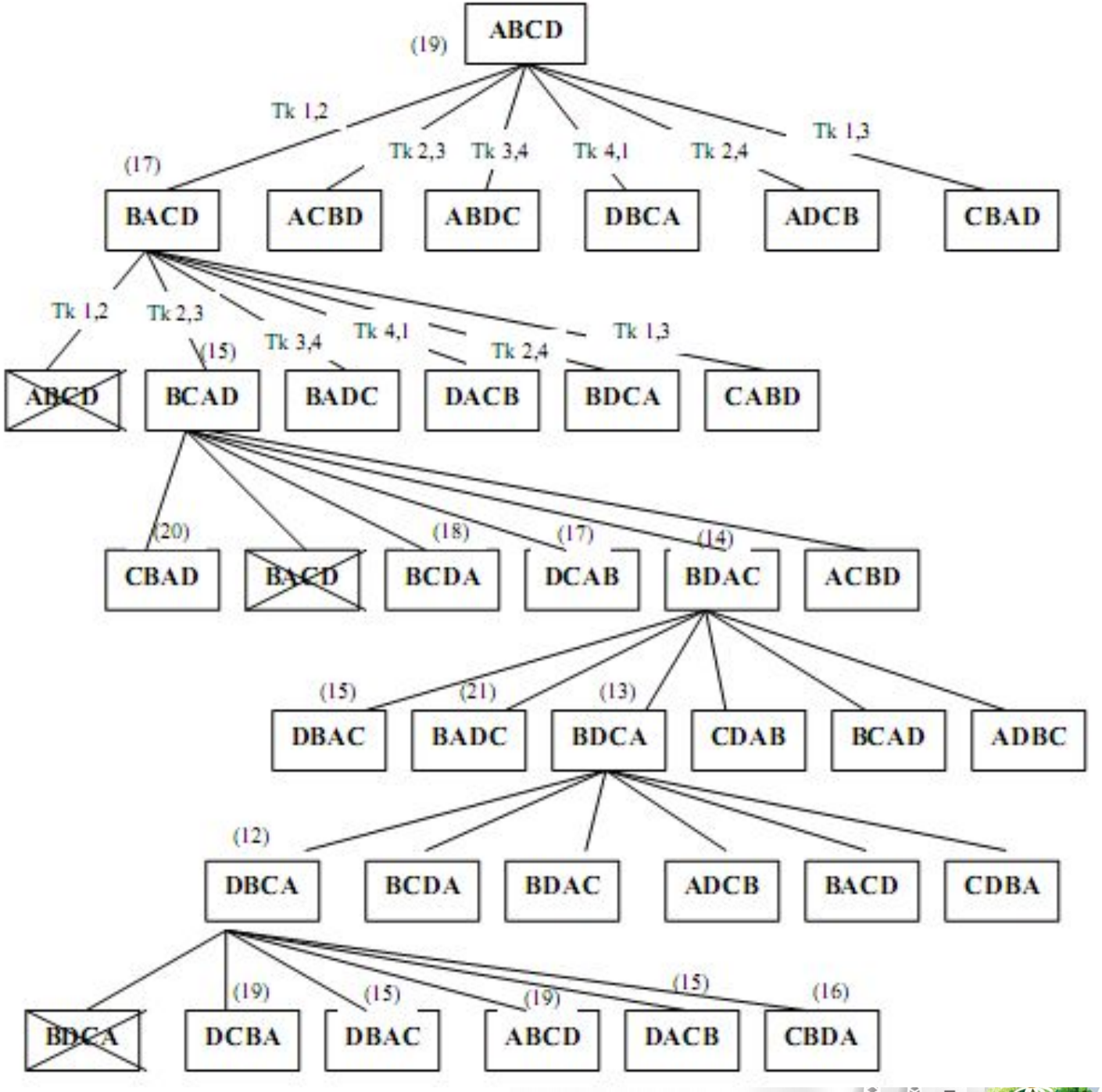
- Mulai dari keadaan awal, lakukan pengujian: jika merupakan tujuan, maka berhenti; dan jika tidak, lanjutkan dengan keadaan sekarang sebagai keadaan awal.
- Kerjakan langkah-langkah berikut sampai solusinya ditemukan, atau sampai tidak ada operator baru yang akan diaplikasikan pada keadaan sekarang:
    - Cari operator yang belum pernah digunakan; gunakan operator ini untuk mendapatkan keadaan yang baru.
    - Evaluasi keadaan baru tersebut.
    - Jika keadaan baru merupakan tujuan, keluar.
    - Jika bukan tujuan, namun nilainya lebih baik daripada keadaan sekarang, maka jadikan keadaan baru tersebut menjadi keadaan sekarang.
    - Jika keadaan baru tidak lebih baik daripada keadaan sekarang, maka lanjutkan iterasi.

SIMPLE HILL CLIMBING

- Operator : Tukar kota ke-i dengan kota ke-j (Tk i,j)
- 6 kombinasi tersebut digunakan sbg operator untuk 4 kota :

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

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


(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

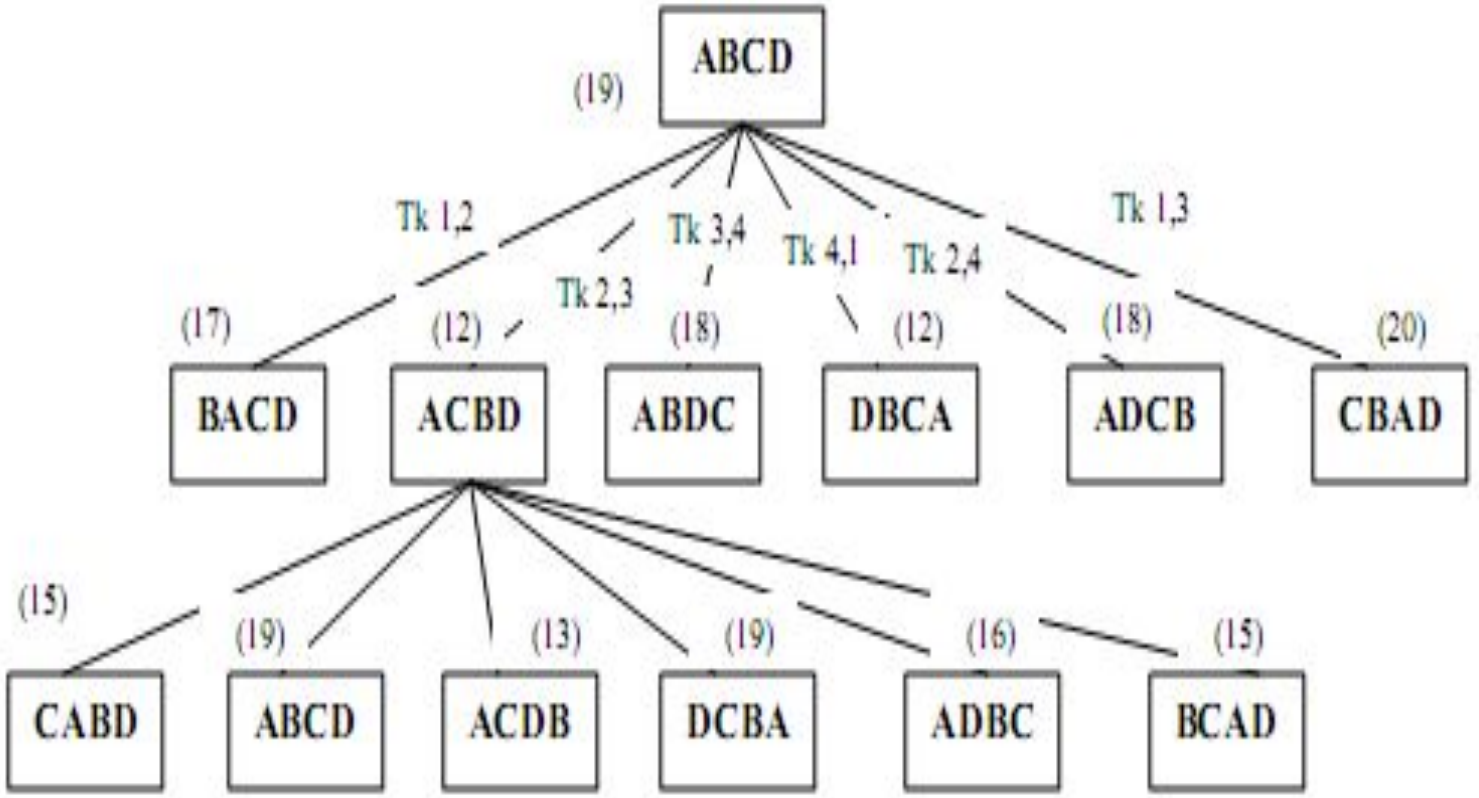
STEEPEST-ASCENT HILL CLIMBING

- Steepest-ascent hill climbing sebenarnya hampir sama dengan simple hill climbing, hanya saja gerakan pencarian tidak dimulai dari posisi paling kiri.
- Gerakan selanjutnya dicari berdasarkan nilai heuristik terbaik.
- Dalam hal ini urutan penggunaan operator tidak menentukan penemuan solusi.

ALGORITMA STEEPEST-ASCENT

- Mulai dari keadaan awal, lakukan pengujian: jika merupakan tujuan, maka berhenti; dan jika tidak, lanjutkan dengan keadaan sekarang sebagai keadaan awal.
- Kerjakan hingga tujuan tercapai atau hingga iterasi tidak memberikan perubahan pada keadaan sekarang.
- Tentukan SUCC sebagai nilai heuristic terbaik dari successor-successor.
- Kerjakan untuk tiap operator yang digunakan oleh keadaan sekarang:
- Gunakan operator tersebut dan bentuk keadaan baru.
- Evaluasi keadaan baru tersebut. Jika merupakan tujuan, keluar. Jika bukan, bandingkan nilai heuristiknya dengan SUCC. Jika lebih baik, jadikan nilai heuristic keadaan baru tersebut sebagai SUCC. Namun jika tidak lebih baik, nilai SUCC tidak berubah.
- Jika SUCC lebih baik daripada nilai heuristic keadaan sekarang, ubah node SUCC menjadi keadaan sekarang.

CONTOH


(19)
ABCD
Tk 1,2
Tk 2,3
Tk 3,4
Tk 4,1
Tk 2,4
Tk 1,3
(17)
BACD
(12)
ACBD
(18)
ABDC
(12)
DBCA
(18)
ADCB
(20)
CBAD
(15)
CABD
(19)
ABCD
(13)
ACDB
(19)
DCBA
(16)
ADBC
(15)
BCAD

BEST-FIRST SEARCH

- Metode best-first search ini merupakan kombinasi dari metode depth-first search dan metode breadth-first search dengan mengambil kelebihan dari kedua metode tersebut.
- Apabila pada pencarian dengan metode hill climbing tidak diperbolehkan untuk kembali ke node pada level yang lebih rendah meskipun node pada level yang lebih rendah tersebut memiliki nilai heuristik yang lebih baik, lain halnya dengan metode best-first search ini.
- Pada metode best-first search, pencarian diperbolehkan mengunjungi node yang ada di level yang lebih rendah, jika ternyata node pada lebih yang lebih tinggi ternyata memiliki nilai heuristik yang lebih buruk.

BEST-FIRST SEARCH

- Penentuan node berikutnya adalah node yang terbaik yang pernah dibangkitkan
- Menggunakan informasi :
  - Biaya perkiraan
  - Biaya sebenarnya
- Terdapat 2 jenis algoritma pada best-first search :
  - Greedy Best First Search
    - biaya perkiraan $f(n) = h(n)$
  - A*
    - biaya perkiraan + biaya sebenarnya
    - f(n) = g(n) + h(n)
- Penjelasan :
  - f = fungsi evaluasi
  - g = cost dari initial state ke current state
  - h = prakiraan *cost* dari *current state* ke *goal state*

BEST-FIRST SEARCH

- Untuk mengimplementasikan metode ini menggunakan graph keadaan, dibutuhkan 2 antrian yang berisi node-node, yaitu:
  - OPEN, berisi node,node yang sudah dibangkitkan, namun belum diuji. Umumnya berupa antrian berprioritas yang berisi elemen-elemen dengan nilai heuristik tertinggi
  - CLOSED berisi node-node yang sudah diuji

ALGORITMA BEST-FIRST SEARCH

- Mulai dengan OPEN hanya berisi initial state
- Sampai goal ditemukan atau tidak ada lagi simpul yang tersisa dalam OPEN, lakukan :

a. Pilih simpul terbaik dalam OPEN

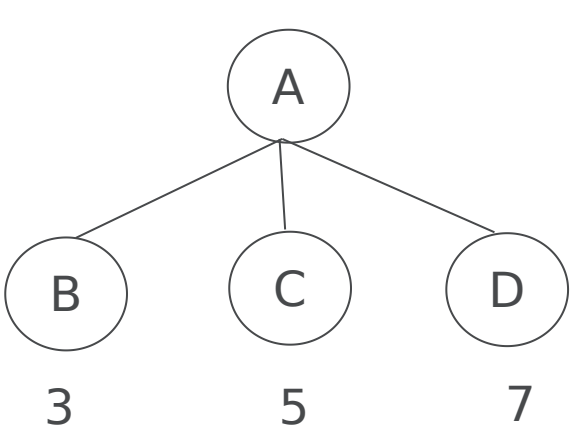
b. Telusuri successor-nya

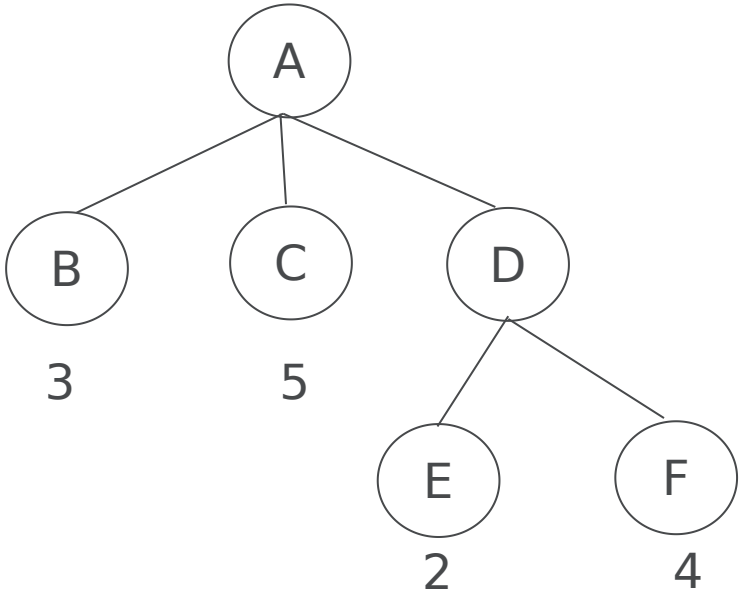
c. Untuk tiap successor, lakukan :

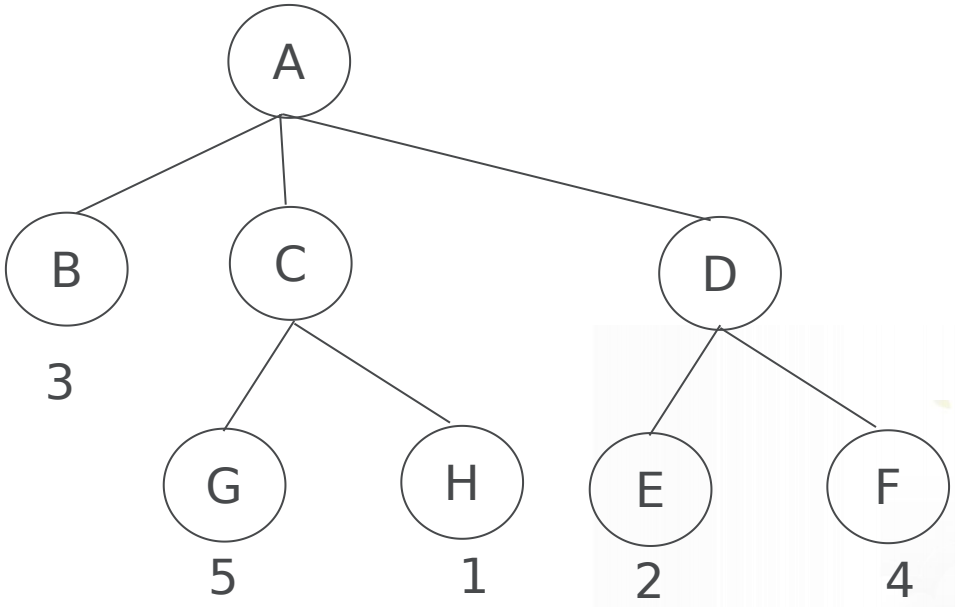
i. Jika belum pernah ditelusuri sebelumnya, evaluasi simpul ini tambahkan dalam OPEN dan catat parentnya.

ii. Jika sudah pernah ditelusuri, ganti parentnya jika jalur baru lebih baik dari sebelumnya

CONTOH







TERIMA KASIH