PERTEMUAN 5

SEARCH BAGIAN 2

Generate-and-Test

Teknik Generate-and-Test adalah teknik yang paling mudah dibandingkan teknik search yang lain, namun relatif lebih lama dalam mendapatkan solusi.

Algoritma Generate-and-Test :

1. Bentuk solusi yang mungkin.

   Untuk beberapa masalah, ini berarti membentuk poin terpisah dari area permasalahan. Pada masalah lain, ini berarti membentuk jalur dari stata awal.

2. Lakukan test untuk melihat apakah poin yang ditemui adalah solusi dengan membandingkan poin yang dipilih atau poin terakhir dari jalur yang dipilih dengan kumpulan stata tujuan

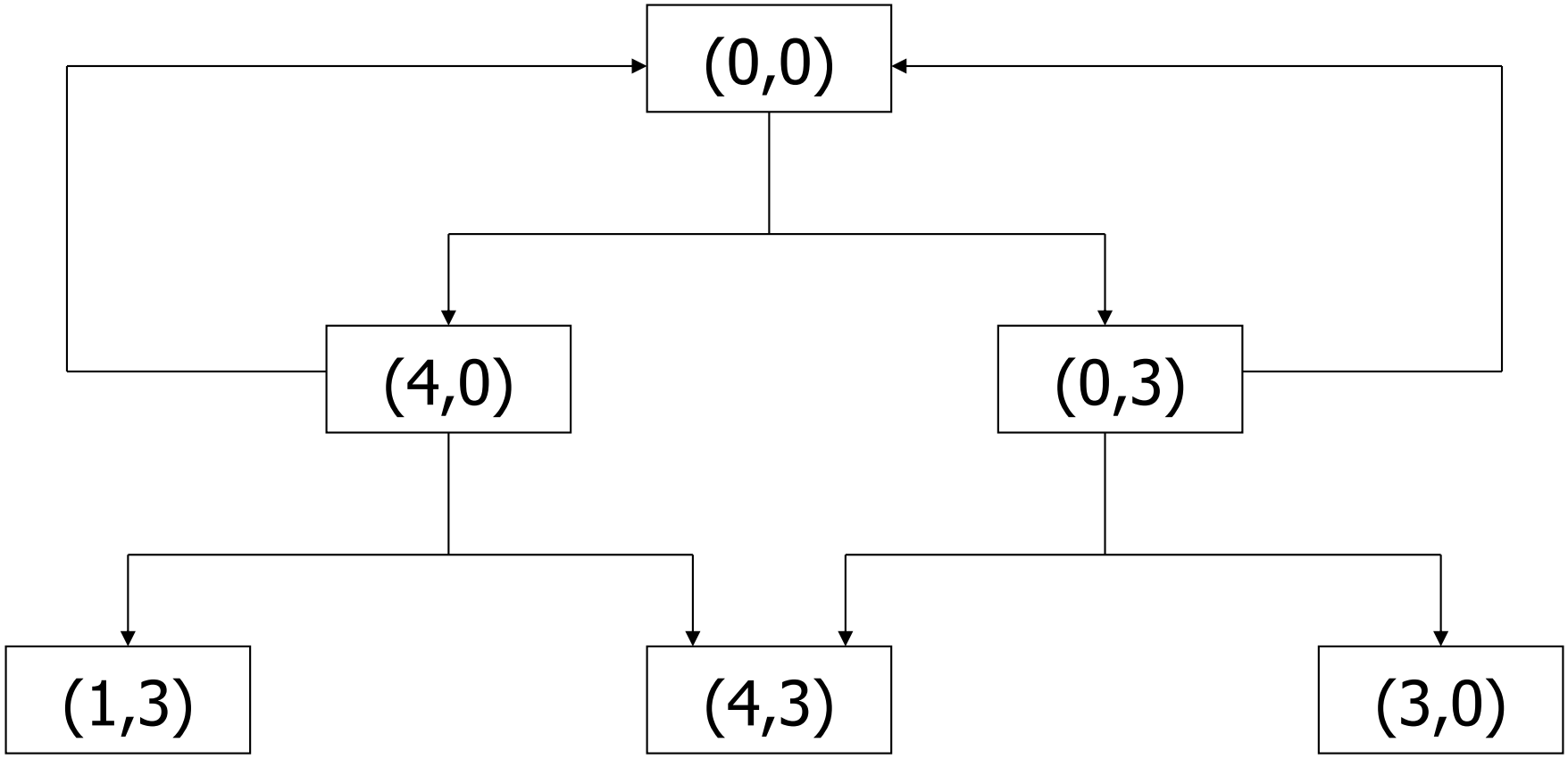
3. Jika solusi sudah ditemukan, quit. Jika belum kembali ke langkah 1.

Kebaikan dan Keburukan Generate-and-Test

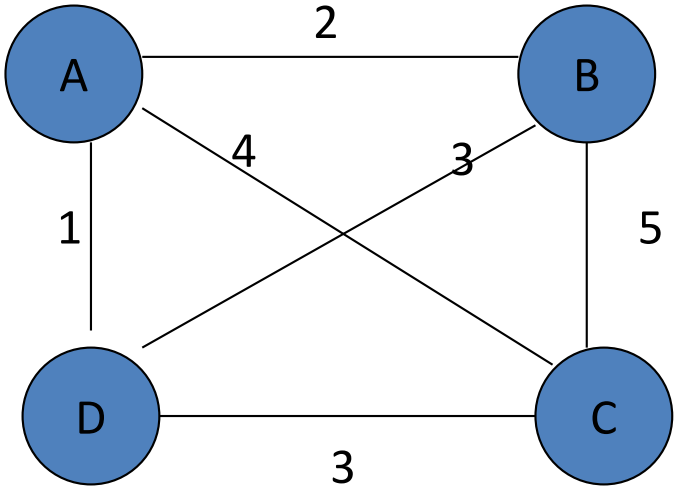
Jika penurunan solusi yang mungkin dilakukan secara sistematis, maka procedure diatas akan dapat menemukan solusi suatu saat, jika memang ada. Tapi sayangnya jika ruang permasalahan sangat luas maka saat ditemukannya solusi akan menjadi sangat lama.

Cara terbaik menerapkan generate-and-test yang sistematis adalah pada tree dari depth-first search dengan backtracking, yaitu kembali ke stata sebelumnya bila ditemui stata yg sudah pernah di test atau memodifikasi prosedurnya untuk menelusuri stata pada bentuk graph.

Contoh Kasus 1
Search Graph untuk masalah bejana air



Contoh kasus TSP



Sebuah rute yng harus dilewati seorang sales dimana sales tersebut harus merlewati setiap kota tepat sekali. Terdapat 4 kota, dengan jarak masing-masing kota AB=2, AC=4, AD=1, BC=5, BD=3, CD=3. Tujuannya adalah mencari jarak terpendek bagi sales untuk mengunjungi semua kota sekali.

Penyelesaian menggunakan generate-test adalah dengan membangkitkan solusi-solusi yang mungkin ada sesuai permasalahan yang dihadapi oleh sales tersebut. Kombinasi abjad sebagai solusi yang mungkin adalah n! = 4! = 24. Tujuannya adalah mencari solusi dengan panjang terpendek.

No pencarian	Lintasan	Panjang Lintasan	Lintasan yang dipilih	Panjang lintasan
1	ABCD	10	ABCD	10
2	ABDC	8	ABDC	8
3	ACBD	12	ABDC	8
4	ACDB	10	ABDC	8
5	ADCB	9	ABDC	8
6	ADCB	9	ABDC	8
7	BACD	9	ABDC	8
8	BADC	6	BADC	6
9	BCAD	10	BADC	6
10	BCDA	9	BADC	6
11	BDAC	8	BADC	6
12	BDCA	10	BADC	6
13	CABD	9	BADC	6
14	CADB	8	BADC	6
15	CBAD	8	BADC	6
16	CBDA	9	BADC	6
17	CDAB	6	BADC/CDAB	6
18	CDBA	8	BADC/CDAB	6
19	DABC	8	BADC/CDAB	6
20	DACB	10	BADC/CDAB	6
21	DBAC	9	BADC/CDAB	6
22	DBCA	12	BADC/CDAB	6
23	DCAB	9	BADC/CDAB	6
24	DCBA	10	BADC/CDAB	6

Dari tabel diatas, solusi pertama yang dibangkitkan adalah ABCD = 10, solusi kedua ABDC=8. Ternyata solusi kedua menghasilkan jarak yang lebih pendek sehingga dipilih lintasan ABDC=8. Lakukan untuk langkah selanjutnya. Pada tabel didapat solusi terpendek adalah BADC atau CDBA.

Kelemahan dari teknik ini pelru dibangkitkan semua kemungkinan yang ada sehingga apabila ditambahkan satu kota untuk permasalahan TSP ini diatas 5 kota. Maka akan diperlukan 120 kombinasi lintasan, kecuali diberikan kondisi tertentu misalnya kota awal bagi sales telah ditentukan.

Hill Climbing

Teknik Hill Climbing adalah pengembangan dari teknik Generate-and-Test, dengan penambahan adanya umpan balik dari prosedur test yang sudah digunakan untuk membantu memilih arah mana yang harus ditelusuri pada setiap area search.

Pada prosedur Generate-and-Test yang murni, fungsi test hanya ditanggapi dengan Ya atau Tidak. Tetapi pada Hill-Climbing fungsi test ditambahkan dengan fungsi heuristic atau fungsi objectif yang memungkinkan perkiraan seberapa dekat simpul yang ditelusuri terhadap goal state.

Hill-climbing sering kali digunakan jika fungsi heuristic yang baik tersedia untuk mengevaluasi stata, tapi ketika tidak ada lagi pengetahuan yang dapat digunakan.

Sebagai contoh, anda berada disuatu kota yang belum pernah anda kunjungi tanpa memiliki peta. Tujuannya menuju gedung tertinggi yang terlihat dari tempat anda berada.

Fungsi heuristic adalah hanya masalah jarak antara lokasi anda berada dengan letak gedung tertinggi dan bagaimana menemukan jarak yang terdekat atau cara tercepat menuju gedung tertinggi.

Penyelesaian masalah diatas dimulai dengan meninjau karakteristik masalah, apakah solusi yang pertama ditemukan dapat diterima sebagai solusi yang baik ? (mutlak atau relatif ?) Karena tidak ada peta dan tidak ada pengalaman memilih jalan (tidak ada pengetahuan) maka dipilih saja jalan yang arahnya menuju solusi sampai kita tiba di tujuan tanpa mengulangi atau mencoba lagi jalur yang lain dan kita terima itu sebagai solusi terbaik (dgn mengabaikan kemungkinan lain).

Jadi adalah masuk akal menerapkan hill-climbing ketika tidak ada alternatif yang dapat diterima untuk memilih atau menuju pada suatu stata.

Algoritma Simple Hill Climbing :

1. Evaluasi initial state. Jika ini goal state maka return dan keluar. Jika bukan maka lanjutkan dengan initial state sebagai current state.
2. Ulangi langkah berikut sampai menemukan solusi atau sampai tidak ada lagi operator yang dapat digunakan pada current state
   a. Pilih operator yang belum digunakan pada current state dan gunakan untuk menghasilkan/menuju stata baru
   b. Evaluasi stata baru.
      i. Jika ini goal state maka return dan keluar
      ii. Jika bukan goal state tetapi lebih baik dari current state maka jadikan stata baru sebagai current state
      iii. Jika tidak lebih baik dari current state lanjutkan perulangan

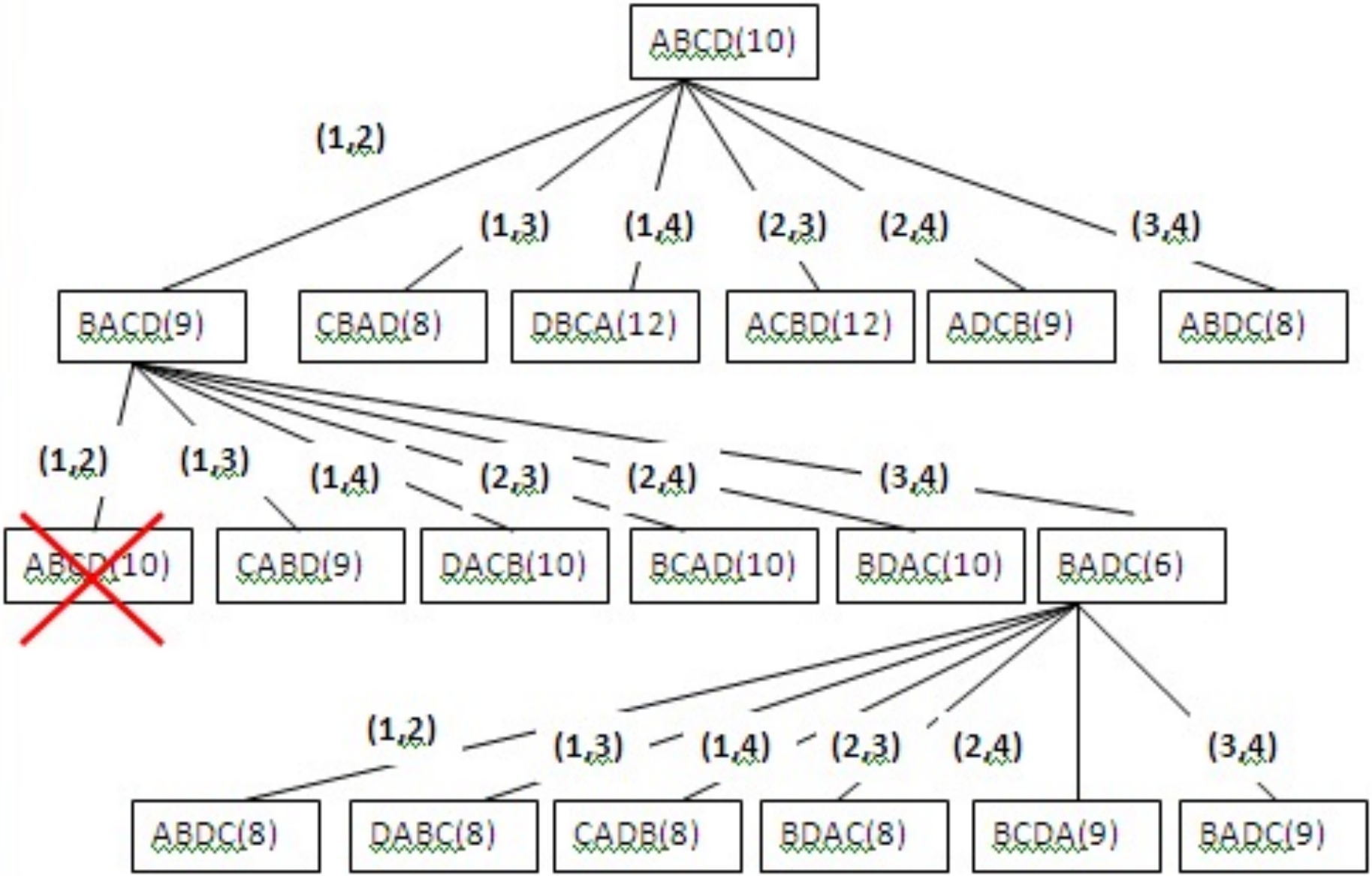
Sebagai ilustrasi teknik pencarian *simple hill climbing* digunakan contoh masalah TSP pada masalah *generate and test* . Operator yang digunakan adalah operator yang dapat menghasilkan kombinasi lintasan kota yang berbeda-beda, yaitu dengan cara menukar posisi masing-masing kota. Untuk mempermudah penukaran posisi, kita cukup menukar posisi 2 kota, operator untuk kombinasi lintasan dengan menukar posisi 2 kota dapat dihitung dengan kalkulasi

$$\frac{4!}{2!\,(4-2)!}$$

Yaitu :

1. (1,2) menukar posisi kota kesatu dan kedua
2. (1,3) menukar posisi kota kesatu dan ketiga
3. (1,4) menukar posisi kota kesatu dengan keempat
4. (2,3) menukar posisi kota kedua dengan kota ketiga
5. (2,4) menukar posisi kota kedua dengan keempat
6. (3,4) menukar posisi kota ketiga dengan keempat

Penggunaan pengurutan operator harus konsisten, tidak boleh berbeda tiap levelnya. urutan penggunaan operator juga sangat menentukan kecepatan dalam menemukan solusi.



Pencarian *simple hill climbing* dimulai dari anak kiri. Apabila nilai heuristik anak kiri lebih baik maka dibuka untuk pencarian selanjutnya. Jika tidak maka akan dilihat tetangga dari anak kiri tersebut, dan seterusnya.

Level 1 : (ABCD=10 > BACD =9) buka node BACD tanpa harus mencek node yang selevel dengan BACD.

Level 2 : node ABCD dilewati.

(BACD=9 = CABD=9) periksa node tetangga CABD

(BACD=9 < DABC=10) periksa node tetangga DABC

(BACD=9 < BCAD=10) periksa node tetangga BCAD

(BACD=9 < BDAC=10) periksa node tetangga BDAC

(BACD=9 > BADC=6) buka node BADC

Level 3 : (BADC=6 < ABDC=8) periksa tetangga ABDC

(BADC=6 < DABC=8) periksa tetangga DABC

(BADC=6 < CADB=8) periksa tetangga CADB

(BADC=6 < BDAC=8) periksa tetangga BDAC

(BADC=6 <BCDA=9) periksa tetangga BCDA

(BADC=6 < BADC=9) selesai.

Best-First-Search

Teknik Best-First-Search adalah teknik search yang menggabungkan kebaikan yang ada dari teknik Depth-First-Search dan Breadth-First-Search.

Tujuan menggabungkan dua teknik search ini adalah untuk menelusuri satu jalur saja pada satu saat, tapi dapat berpindah ketika jalur lain terlihat lebih menjanjikan dari jalur yang sedang ditelusuri. Untuk mendapatkan jalur yang menjanjikan adalah dengan memberikan skala prioritas pada setiap stata saat dihasilkan dengan fungsi heuristic.

Untuk menggunakan Best-First-Search, kita memerlukan dua daftar simpul, yaitu :

1. OPEN

   berisi simpul yang dihasilkan dari fungsi heuristic tapi belum dievaluasi, memilki antrian prioritas dimana elemen dengan prioritas tertinggi adalah yang memiliki nilai paling baik yang dihasilkan fungsi heuristic.

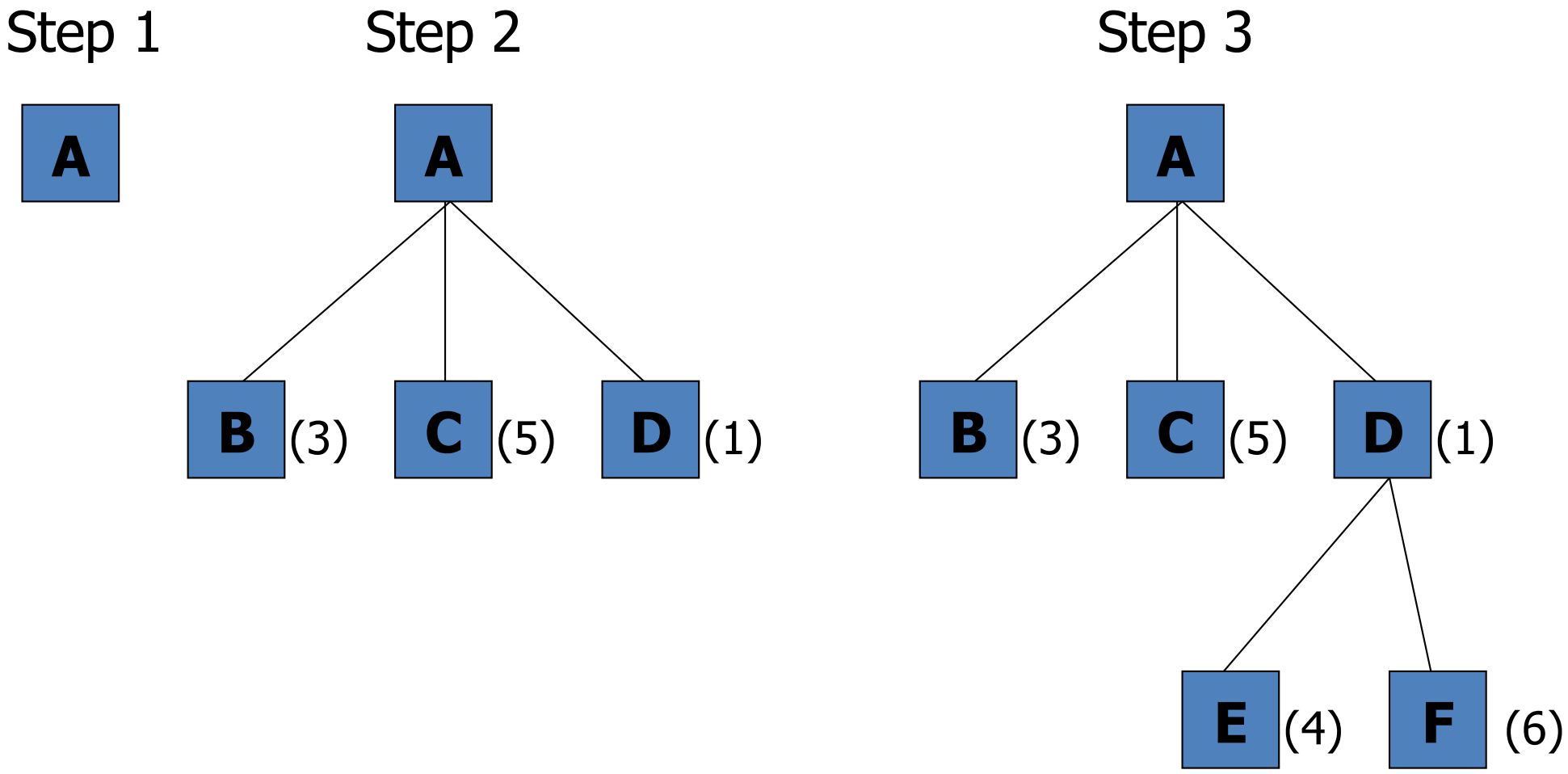
2. CLOSED

   berisi simpul yang sudah dievaluasi. Kita perlu tetap menyimpan simpul-simpul ini dalam memori jika kita ingin melakukan search pada Graph, sehingga jika kita menemui suatu simpul kita bisa memeriksa apakah simpul ini sudah pernah dieavaluasi atau belum

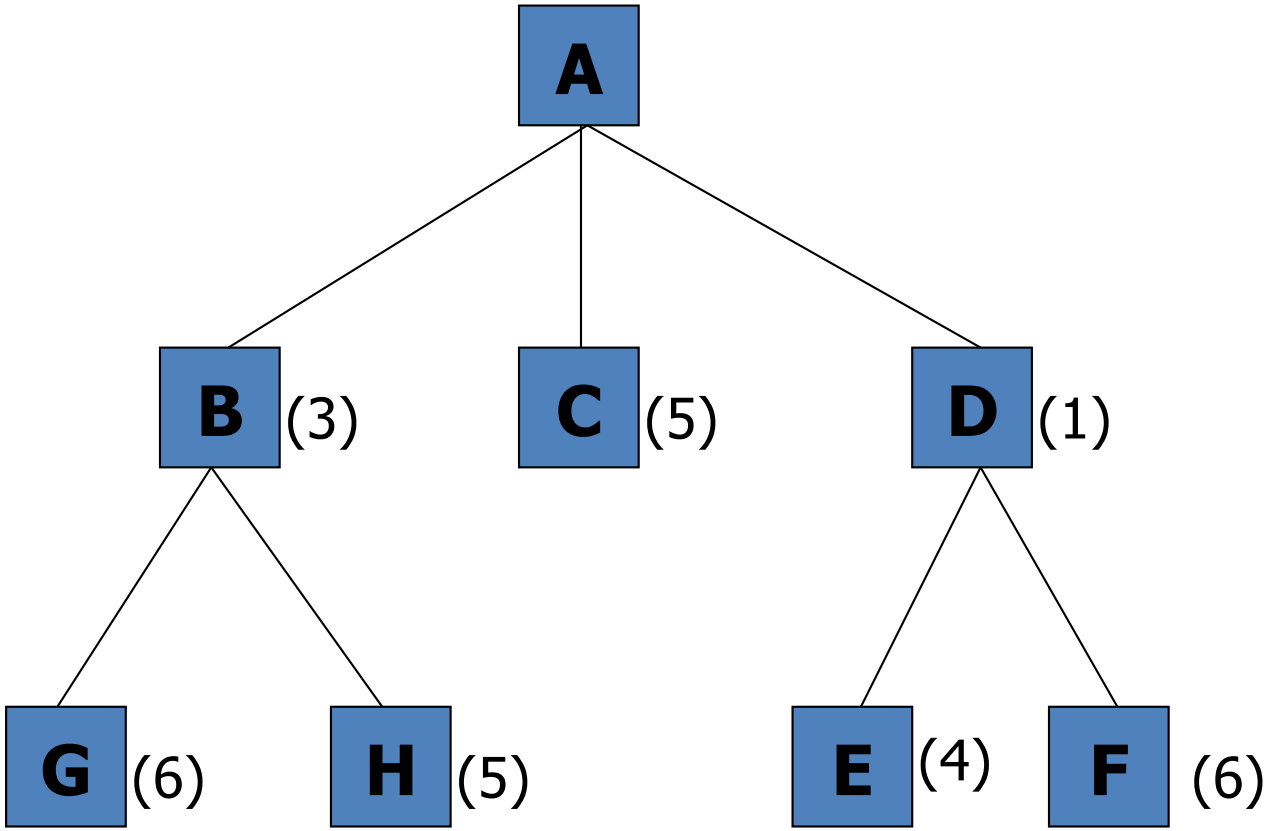
Algoritma Best-First-Search :

1. Mulai dengan OPEN hanya berisi initial state
2. 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 kedalam OPEN dan catat parentnya.
      ii. Jika sudah pernah ditelusuri, ganti parent nya jika jalur baru lebih baik dari sebelumnya.

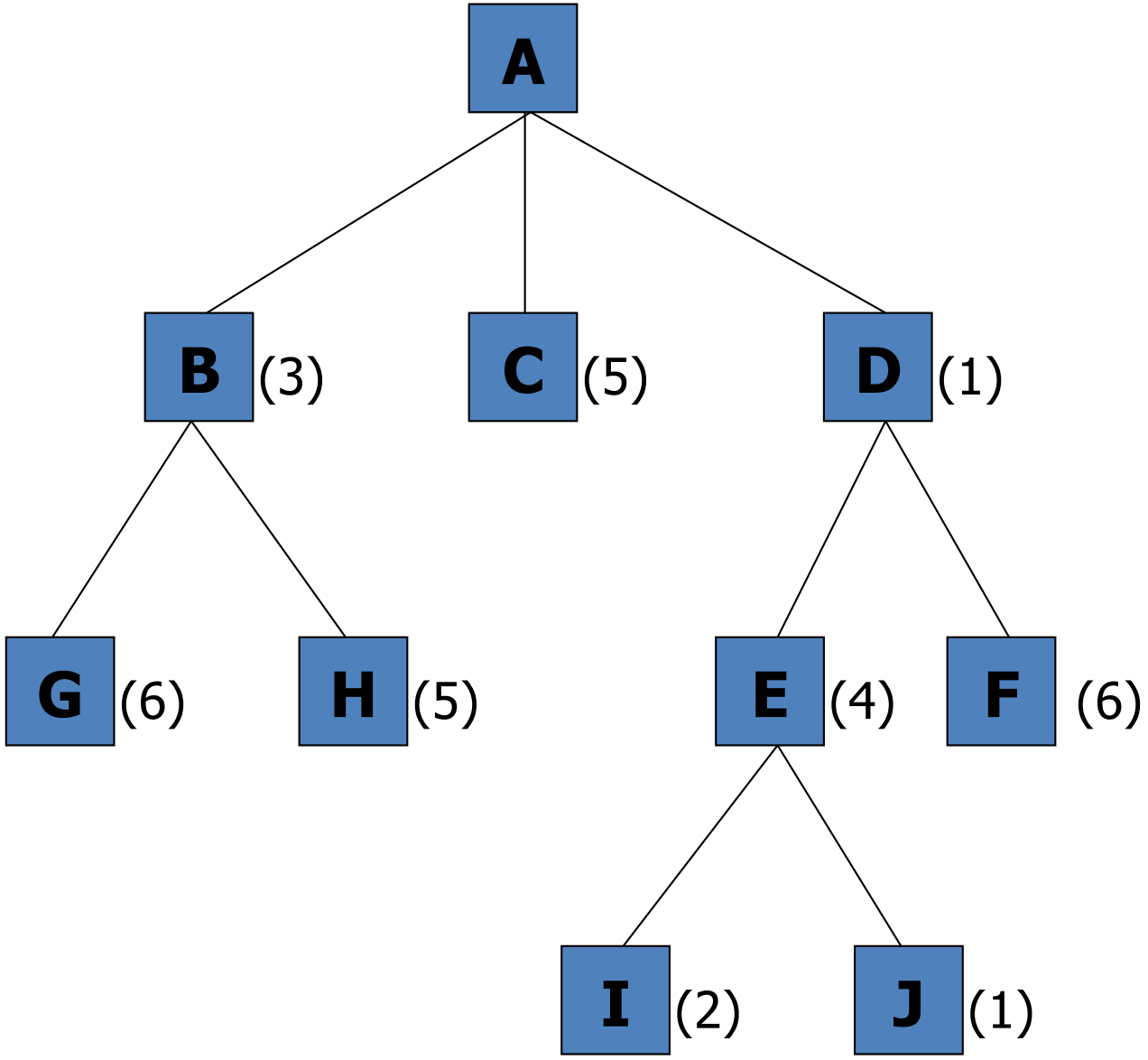
Contoh Best First Search



Step 4



Step 5



Contoh lain *Best-First-Search*

Diketahui sebuah puzzle berukuran 3X3 yang berisi angka. Permasalahan adalah angka-angka dalam puzzle tersebut belum teratur.

Nilai awal puzzle :

1	2	
4	5	3
7	8	6

Goal :

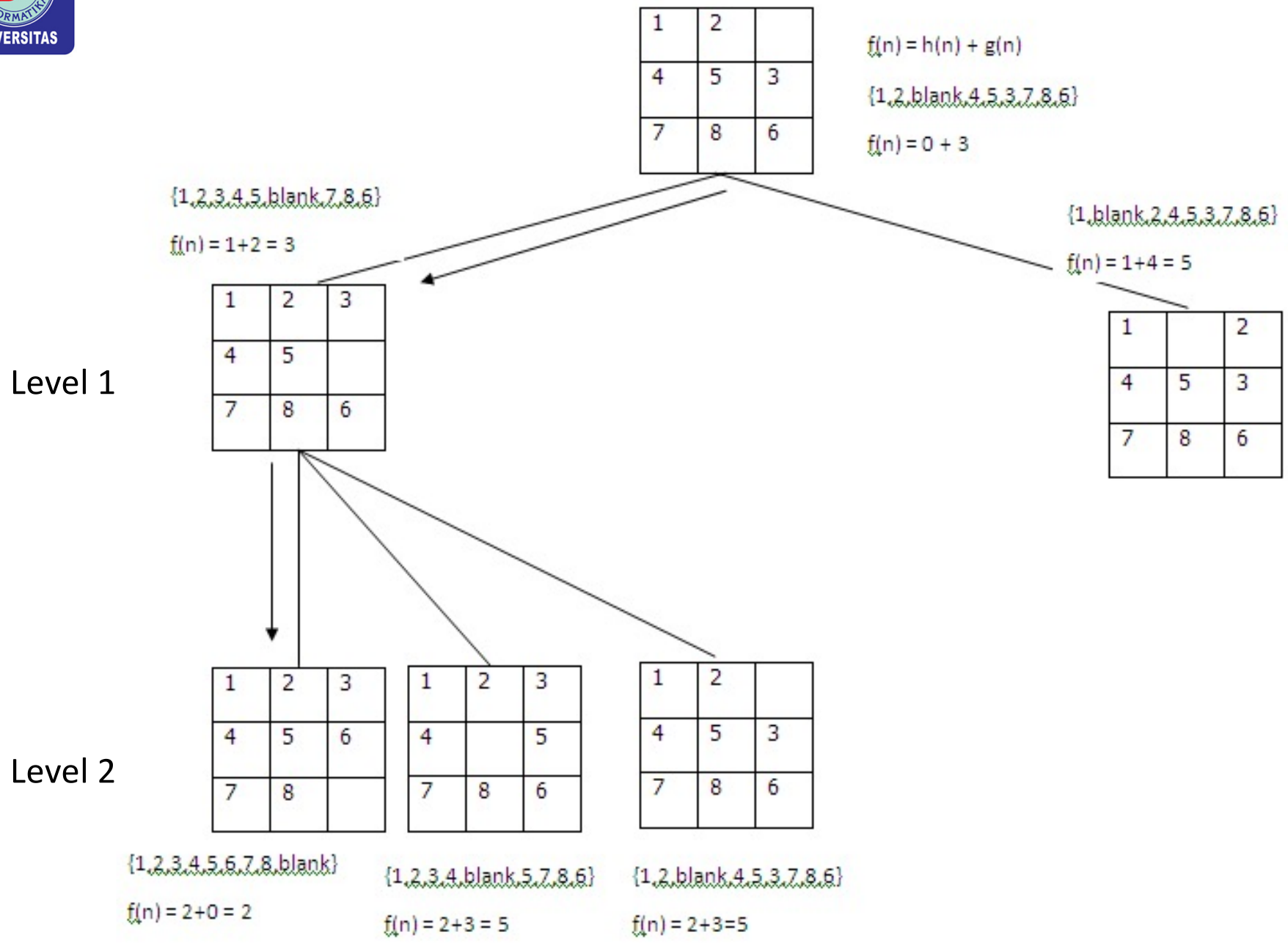
1	2	3
4	5	6
7	8	

Nilai awal = {1,2,blank,4,5,3,7,8,6} Goal = {1,2,3,4,5,6,7,8,blank}

Nilai heuristic = f(n) = g(n) + h(n)

g(n) = kedalaman dari pohon

h(n) = jumlah angka yang masih salah posisi



Latihan soal di rumah

Sebuah puzzle berukuran 3X3

Nilai awal :

2	8	3
1	6	4
7		5

Goal :

1	2	3
8		4
7	6	5

f(n) =g(n) + h(n)

g(n) = kedalaman pohon

h(n) = jumlah angka yang salah posisi

THE END