

Pencarian

Kecerdasan Buatan
Pertemuan 3
Yudianto Sujana

Metode Pencarian dan Pelacakan

- 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.

Metode Pencarian dan Pelacakan

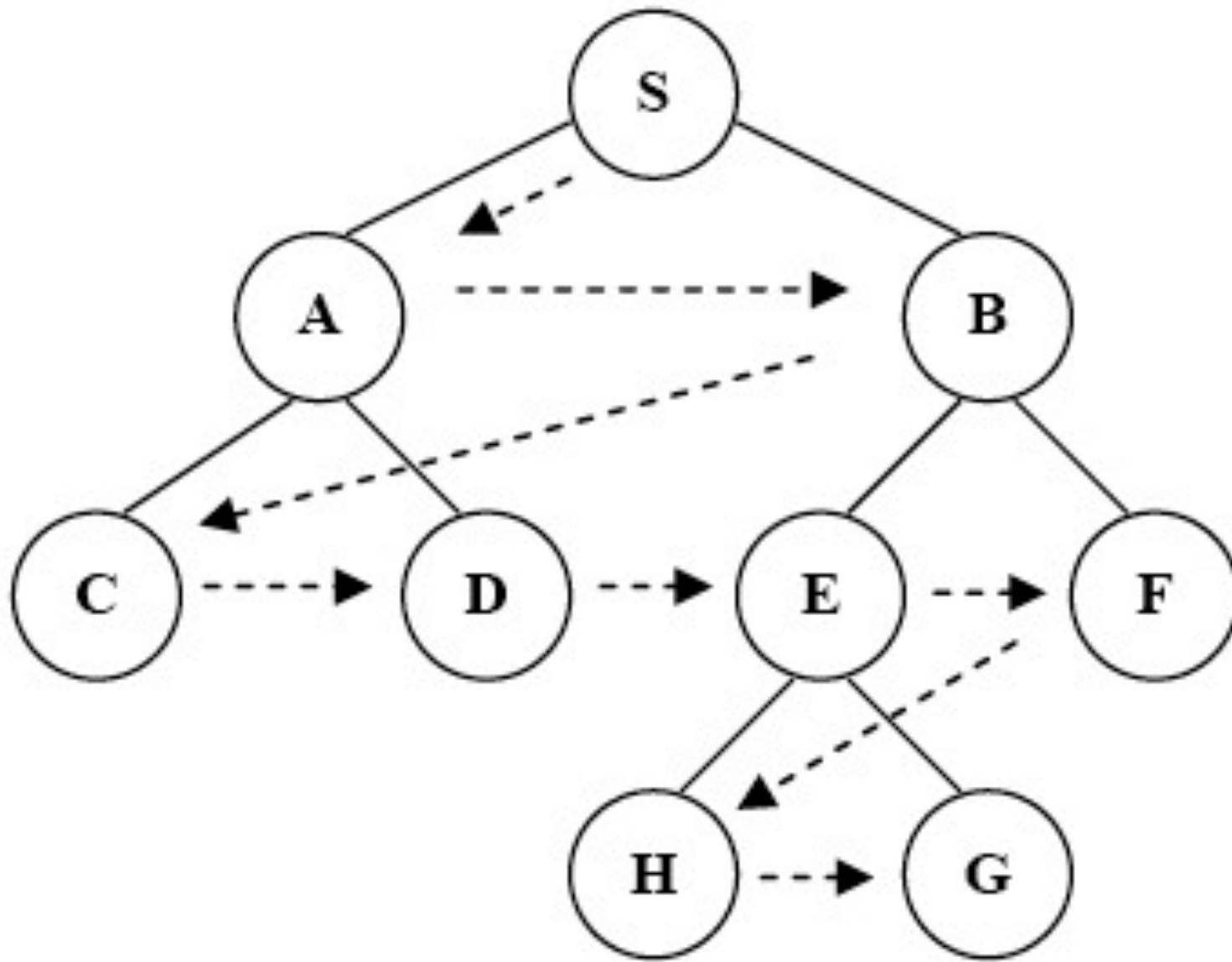
- Untuk mengukur perfomansi metode pencarian, terdapat empat kriteria yang dapat digunakan :
 - Completeness : apakah metode tersebut menjamin penemuan solusi jika solusinya memang ada?
 - Time complexity : berapa lama waktu yang diperlukan?
 - Space complexity : berapa banyak memori yang diperlukan
 - Optimality : apakah metode tersebut menjamin menemukan solusi yang terbaik jika terdapat beberapa solusi berbeda?

Metode Pencarian dan Pelacakan

- Dua teknik pencarian dan pelacakan
 - Pencarian buta (blind search)
 - Pencarian melebar pertama (Breadth – First Search)
 - Pencarian mendalam pertama (Depth – First Search)
 - Pencarian terbimbing (heuristic search)
 - Pendakian Bukit (Hill Climbing)
 - Pencarian Terbaik Pertama (Best First Search)

Pencarian Melebar Pertama (Breadth-First Search)

- Semua node pada level n akan dikunjungi terlebih dahulu sebelum level $n+1$
- Mulai dari akar terus ke level 1 dari kiri ke kanan
- Kemudian ke level selanjutnya hingga solusi ditemukan

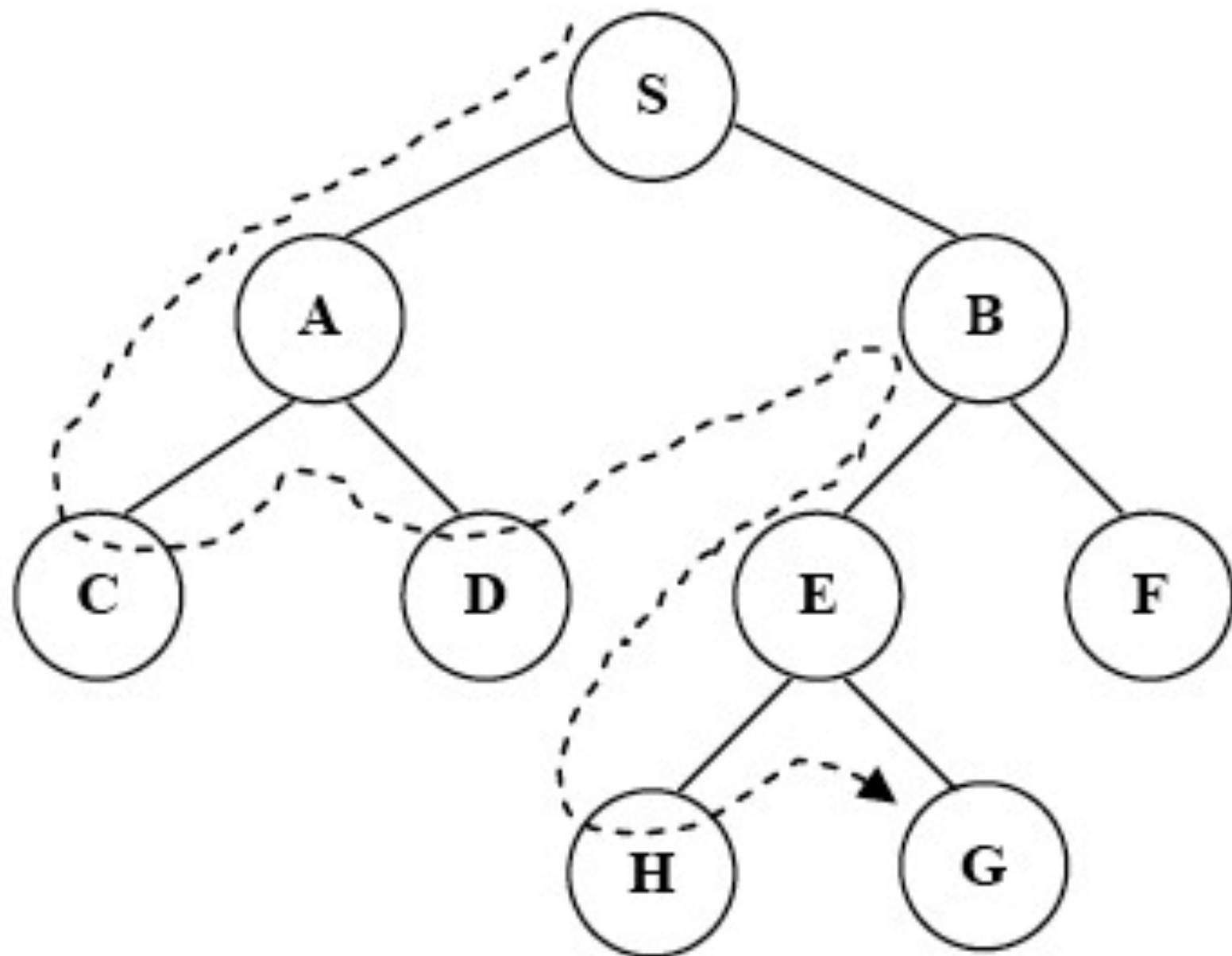


Pencarian Melebar Pertama (Breadth-First Search)

- Keuntungan
 - Tidak akan menemui jalan buntu
 - Menjamin ditemukannya solusi (jika solusinya memang ada) dan solusi yang ditemukan pasti yang paling baik
 - Jika ada satu solusi maka bread-first search akan menemukannya
- Kelemahannya
 - Membutuhkan memori yang cukup banyak
 - Membutuhkan waktu yang cukup lama

Pencarian mendalam pertama (Depth-First Search)

- Proses pencarian dilakukan pada semua anaknya sebelum dilakukan pencarian ke node-node yang selevel
- Keuntungan
 - Memori yang relatif kecil
 - Secara kebetulan, akan menemukan solusi tanpa harus menguji lebih banyak lagi



Pencarian Heuristik

- Pencarian buta tidak selalu dapat diterapkan dengan baik
 - Waktu aksesnya yang cukup lama
 - Besarnya memori yang diperlukan
- 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

Pencarian Heuristik

- Contoh pada masalah 8 puzzle

Keadaan Awal

1	2	3
7	8	4
6		5



Tujuan

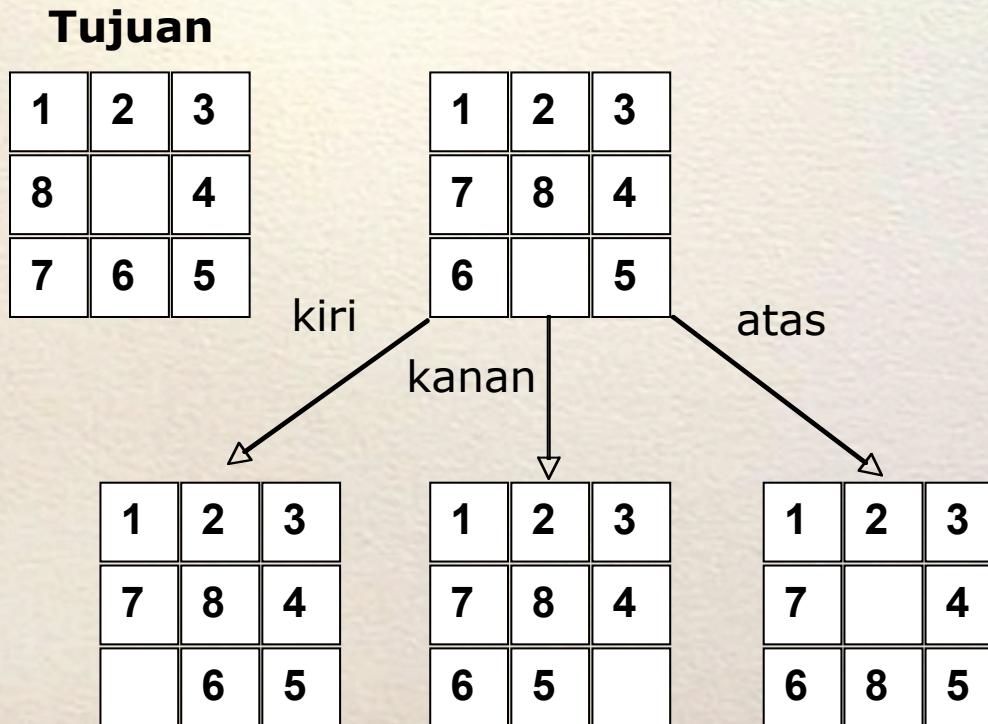
1	2	3
8		4
7	6	5

Pencarian Heuristik

- Operator
 - Ubin kosong geser ke kanan
 - Ubin kosong geser ke kiri
 - Ubin kosong geser ke atas
 - Ubin kosong geser ke bawah

Pencarian Heuristik

- Langkah Awal



Pencarian Heuristik

- Langkah Awal hanya 3 operator yang bisa digunakan
 - Ubin kosong digeser ke kiri, ke kanan dan ke atas.
- Jika menggunakan pencarian buta, tidak perlu mengetahui operasi apa yang akan dikerjakan (sembarang)
- Pada pencarian heuristik perlu diberikan informasi khusus dalam domain tersebut

Informasi yang bisa diberikan

- 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

1	2	3
7	8	4
6		5

1	2	3
7	8	4
	6	5

h=6

kiri

kanan

1	2	3
7	8	4
6	5	

h=4

atas

1	2	3
7		4
6	8	5

h=5

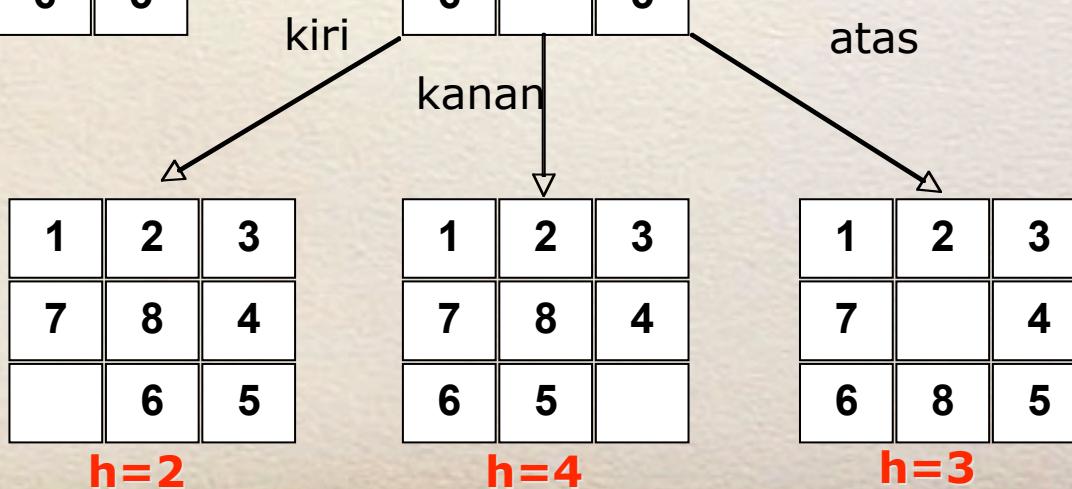
Informasi yang bisa diberikan

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

Tujuan

1	2	3
8		4
7	6	5

1	2	3
7	8	4
6		5



Informasi yang bisa diberikan

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

Tujuan

1	2	3
8		4
7	6	5

1	2	3
7	8	4
6		5

1	2	3
7	8	4
	6	5

$h=2$

kanan

1	2	3
7	8	4
6	5	

$h=4$

atas

1	2	3
7		4
6	8	5

$h=4$

Pencarian Heuristik

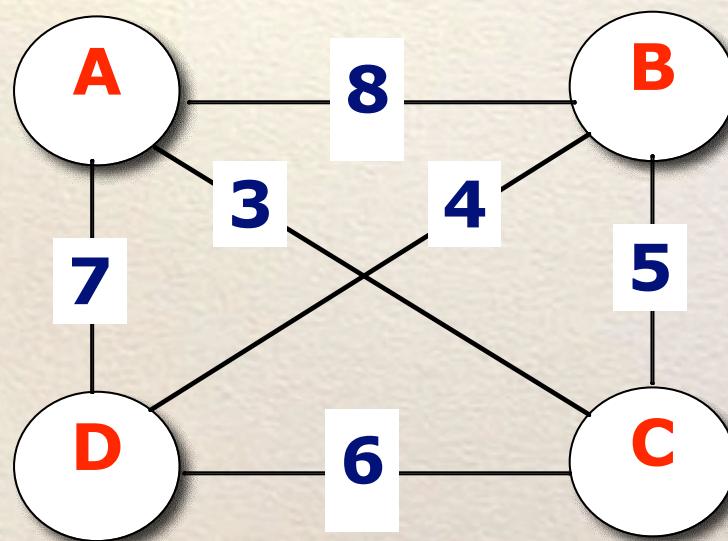
- Ada 4 metode pencarian heuristik
 - Pembangkit & Pengujian (Generate and Test)
 - Pendakian Bukit (Hill Climbing)
 - Pencarian Terbaik Pertama (Best First Search)
 - Simulated Annealing

Pembangkit & Pengujian (Generate and Test)

- Pada prinsipnya metode ini merupakan penggabungan antara depth-first search dengan pelacakan mundur (backtracking), yaitu bergerak ke belakang menuju pada suatu keadaan awal.
- Algoritma:
 - Bangkitkan suatu kemungkinan solusi (membangkitkan suatu titik tertentu atau lintasan tertentu dari keadaan awal).
 - 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.
 - Jika solusi ditemukan, keluar. Jika tidak, ulangi kembali langkah yang pertama.

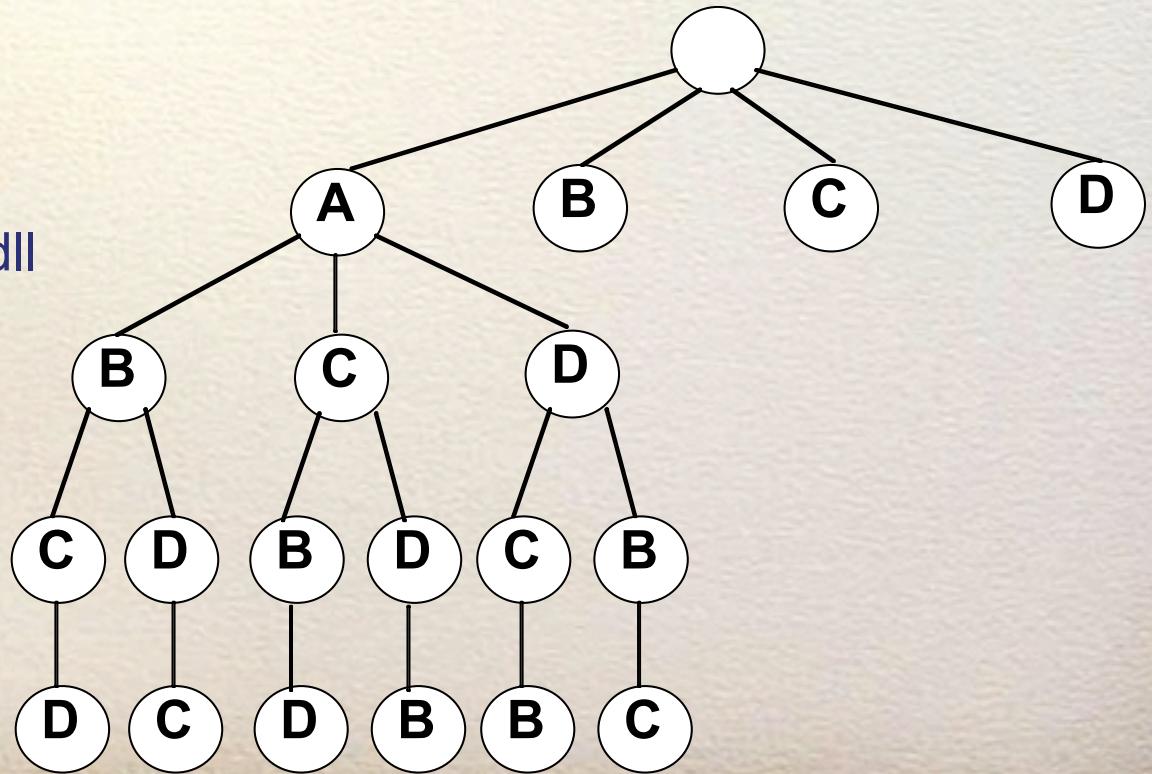
Contoh Traveling Salesman Problem (TSP)

- Seorang salesman ingin mengunjungi n kota. Jarak antara tiap-tiap kota sudah diketahui. Ingin diketahui rute terpendek dimana setiap kota hanya boleh dikunjungi tepat 1 kali.



Contoh Traveling Salesman Problem (TSP)

- Generate & test akan membangkitkan semua solusi yang mungkin:
 - A – B – C – D
 - A – B – D – C
 - A – C – B – D
 - A – C – D – B, dll



Lintasan

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

Pencarian ke-	Lintasan	Panjang Lintasan	Lintasan terpilih	Panjang Lintasan terpilih
13.	CABD	15	ACBD	12
14.	CADB	14	ACBD	12
15.	CBAD	20	ACBD	12
16.	CBDA	16	ACBD	12
17.	CDAB	21	ACBD	12
18.	CDBA	18	ACBD	12
19.	DABC	20	ACBD	12
20.	DACD	15	ACBD	12
21.	DBAC	15	ACBD	12
22.	DBCA	12	ACBD atau DBCA	12
23.	DCAB	17	ACBD atau DBCA	12
24.	DCBA	19	ACBD atau DBCA	12

Pembangkit & Pengujian (Generate and Test)

- Kelemahan
 - Perlu membangkitkan semua kemungkinan sebelum dilakukan pengujian
 - Membutuhkan waktu yang cukup lama dalam pencarinya

Pendakian Bukit (Hill Climbing)

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

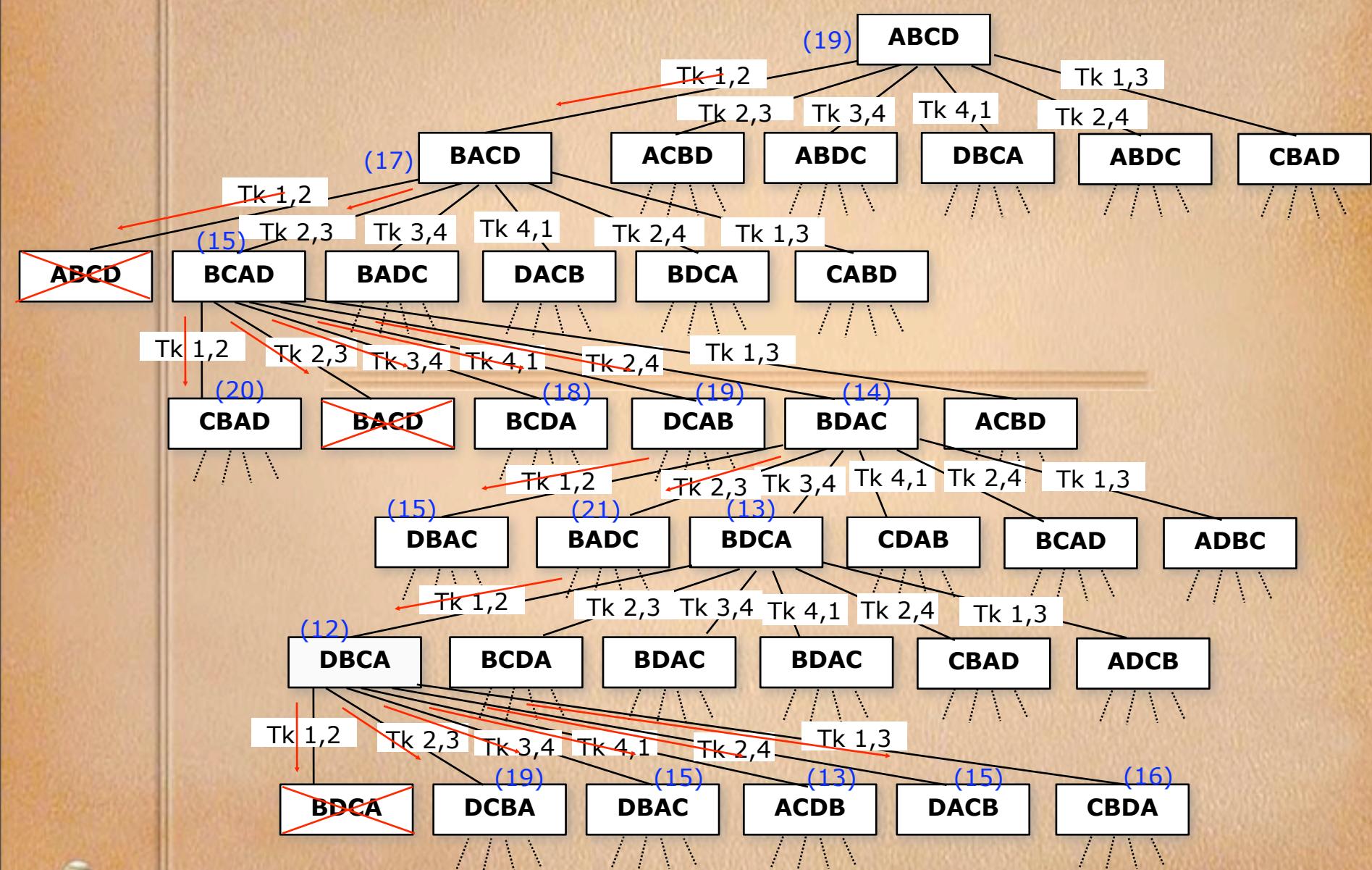
Simple Hill Climbing

- Algoritma
 - 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.

Contoh TSP

- Operator : Tukar kota ke-i dengan kota ke-j (Tk i,j)
- Untuk 4 kota:
 - Tk 1,2 : tukar kota ke-1 dengan kota ke-2.
 - Tk 1,3 : tukar kota ke-1 dengan kota ke-3.
 - Tk 1,4 : tukar kota ke-1 dengan kota ke-4.
 - Tk 2,3 : tukar kota ke-2 dengan kota ke-3.
 - Tk 2,4 : tukar kota ke-2 dengan kota ke-4.
 - Tk 3,4 : tukar kota ke-3 dengan kota ke-4.
- Untuk N kota, akan ada operator sebanyak:

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



Steepest Ascent Hill Climbing

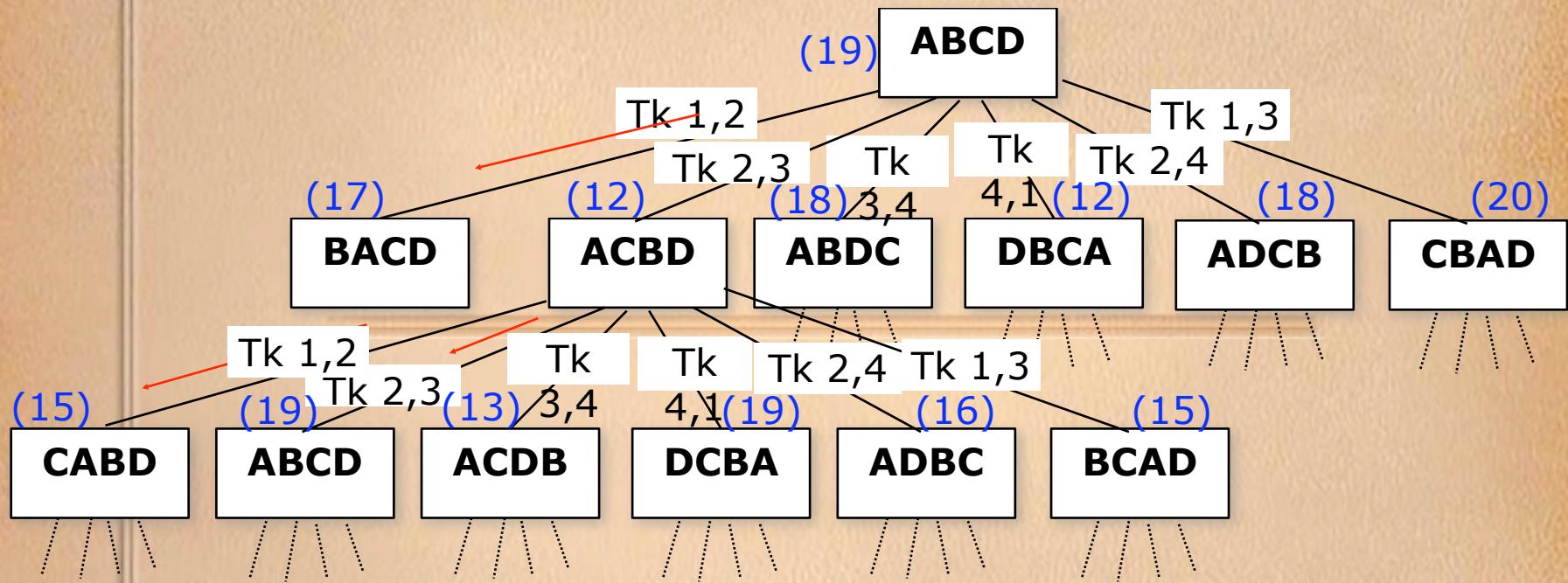
- Steepest-ascent hill climbing sebenarnya hampir sama dengan simple hill climbing, hanya saja gerakan pencarian tidak dimulai dari posisi paling kiri.
- Gerakan selanjutnya dicari berdasarkan nilai heuristik terbaik.
- Dalam hal ini urutan penggunaan operator tidak menentukan penemuan solusi.

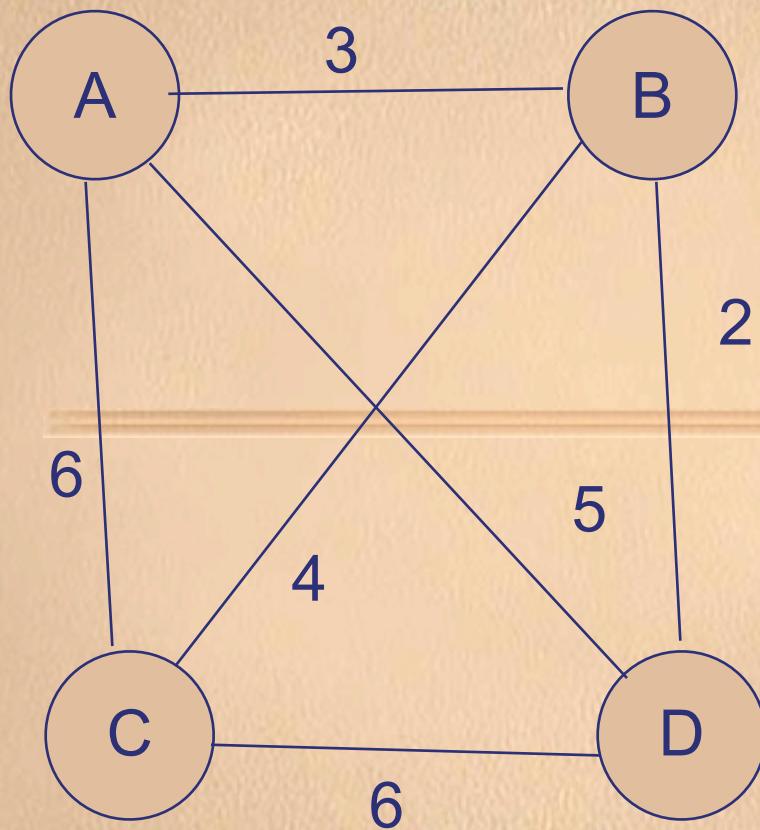
Steepest Ascent Hill Climbing

- 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

- 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.





Any Questions?
