METODE PENCARIAN

Irvanizam Zamanhuri, M.Sc
Zulfan, S.Si, M.Sc
Dalila Husna Yunardi, B.Sc, M.Sc

Jurusan Informatika
Universitas Syiah Kuala

Teknik--Teknik Search (1/3)

- Hal-hal yang muncul dalam teknik pencarian :
  - Arah Search
  - Topologi proses search
  - Penggunaan fungsi heuristik untuk memandu proses tersebut
- ARAH SEARCH
  - Dapat dilakukan :
    - Maju → bermula dari keadaan awal (*start state*)
    - Mundur → diawali dari keadaan tujuan (*goal state*)

Teknik--Teknik Search (2/3)

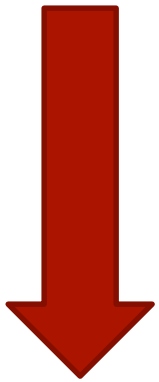
- TOPOLOGI SEARCH
  - Ada 2 macam penggambaran problem, yaitu dalam bentuk :
    - TREE (Tree)
    - Graph
  - TREE
    - Merupakan graf dimana 2 simbol memiliki paling banyak satu lintasan yang menghubungkannya.
    - Tidak ada loop dalam TREE.
    - Contoh : problem ember air.

Teknik--Teknik Search (3/3)

- GRAPH
  - Graph dibedakan menjadi 2 (dua):
    - Graph berarah
    - Graph Tidak berarah
- Teknik searching dalam Kecerdasan Buatan (AI) digunakan untuk mencari solusi dari suatu permasalahan.
- Langkahnya adalah dengan mendefinisikan terlebih dahulu Ruang Masalah (*State*)

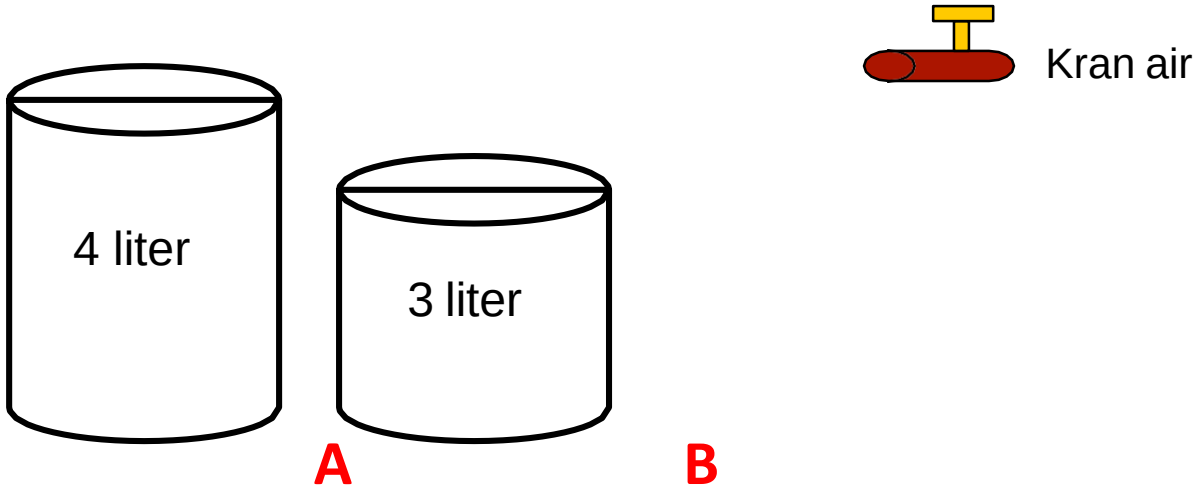
Ruang Masalah

Keadaan Awal (Initial State)



Tujuan (Goal)

Studi Kasus : Masalah Galon Air



- Bagaimana caranya bisa didapatkan air dengan ukuran tepat 2 liter di Galon B?

Studi Kasus : Masalah Galon Air

- Keadaan Awal →→Galon A dan B masih kosong

No	Kejadian yang mungkin untuk masalah galon air
1	Isi penuh galon A
2	Isi penuh galon B
3	Buang sebagian air dari galon A
4	Buang sebagian air dari galon B
5	Kosongkan isi galon A
6	Kosongkan isi galon B
7	Tuang air dari galon A ke galon B sampai galon B penuh
8	Tuang air dari galon B ke galon A sampai galon A penuh
9	Tuang semua air dari galon A ke galon B
10	Tuang semua air dari galon B ke galon A

- Tujuan (Goal) →→Galon B berisi 2 liter air, Galon A berisi n liter air

Studi Kasus : Masalah Galon Air

Galon A	Galon B	No. Kejadian
0 liter	0 liter	Initial State
0 liter	3 liter	2
3 liter	0 liter	10
3 liter	3 liter	2
4 liter	2 liter	8 →→Goal

Performance Searching (1/3)

- Evaluasi sebuah pencarian akan sangat kompleks
- Dasar pengukuran dari evaluasi :
  - Seberapa cepat search menemukan penyelesaian
  - Seberapa cepat search menemukan penyelesaian yang baik
- Kecepatan search ditentukan :
  - Panjang Lintasnya.
  - Jumlah sesungguhnya penulusuran node.

Performance Searching (2/3)

- Untuk mengukur performansi metode pencarian, terdapat 4 kriteria yang dapat digunakan :
  - Completeness
    - Apakah solusi pasti ditemukan?
  - Time Complexity
    - Berapa lama waktu yang diperlukan
  - Space Complexity
    - Berapa banyak memori yang dibutuhkan
  - Optimally
    - Apakah solusi yang ditemukan adalah solusi yang terbaik?

Performance Searching (3/3)

- Time& Space complexity diukur berdasarkan :
  - *b* → faktor percabangan dari search tree
  - *d* → depth (kedalaman) dari solusi optimal
  - *m* → kedalaman maksimum dari search tree (bisa infinite)

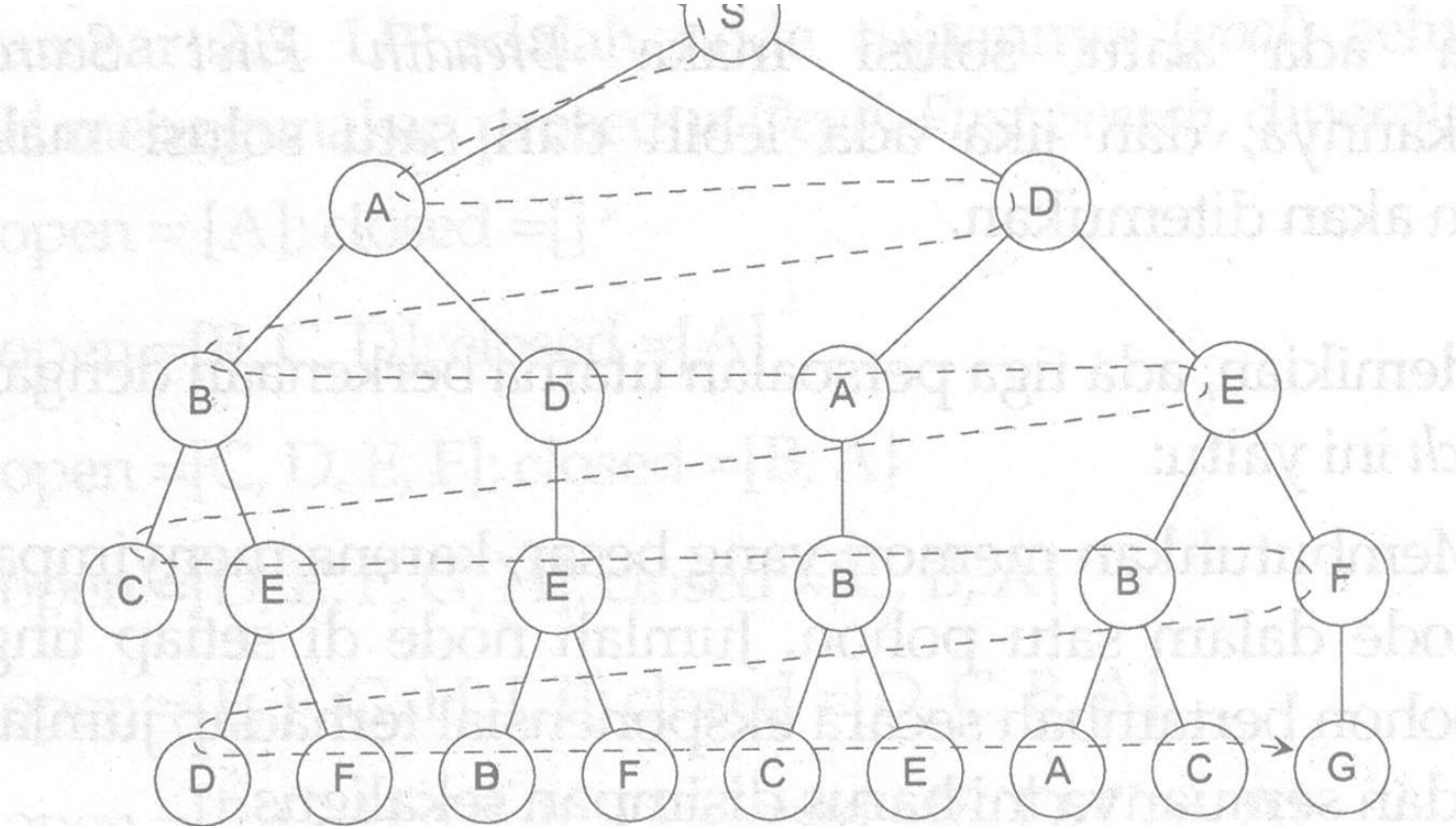
Jenis Teknik Pencarian

- Ada beberapa teknik untuk mencari kemungkinan penyelesaian, yaitu :
  - Blind Search (Uninformed Search)
    - *Depth First Search (DFS)*
    - *Breadth First Search (BFS)*
    - *Uniform Cost Search (UCS)*
    - *Depth Limited Search (DLS)*
    - *Iterative Deepening Search (IDS)*
  - *Heuristik Search (Informed Search)*
    - *Hill--Climbing Search*
    - *Least--Cost Search*
    - *Best First Search*

BLIND SEARCH (Breadth First Search)

- Pada metode ini diperiksa setiap node pada level yang sama sebelum mengolah ke level berikutnya yang lebih dalam.
- Pencarian dilakukan pada semua simpul dalam setiap level secara berurutan dari kiri →→kanan.
- Jika pada satu level belum ditemukan solusinya, maka pencarian dilanjutkan pada level berikutnya.

BLIND SEARCH (Breadth First Search)



BLIND SEARCH (Breadth First Search)

- Keuntungan Breadth First Search :
  - Tidak akan menemui jalan buntu
  - Jika ada solusi, maka Breadth First Search akan menemukannya, jika lebih dari satu maka solusi akan ditemukan
- Kelemahan Breadth First Search
  - Membutuhkan memori yang cukup banyak, karena menyimpan semua node dalam satu pohon
  - Membutuhkan waktu yang cukup lama, karena menguji n level untuk mendapatkan solusi pada level yang ke(n+1)

BLIND SEARCH (Breadth First Search)

- Sifat Breadth First Search :
  - Complete
    - Ya, jika *b* terbatas
  - Time Complexity

$$b + b^2 + b^3 + \ldots + b^d = b(b^d ! \ 1) / (b ! \ 1) = O\left(b^{d+1}\right) = O\left(b^d\right)$$

  - Space Complexity

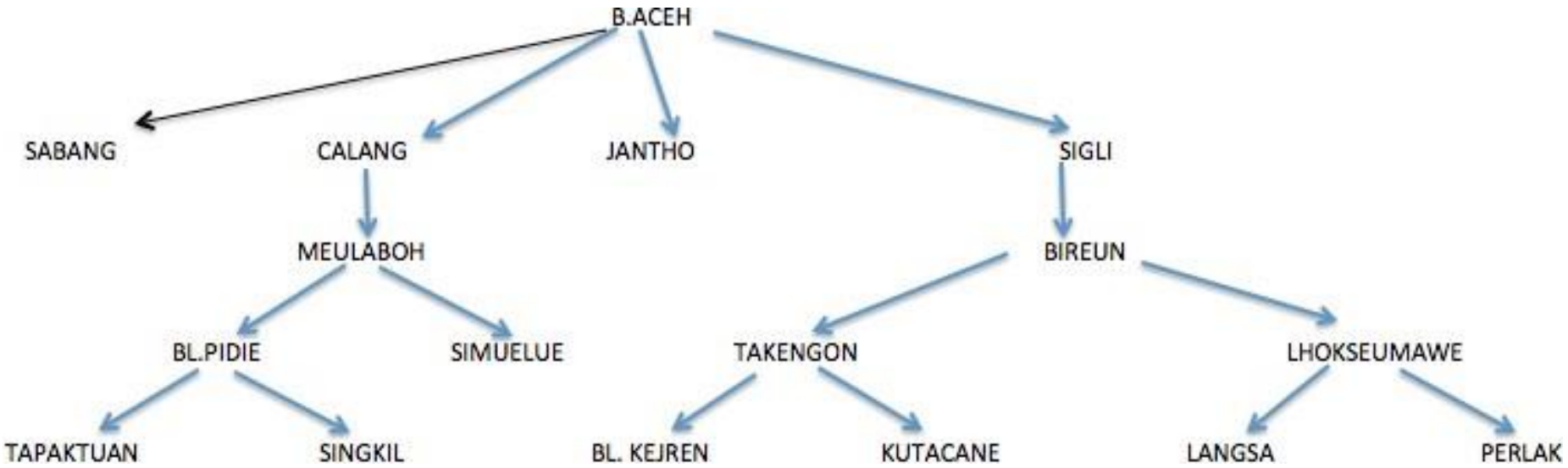
$$O\left(b^d\right)$$

  - Optimal
    - Ya, jika semua step cost sama, tapi pada umumnya tidak optimal

Peta Aceh

- Contoh Kasus :
  - B.Aceh–Sabang : 1000 KM
  - Sabang–Calang : 1000KM
  - Calang–Jantho : 800 KM
  - Jantho–Sigli : 1900 KM
  - Sigli–Meulaboh : 1500 KM
  - Meulaboh–Bireuen : 1800 KM
  - Bireuen–Bl.Pidie : 500 KM
  - Bl.Pidie–Simeulu : 1000 KM
  - Simeulue–Takengon : 1500 KM
  - Takengon–Lhokseumawe: 1500 KM
  - Lhokseumawe–Tapaktua : 1000KM
  - Tapaktuan--Singkil : 800KM
  - Singkil--Bl.Kejren : 900KM
  - Bl.Kejren-Kutacane : 700KM
  - Kutacane--Langsa : 900KM
  - Langsa--Perlak :1000KM

Tree : Kasus Peta Aceh

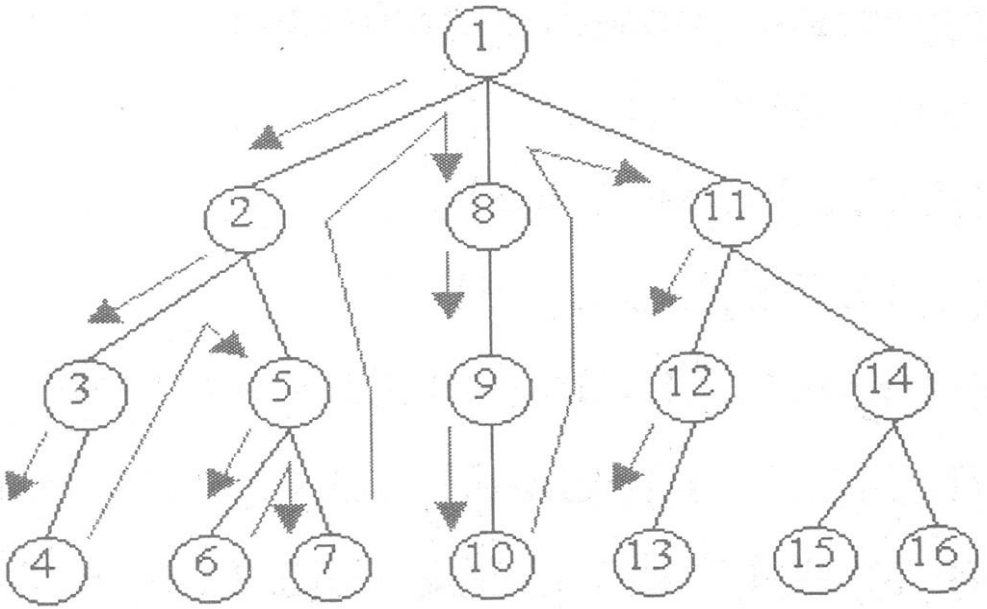


BLIND SEARCH (Depth First Search)

- Pencarian dilakukan pada suatu simpul dalam setiap level dari yang paling kiri.
- Jika pada level yang terdalam, solusi belum ditemukan, 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

BLIND SEARCH (Depth First Search)

- Metode pencarian dapat dilihat seperti gambar :



BLIND SEARCH (Depth First Search)

- Keuntungan :
  - Membutuhkan memori relatif kecil, karena hanya node--node pada lintasan yang aktif saja yang disimpan
  - Secara kebetulan, metode ini akan menemukan solusi tanpa harus menguji lebih banyak
- Kerugian :
  - Memungkinkan tidak ditemukannya tujuan yang diharapkan
  - Hanya akan mendapatkan solusi pada tiap pencarian

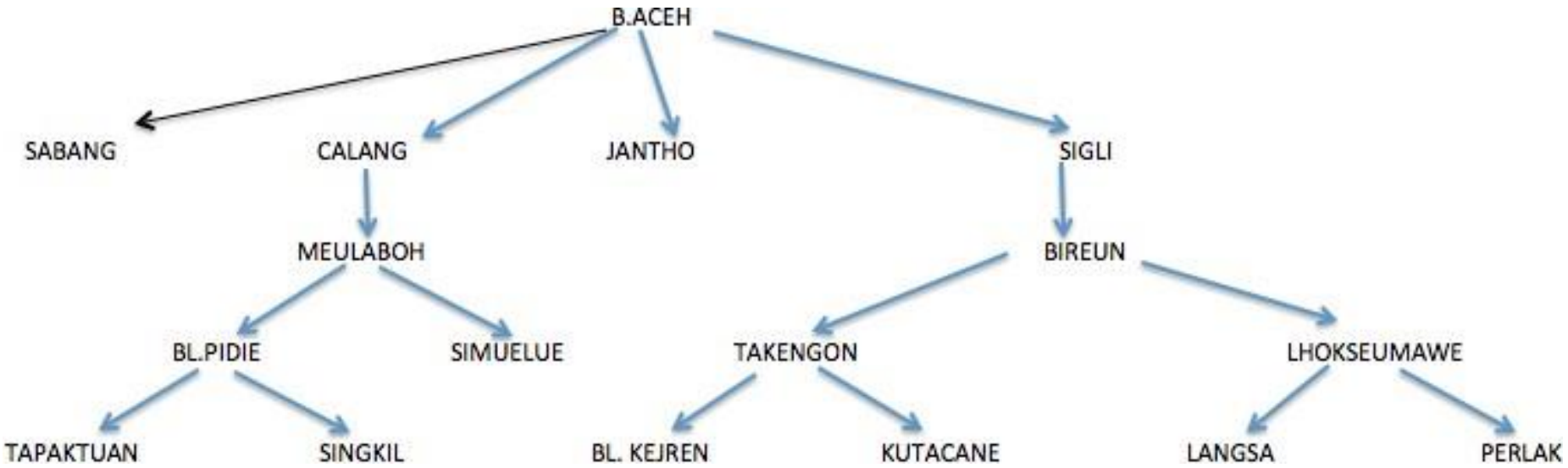
BLIND SEARCH (Depth First Search)

- Sifat Depth First Search
  - Complete
    - Tidak Commplete, jika pohon yang dibangkitkan mempunyai level yang sangat dalam (tidak terhingga)
  - Time Complexity $O(b^m)$
  - Space Complexity $O(bm)$
  - Optimal
    - Tidak optimal, 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

Peta Aceh

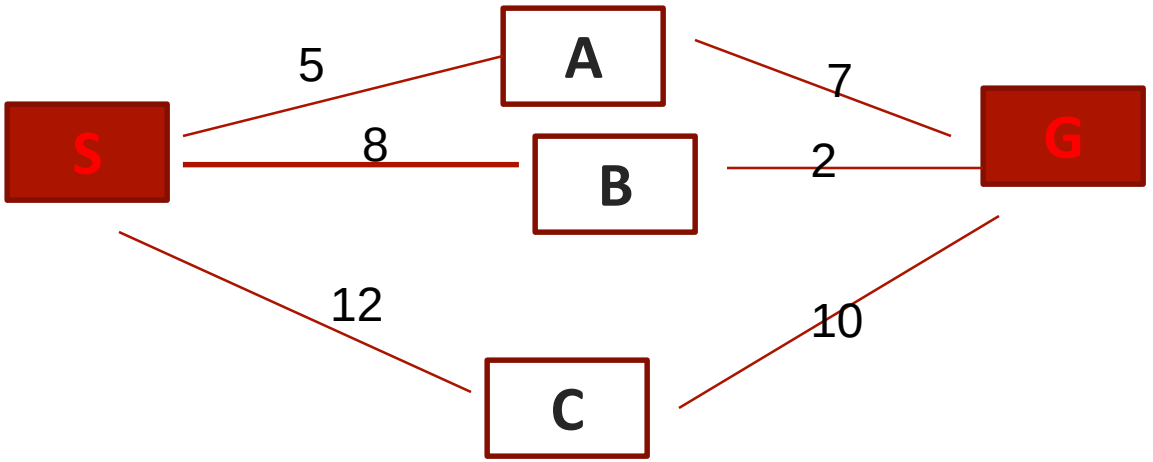
- Contoh Kasus :
  - B.Aceh–Sabang : 1000 KM
  - B.Aceh–Calang : 1000KM
  - Calang–Meulaboh : 800 KM
  - Meulaboh–Bl.Pidie : 1900 KM
  - Bl.Pidie–Tapaktuan : 1500 KM
  - Bl.Pidie–Singkil : 1800 KM
  - Meulaboh–Simulue : 500 KM
  - B.Aceh–Jantho : 1000 KM
  - B.Aceh–Sigli : 1500 KM
  - Sigli–Bireuen : 1500 KM
  - Bireuen–Takengon : 1000KM
  - Mtakengon--Bl.Kejren : 800KM
  - Takengon--Kutacane : 900KM
  - Bireuen--Lhokseumawe : 700KM
  - Lhokseumawe--Langsa : 900KM
  - Lhokseumawe--Perlak :1000KM

Tree : Kasus Peta Aceh



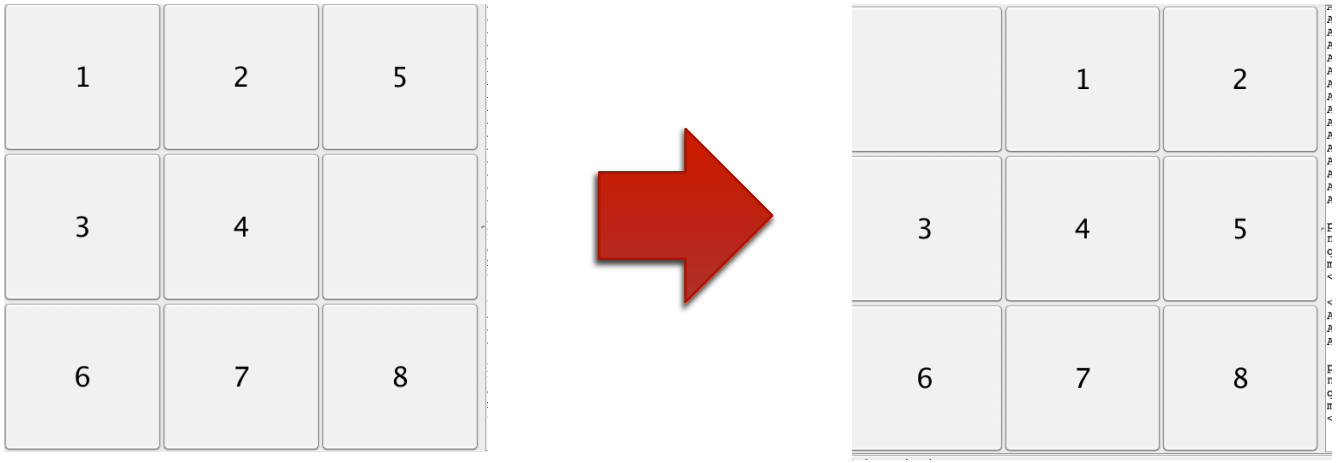
BLIND SEARCH (Uniform Cost Search)

- Konsepnya hampir sama dengan BFS
- Pada UCS, menggunakan urutan biaya dari yang paling kecil sampai terbesar
- UCS berusaha untuk menemukan solusi dengan total biaya terendah.



Latihan: (Problem Solving Agent)

- 8--Puzzle Problem :
- Contoh 8--puzzle, puzzle ukuran 3x3 dengan 8 angka dan satu buah spasi kosong.
- Goal : letakkan angka tersebut berurutan seperti gambar berikut:



Referensi

- Sebagian besar materi(slide) disiapkan oleh (Sekolah Tinggi Ilmu Komputer Indonesia (STIKI) Malang.
- George F. Luger, Artificial Intelligence, Addison Wesley, Fourth Edition.
- Stuart Russell & Peter Norvig, Artificial Intelligence: A Modern Approach, Third Edition.