Uninformed Search

Metode Pelacakan / 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.

Kriteria

- 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

- Pencarian buta (uninformed/blind search) : tidak ada informasi awal yang digunakan dalam proses pencarian
  - Pencarian melebar pertama (Breadth – First Search)
  - Pencarian mendalam pertama (Depth – First Search)
- Pencarian terbimbing (informed/heuristic search) : adanya informasi awal yang digunakan dalam proses pencarian
  - A*

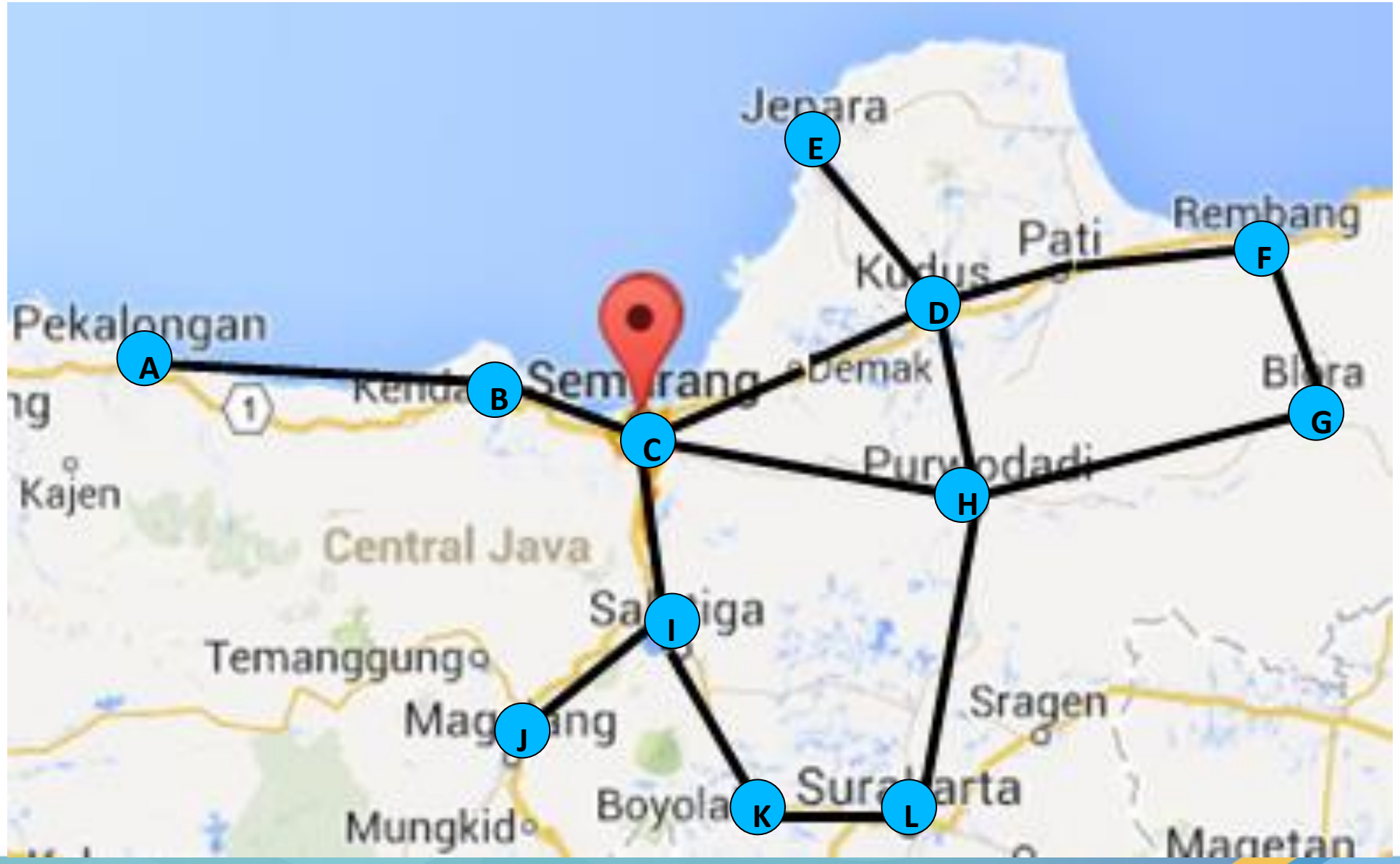
Breadth First Search

- Menggunakan FIFO (Antrian / Queue)
- Algoritma :
  1. Masukkan node akar ke queue
  2. Ambil node dari awal queue dan cek apakah merupakan solusi
  3. Jika solusi maka pencarian selesai, return hasil
  4. Jika bukan solusi, masukkan seluruh node anak ke dalam queue
  5. Jika queue kosong dan tiap node sudah dikunjungi, pencarian selesai
  6. Jika queue tidak kosong, ulang pencarian mulai dari poin 2

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 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

Contoh : Dari Semarang (C) ke Surakarta (L)



Langkah Penyelesaian

Sebelum mulai mengerjakan, perhatikan dulu keterangan berikut.

Node dilambangkan dengan 1 huruf, seperti A, B dst

Jalur dilambangkan dengan urutan node, misal ABC maka urutannya/jalurnya adalah node A, lalu ke B, lalu ke C.

Jika ada antrian semacam ini : AC AB ABC, maka cara membacanya :

- ada 3 jalur yang antri, yaitu AC, AB dan ABC
- antrian paling depan adalah paling kiri, jadi AC adalah antrian paling depan.
- Sebaliknya antrian paling belakang di sebelah paling kanan. ABC berarti paling belakang / terakhir.
- jika ada jalur baru masuk antrian maka dia masuk di sebelah kanan. Contoh : AC AB ABC kemudian ada jalur baru ABCD masuk antrian, maka dia masuk dari kanan -> AC AB ABC ABCD
- Jika sudah diproses maka jalur akan hilang dari antrian Misal : AC AB ABC ABCD jika AC(antrian paling depan) sudah diproses maka hilang dari antrian, ganti selanjutnya : AB ABC ABCD

Langkah Penyelesaian

1. Masukkan C ke queue, beri tanda visited untuk C, cek apakah C adalah tujuan
   - **C**, visited C
2. C bukan tujuan, masukkan cabang dari C ke queue, proses antrian paling depan dan set visited
   - **CH** CI CD CB, visited C,H
3. H bukan tujuan, masukkan cabang dari H ke queue, proses antrian paling depan dan set visited
   - **CI** CD CB CHL CHG CHD, visited C,H,I
4. I bukan tujuan, masukkan cabang dari I ke queue, proses antrian paling depan dan set visited
   - **CD** CB CHL CHG CHD CIK CIJ, visited C,H,I,D
5. D bukan tujuan, masukkan cabang dari D ke queue, proses antrian paling depan dan set visited
   - **CB** CHL CHG CHD CIK CIJ CDH CDF CDE, visited C,H,I,D,B
6. B bukan tujuan, masukkan cabang dari B ke queue, proses antrian paling depan dan set visited
   - **CHL** CHG CHD CIK CIJ CDH CDF CDE CBA, visited C,H,I,D,B,L
7. L adalah tujuan maka masukkan CHL sebagai jalur yang dipilih

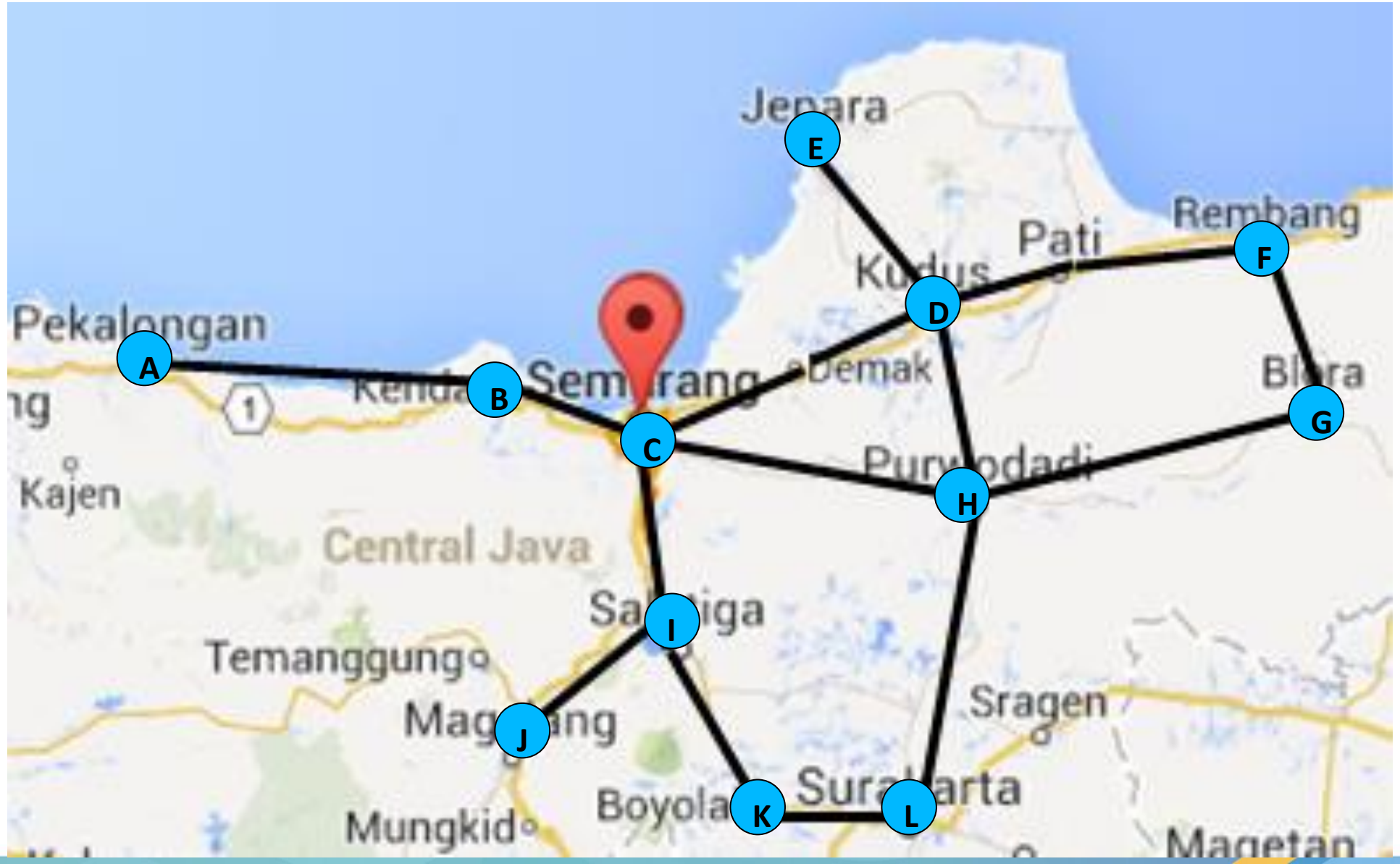
Depth First Search

- Pencarian dilakukan pada salah satu cabang dan mencarinya sampai level yang paling dalam dari cabang tersebut.
- Menggunakan LIFO (stack)
- Algoritma :
  1. Tempatkan node awal di atas stack
  2. Jika stack kosong, stop pencarian, return failure
  3. Jika node merupakan goal, maka stop pencarian dan return success
  4. Jika node di atas stack bukan merupakan solusi, expand dan tempatkan anak2nya di atas tumpukan
  5. Selama belum ditemukan solusi, kembali ke langkah 2

Depth First Search

- 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).

Contoh : Dari Semarang (C) ke Surakarta (L)



Langkah Penyelesaian

Sebelum mulai mengerjakan, perhatikan dulu keterangan berikut.

Node dilambangkan dengan 1 huruf, seperti A, B dst

Jalur dilambangkan dengan urutan node, misal ABC maka urutannya/jalurnya adalah node A, lalu ke B, lalu ke C.

Jika ada antrian semacam ini : AC AB ABC, maka cara membacanya :

- ada 3 jalur yang antri, yaitu AC, AB dan ABC
- antrian paling depan adalah paling kiri, jadi AC adalah antrian paling depan.
- Sebaliknya antrian paling belakang di sebelah paling kanan. ABC berarti paling belakang / terakhir.
- jika ada jalur baru masuk antrian maka dia masuk di sebelah kanan. Contoh : AC AB ABC kemudian ada jalur baru ABCD masuk antrian, maka dia masuk dari kanan -> AC AB ABC ABCD
- Jika sudah diproses maka jalur akan hilang dari antrian Misal : AC AB ABC ABCD jika AC(antrian paling depan) sudah diproses maka hilang dari antrian, ganti selanjutnya : AB ABC ABCD

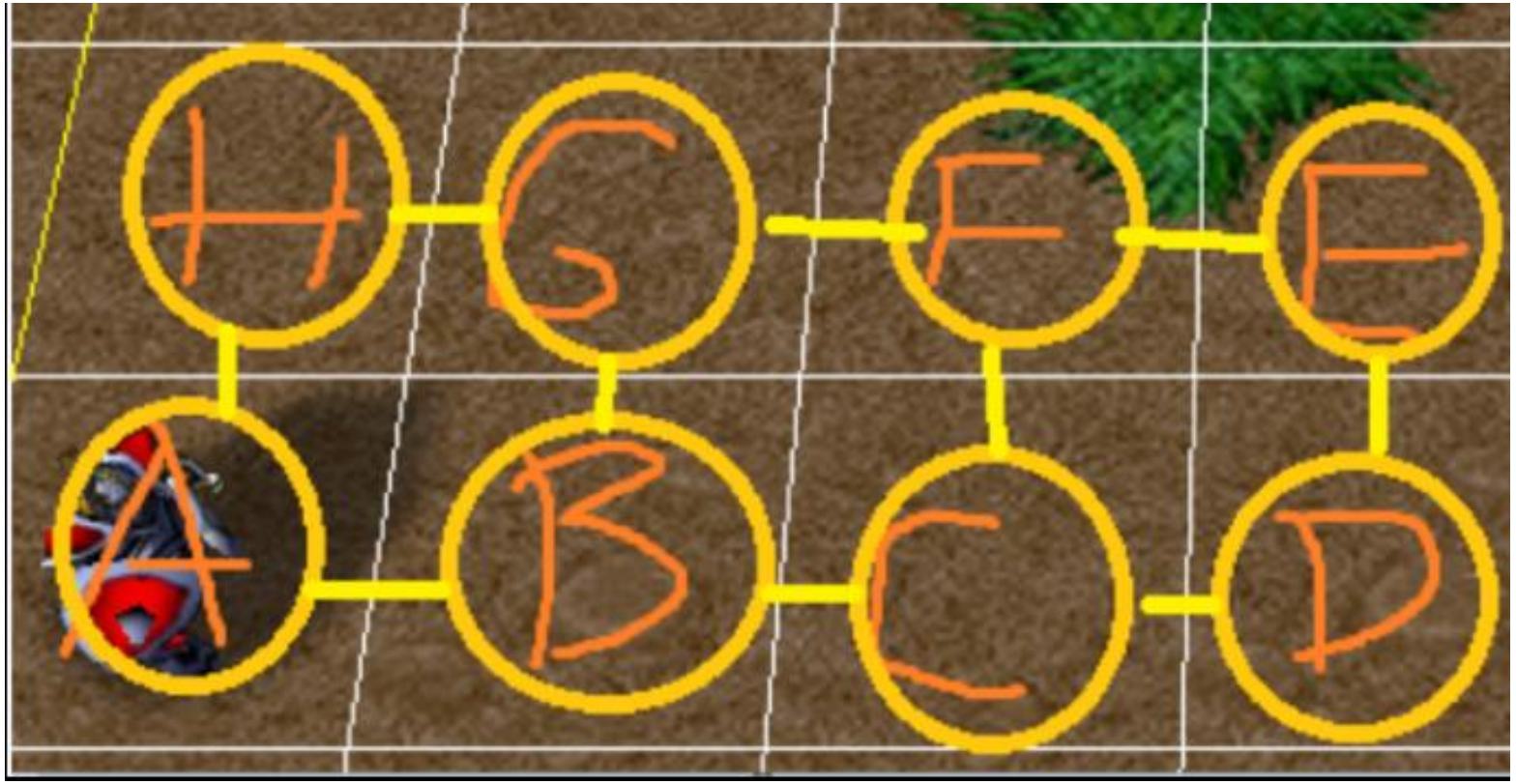
Langkah Penyelesaian

Keterangan : bedanya dengan BFS, disini yang dipilih adalah urutan paling belakang, yaitu paling kanan. Jika BFS yang dicek urutan pertama (paling kiri)

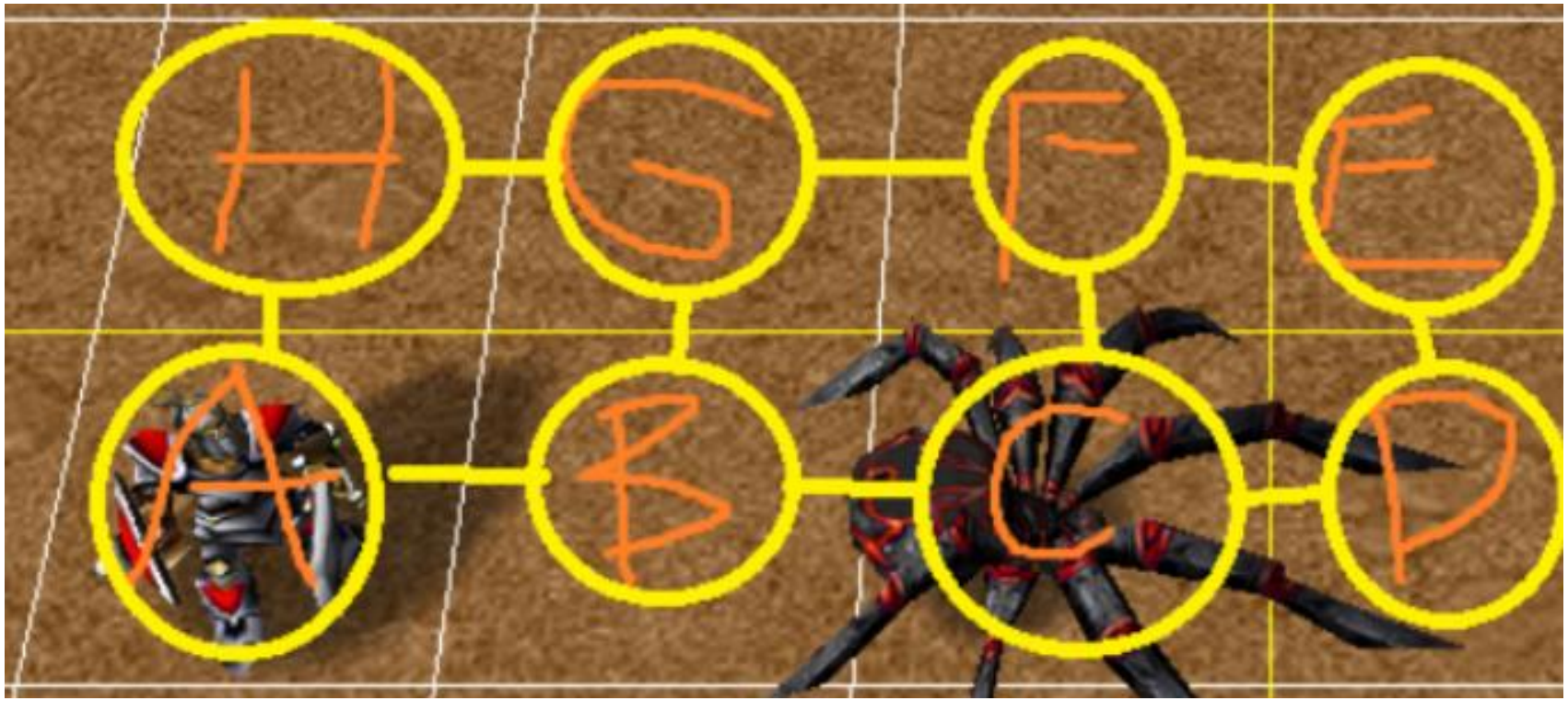
1. Masukkan C ke stack, beri tanda visited untuk C, cek apakah C adalah tujuan
   - **C**, visited C
2. C bukan tujuan, masukkan cabang dari C ke stack, proses node paling belakang dan set visited
   - CB CH CD **CI**, visited C,I
3. I bukan tujuan, masukkan cabang dari I ke stack, proses node paling belakang dan set visited
   - CB CH CD CIJ **CIK**, visited C,I,K
4. K bukan tujuan, masukkan cabang dari K ke stack, proses node paling belakang dan set visited
   - CB CH CD CIJ **CIKL**, visited C,I,K,L
5. L adalah tujuan maka jalur yang dipilih adalah CIKL

CONTOH LAIN : BAGAIMANA REPRESENTASI GRAF DARI PERMASALAHAN BERIKUT ?

BUAT JADI GRAF : BERI NODE DAN EDGE

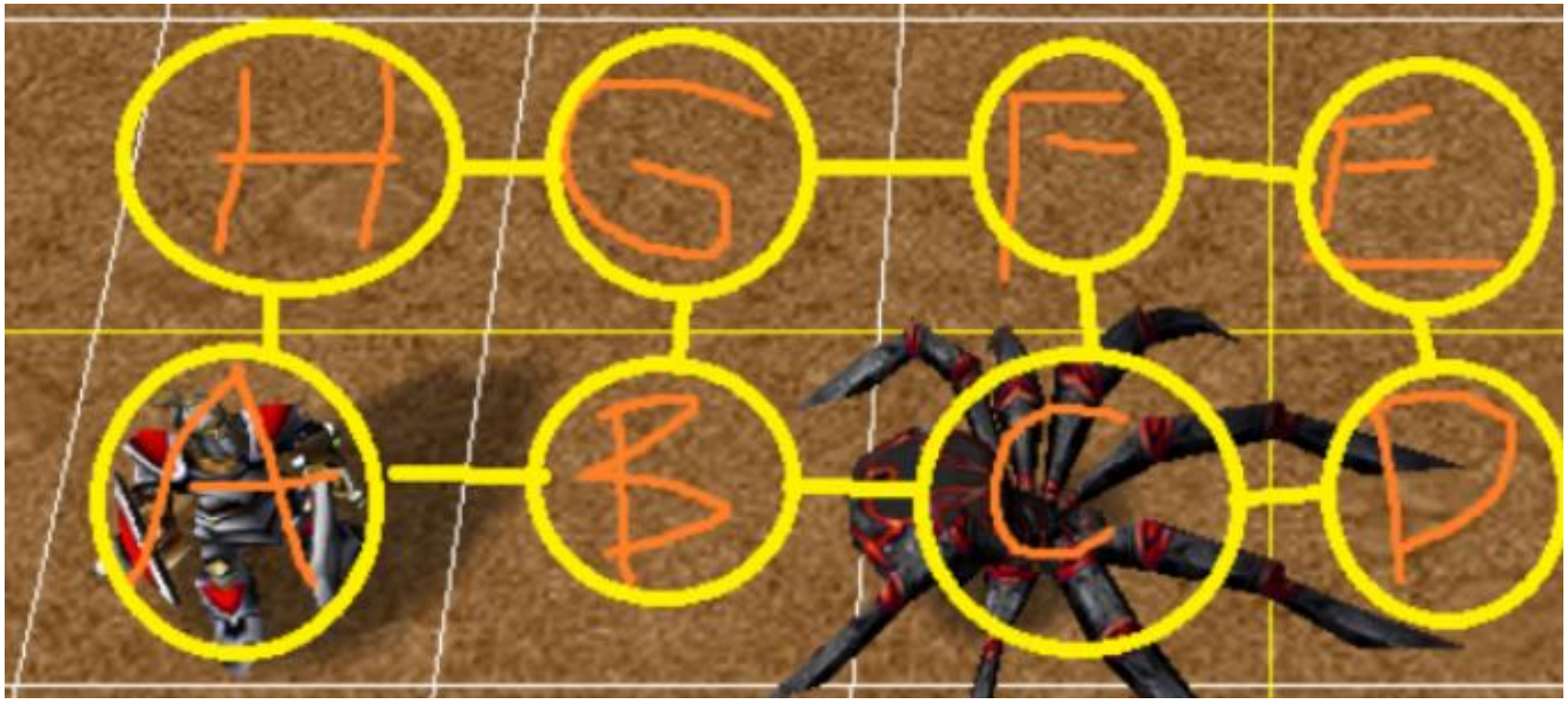


MISAL ADA MUSUH DI NODE C : BAGAIMANA KNIGHT DI NODE A MENGHAMPIRI SPIDER DI NODE C ?



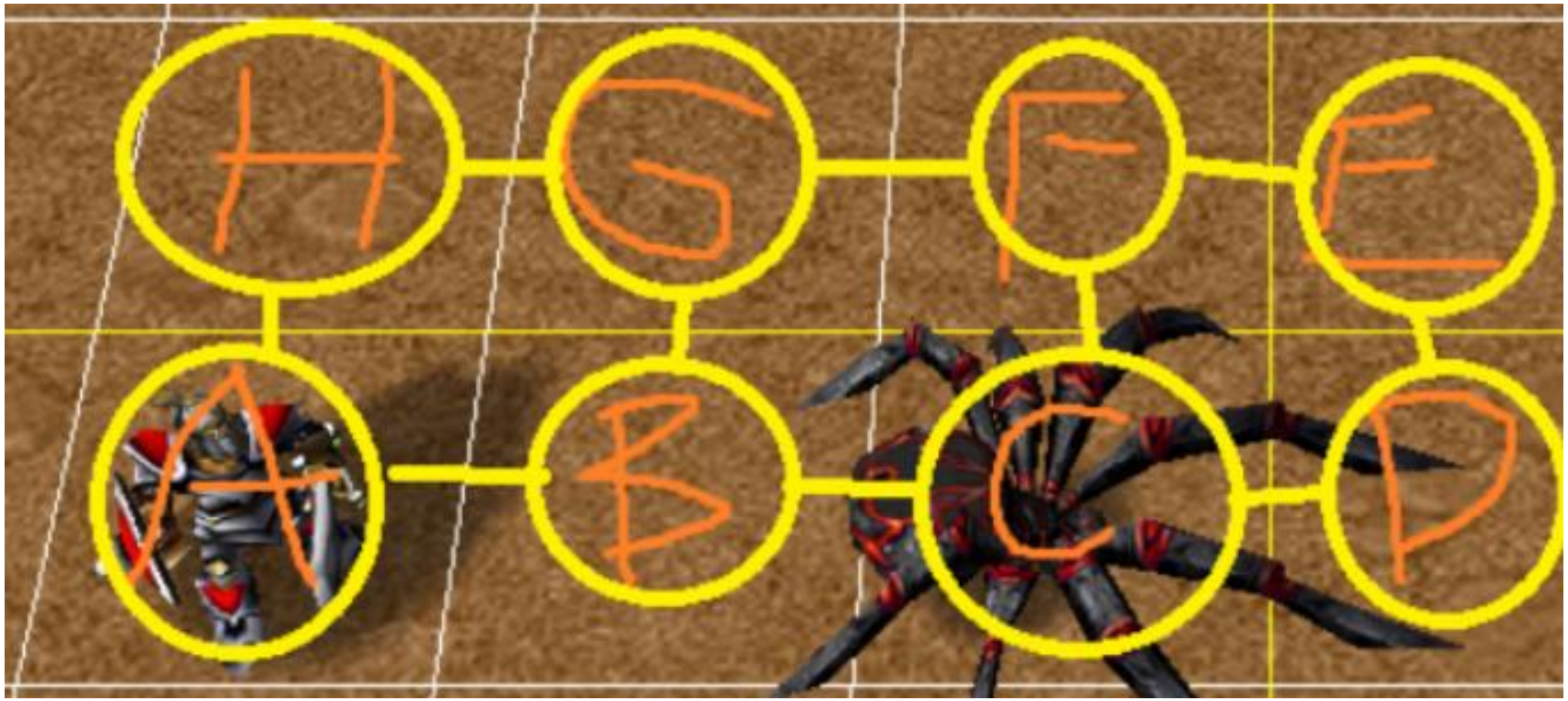
START : A

GOAL / TUJUAN : C



START : A

GOAL / TUJUAN : C



LANGKAH PENYELESAIAN DENGAN BFS

1. **A**, sudah dicek : A
2. **AB** AH, sudah dicek : A; B
3. **AH** ABC ABG, sudah dicek : A; B; H
4. **ABC** ABG AHG, sudah dicek : A; B; H; C

Berarti sekarang C dicek. Tandai sudah dicek. C adalah tujuan. Maka AI akan memilih ABC sebagai jalur dari A ke C.

LANGKAH PENYELESAIAN DENGAN DFS

1. **A**, SUDAH DICEK : A
2. AB **AH**, SUDAH DICEK : A, H
3. AB **AHG**, SUDAH DICEK : A, H, G
4. AB AHGB **AHGF**, SUDAH DICEK : A, H, G, F
5. AB AHGB AHGFE **AHGFC**, SUDAH DICEK : A, H, G, F, C

JADI JALURNYA ADALAH **AHGFC**

KELEBIHAN DARI BFS :

- MENGHASILKAN JALUR YANG PENDEK (NOTE : BUKAN JARAK TAPI URUTAN NODE). SEBAGAI CONTOH : MISAL DARI A KE C ADA JALUR AC DAN ABC, MAKA AC ITU YANG PALING PENDEK. BFS AKAN PASTI MENEMUKAN YANG AC (YANG PALING PENDEK)

KELEMAHAN DARI BFS :

- PEMROSESAN BERAT, KARENA SELALU CEK SEMUA CABANG TERDEKAT.

KELEBIHAN DARI DFS :

- JIKA MENEMUKAN JALUR YANG TEPAT MAKA PEMROSESAN AKAN SANGAT CEPAT
- DIA TIDAK MENGECEK SEMUA CABANG TERDEKATNYA - BISA MENGHASILKAN VARIASI JALUR

KELEMAHAN DARI DFS :

- JIKA CABANG YANG DIPILIH TIDAK MENGANDUNG TUJUAN DAN TERLALU DALAM