SEARCHING FOR SOLUTIONS

Sistem Cerdas dan Pendukung Keputusan

Dr. Herlina Jayadianti, S.T.,M.T. & Andiko Putro S., S.Kom., M.Cs.

Klasifikasi Metode Pencarian

- 2 kategori utama:
  - *uninformed (blind) search;*
  - *informed (heuristic) search.*
- Contoh *Blind Search:*
  - *breadth-first;*
  - *depth-first;*
  - *iterative deepening depth-first;*
  - *bidirectional;*
  - *branch and bound (uniform cost search).*
- Contoh *Heuristc Search:*
  - *hill climbing;*
  - *beam search*
  - *greedy;*
  - *best first search*
  - *heuristics.*

Definisi Pencarian

- Pemeriksaan sistematis dari status-status untuk menemukan suatu jalur dari status awal ke status tujuan.
- *Search space (ruang pencarian) terdiri dari* himpunan status yang mungkin, dan operator yang mendefinisikan sambungannya.
- **Solusi: Jalur dari status awal ke status yang** memenuhi uji tujuan.
- Ada 3 kelas teknik/metode:
  - Menemukan suatu jalur awal $\rightarrow$ tujuan
  - Menemukan jalur terbaik
  - Buntu, tidak mendapatkan tujuan

Strategi Pencarian

Kriteria dalam strategi pencarian:

- **Completeness:** Apakah strategi tersebut menjamin penemuan solusi jika solusinya memang ada?
- **Time complexity:** Berapa lama waktu yang diperlukan?
- **Space complexity:** Berapa banyak memori yang diperlukan?
- **Optimality:** Apakah strategi tersebut menemukan solusi yang paling baik jika terdapat beberapa solusi berbeda pada permasalahan yang ada?

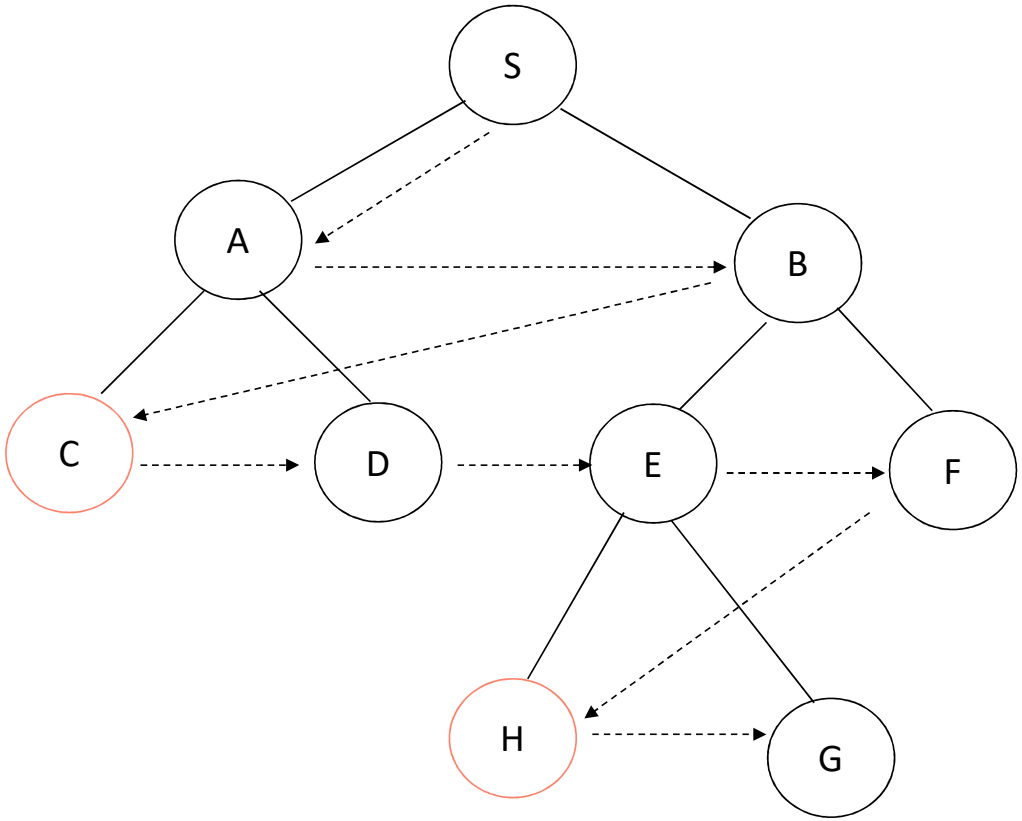
Breadth-First Search (BFS)

- **Pencarian melebar lebih dulu**

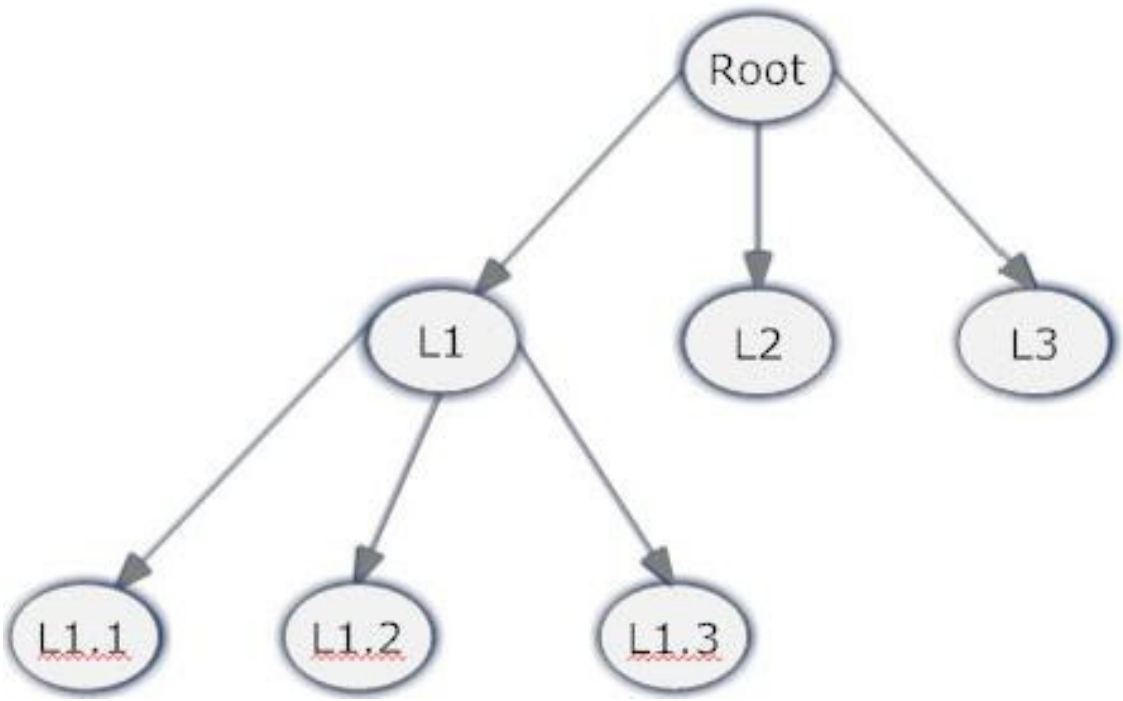
- Setiap status pada layer/level yang diberikan di*expand sebelum* bergerak ke status-status pada layer berikutnya.

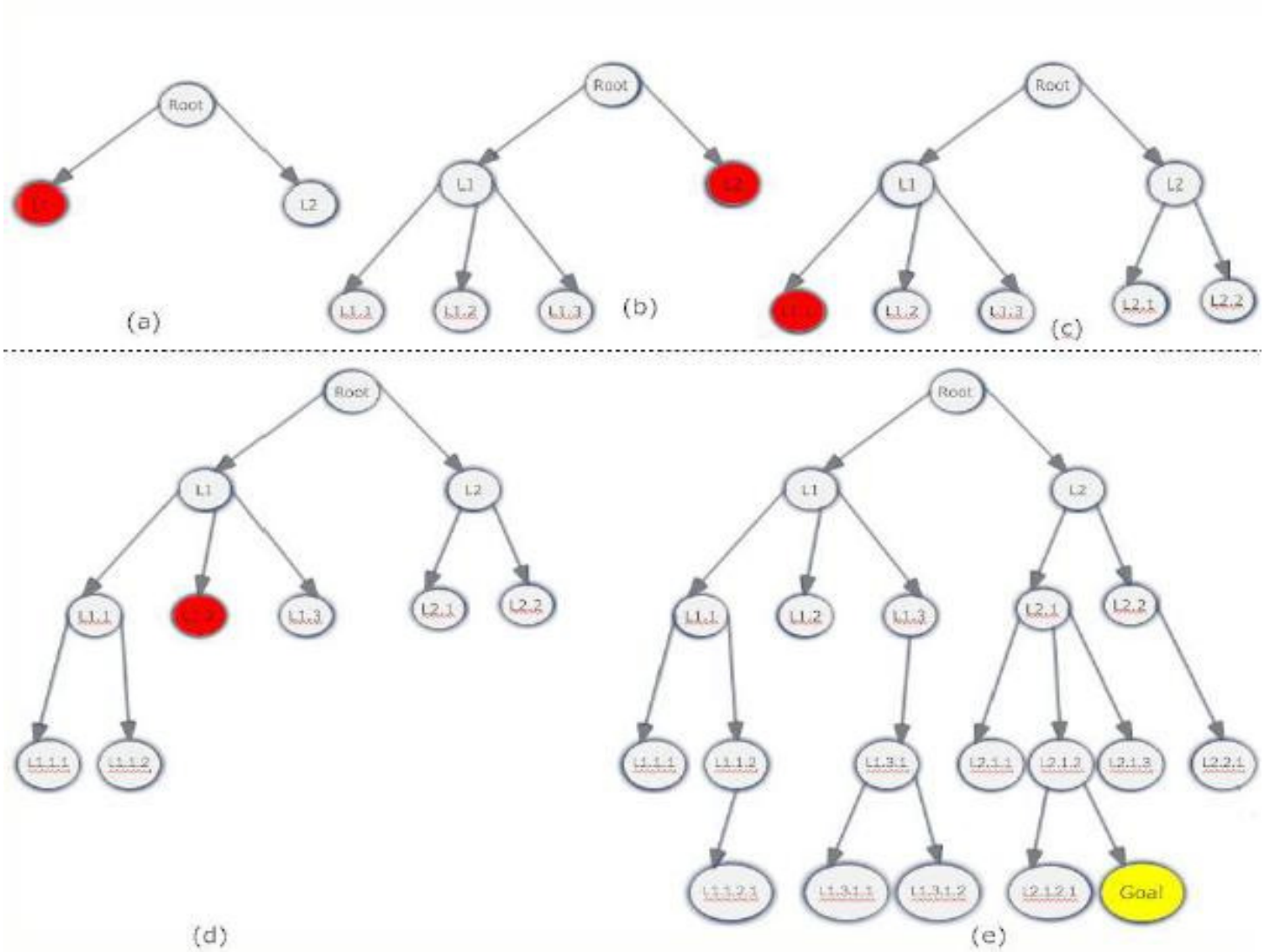
- Cocok untuk pohon yang tak seimbang, tetapi boros jika semua node tujuan ada pada level yang dalam dan di sebelah kiri.

- Pencarian dilakukan pada semua node dalam setiap level secara berurutan dari kiri ke kanan. Jika pada satu level belum ditemukan solusi, maka pencarian dilanjutkan pada level berikutnya. Demikian seterusnya sampai ditemukan solusi.



Pohon Breadth First Search


Root
L1
L2
L3
L1.1
L1.2
L1.3


Root
L1
L2
(a)
L1.1
L1.2
L1.3
(b)
(c)
L2.1
L2.2
(d)
L1.1.1
L1.1.2
L1.3.1
L2.1.1
L2.1.2
L2.2.1
L1.1.2.1
L1.3.1.1
L1.3.1.2
L2.1.2.1
Goal
(e)

- Keuntungan BFS
  - Tidak akan menemui jalan buntu
  - Jika ada satu solusi, maka BFS akan menemukannya. Jika ada lebih dari satu solusi, maka solusi minimum akan ditemukan.

- Kelemahan BFS
  - Memori banyak (untuk menyimpan semua node dalam satu pohon)
  - Waktu lama (menguji n level untuk mendapatkan solusi pada level (n+1)

2. Depth-First Search (DFS)

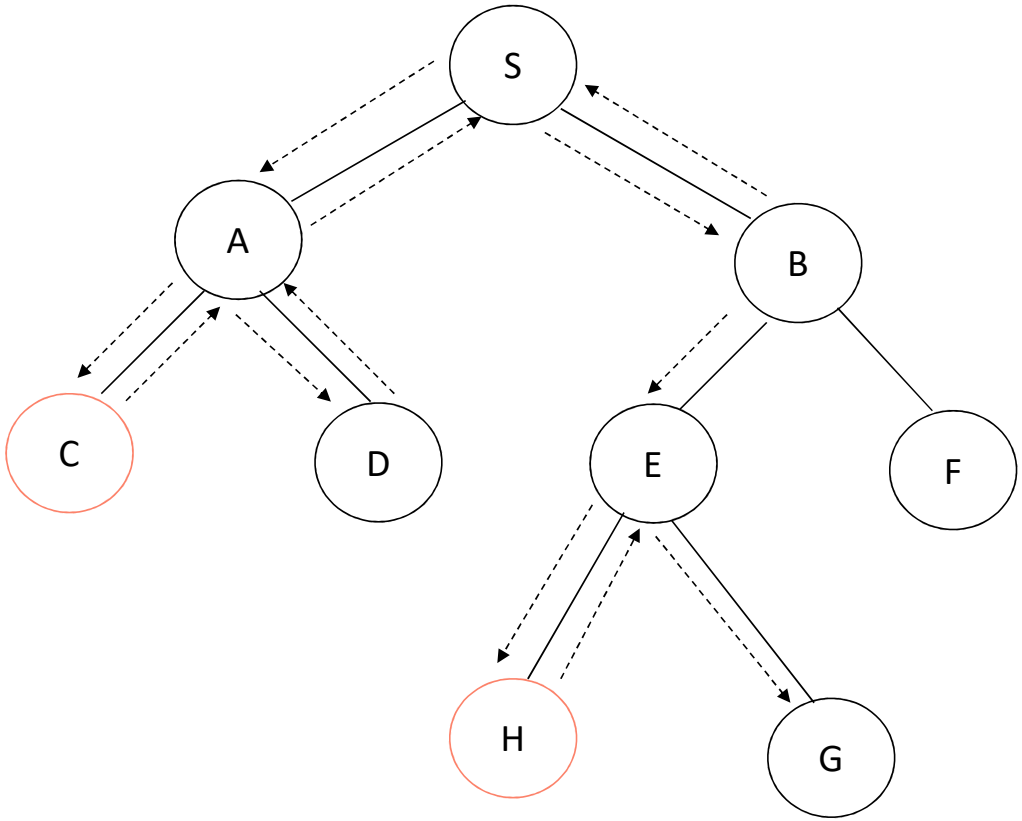
- Pencarian dilakukan pada satu node dalam setiap level dari yang paling kiri. Jika pada level yang paling dalam, solusi belum ditemukan, maka pencarian dilanjutkan pada node sebelah kanan. Demikian seterusnya sampai ditemukan solusi atau jika menemui jalan buntu akan dilacak kebelakang (*Backtracking*)

- Mirip algoritma BFS, bedanya anak diletakkan di awal antrian (*queue), bukan di akhir.*

- Antrian dapat diganti dengan *stack. Node-node di* dalam *stack di-**pop** **dan node-node baru di-push.***

- Strategi ini selalu meng*expand node di level paling* dalam pada pohon pencarian

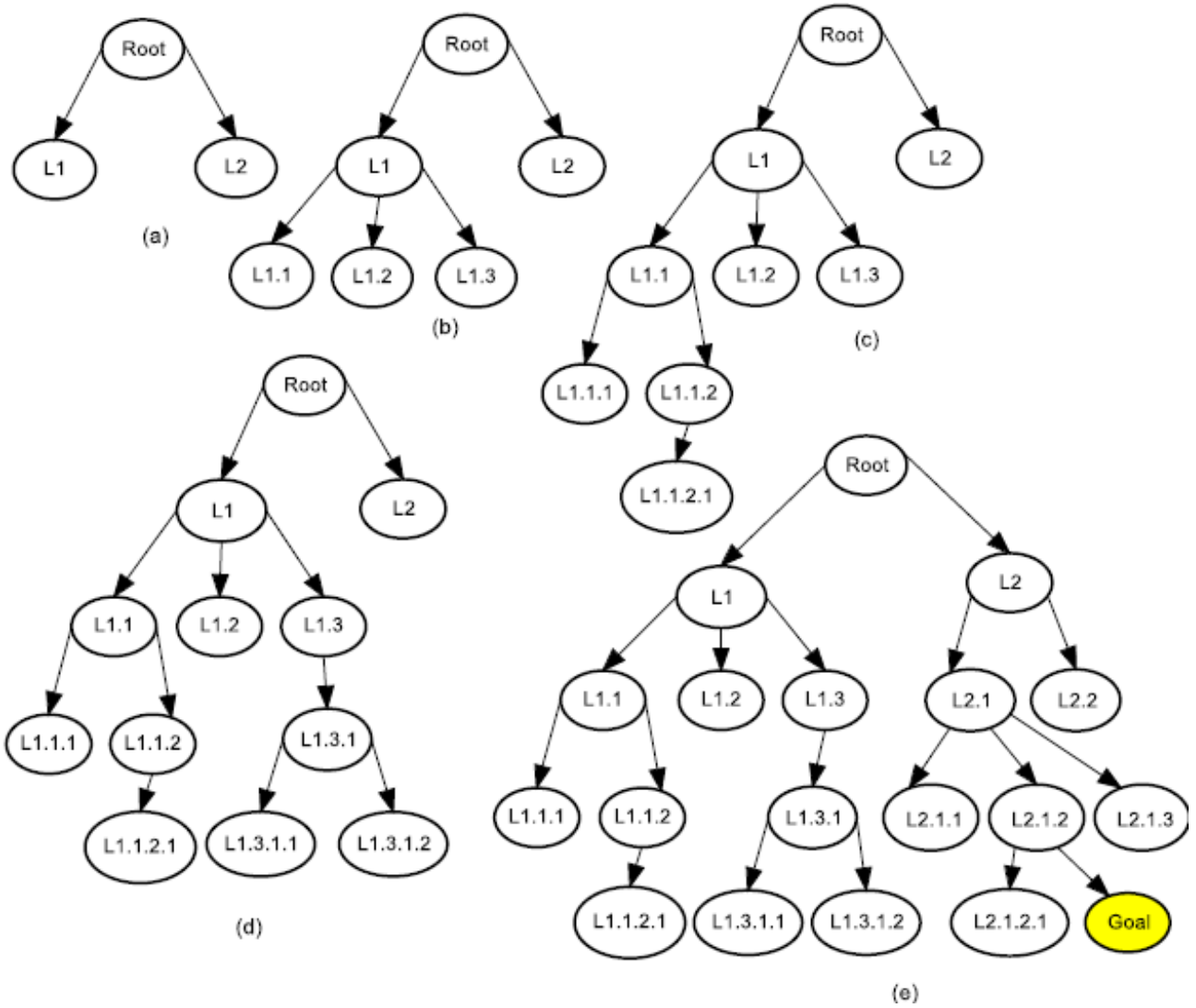
- Jalur akan di*expand terus sampai ditemukan tujuan* atau tidak ada lagi jalur yang dapat di*expand*



Pohon Depth First Search

- State awal (S)
- State akhir (H)

- BFS : S-A-B-C-D-E-F-H
- DFS :S-A-C-D-B-E-H


Root
L1
L2
(a)
Root
L1
L2
L1.1
L1.2
L1.3
(b)
Root
L1
L2
L1.1
L1.2
L1.3
L1.1.1
L1.1.2
L1.1.2.1
(c)
Root
L1
L2
L1.1
L1.2
L1.3
L1.1.1
L1.1.2
L1.3.1
L1.1.2.1
L1.3.1.1
L1.3.1.2
(d)
Root
L1
L2
L1.1
L1.2
L1.3
L2.1
L2.2
L1.1.1
L1.1.2
L1.3.1
L2.1.1
L2.1.2
L2.1.3
L1.1.2.1
L1.3.1.1
L1.3.1.2
L2.1.2.1
Goal
(e)

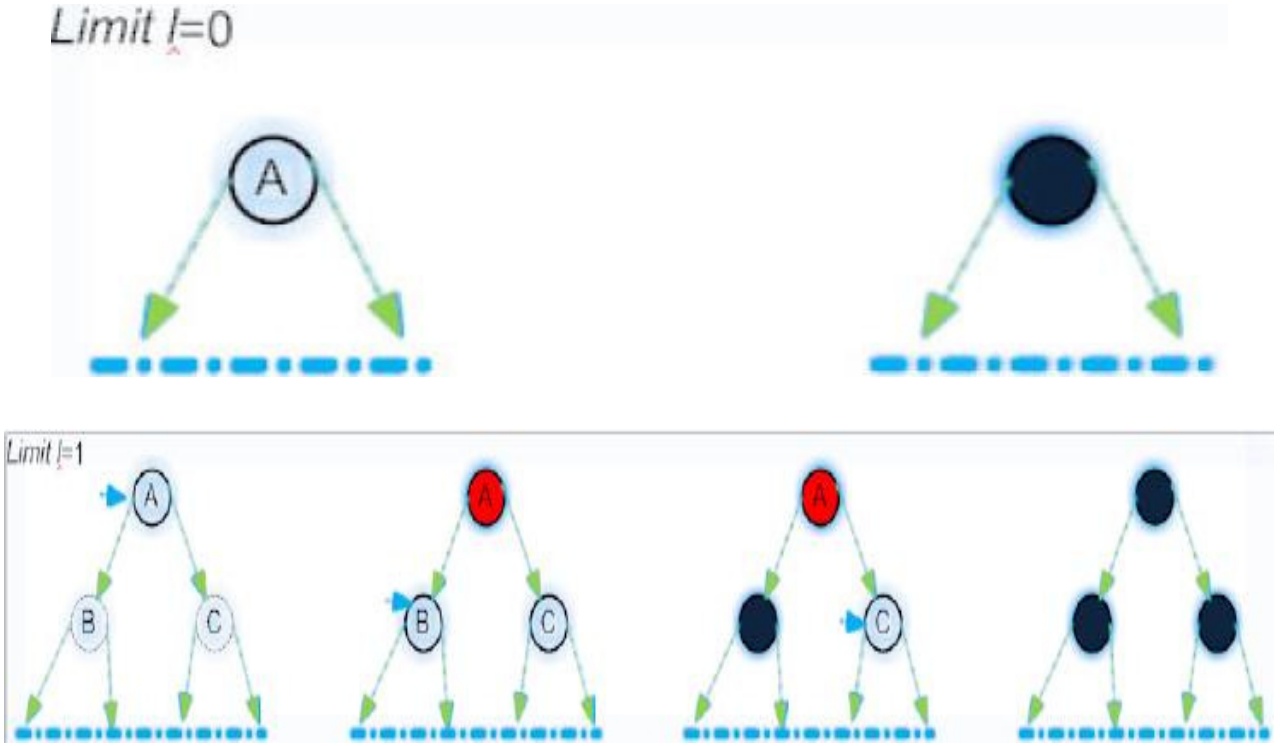
- Keuntungan DFS
  - Memory relatif kecil (menyimpan node pada lintasan aktif)
  - Mungkin menemukan solusi tanpa harus menguji lebih banyak ruang keadaan

- Kelemahan DFS
  - Memungkinkan tidak ditemukan tujuan yang diharapkan.
  - Hanya mendapatkan satu solusi pada setiap pencarian.

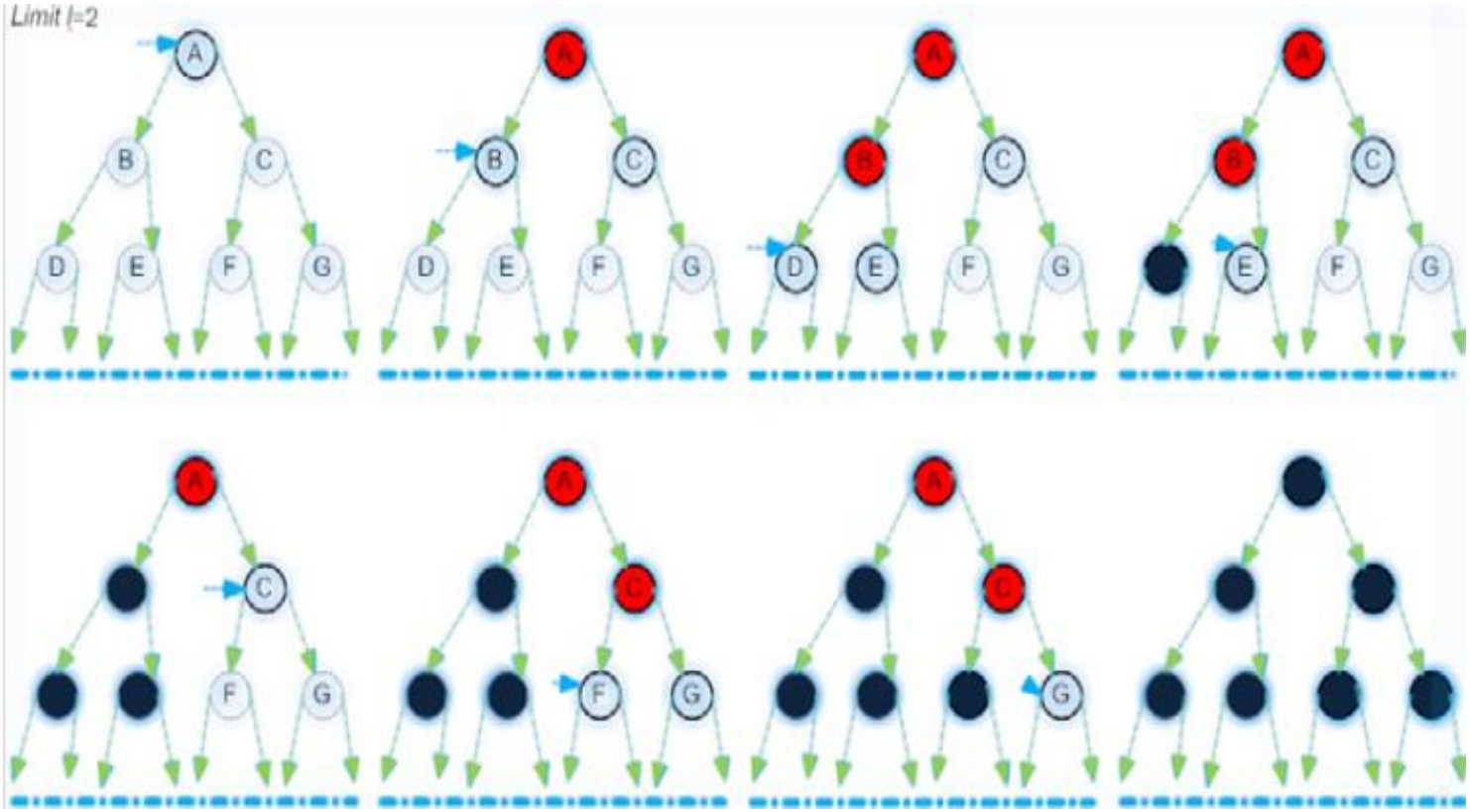
3. DFS diperdalam secara Iteratif *iterative deepening depth-first)*

- Pencarian *iterative deepening depth bounded DFS* atau *iterative deepening.*
- Pencarian sampai level kedalaman terbatas yang dinaikkan secara bertahap sampai ditemukan tujuan.
- Pertama dicoba batas kedalaman l=0, jika tidak ditemukan solusi, dinaikkan l=1, kemudian l=2 dan seterusnya
- Berbasis pada pencarian *depth bounded/limited.* Pencarian *depth bounded yang diulang sampai* solusi ditemukan.

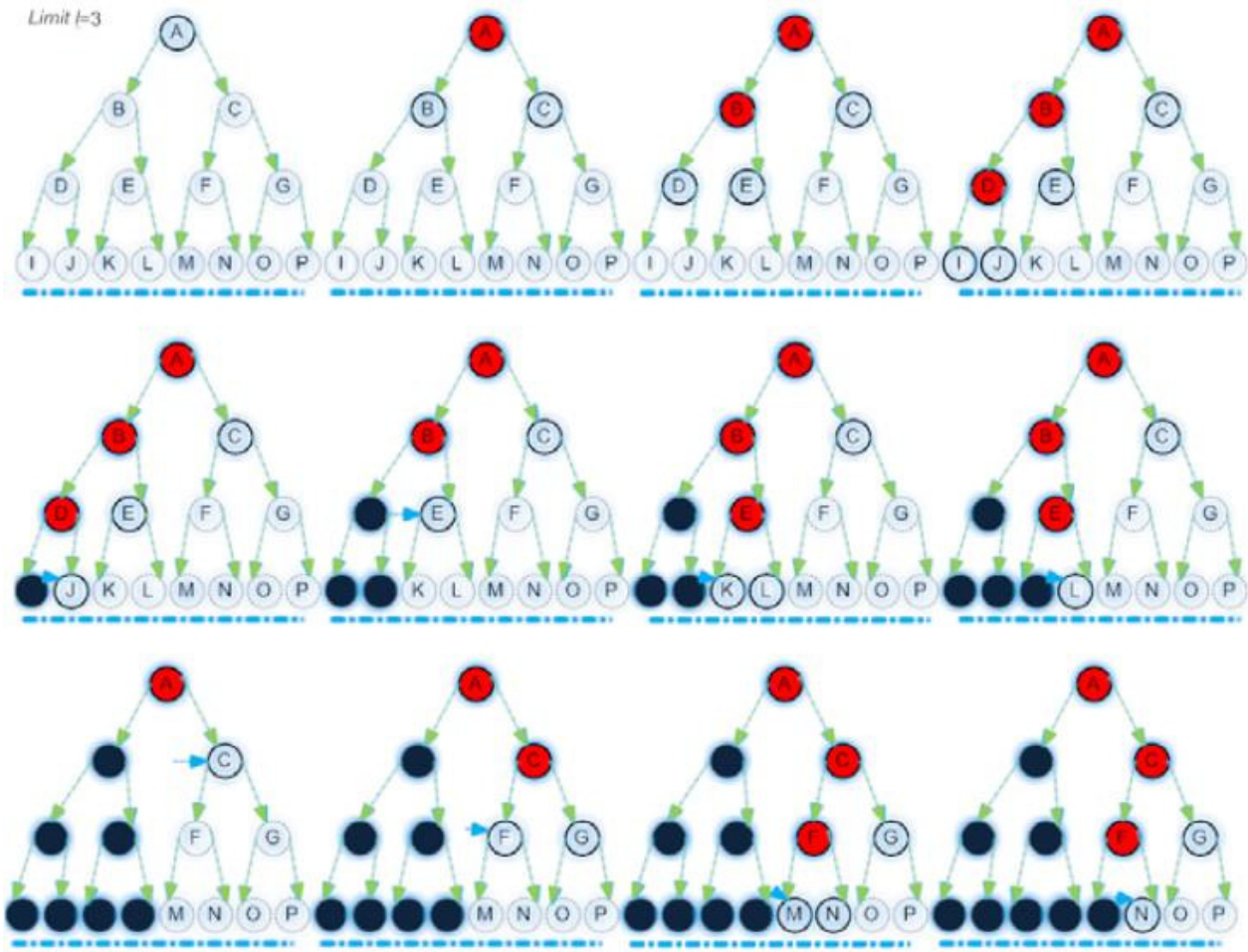
Contoh Pencarian DFS Iteratif



Contoh Pencarian DFS Iteratif



Contoh Pencarian DFS Iteratif



Pencarian *Branch and Bound* (Biaya Seragam)

- Node dengan biaya minimum selalu di*expand.*
- Begitu suatu jalur ke tujuan ditemukan, kemungkinan besar itulah jalur optimal.
- Ini digaransi dengan melanjutkan pembangkitan jalur-jalur parsial sampai semua jalur tersebut mempunyai biaya yang lebih atau sama dengan jalur yang ditemukan ke tujuan.

Algoritma Pencarian *Branch and Bound*

Kembalikan suatu solusi atau gagal

Q adalah antrian prioritas, diurutkan berdasarkan biaya terkini dari

awal (*start) ke tujuan (goal)*

Langkah 1. Tambahkan status awal (*root) ke Q.*

Langkah 2. Sampai tujuan dicapai atau Q kosong

do

Langkah 2.1 Hapus jalur (*path) pertama dari antrian Q;*

Langkah 2.2 Buat jalur baru dengan memperluas jalur pertama ke semua tetangga dari node terminal.

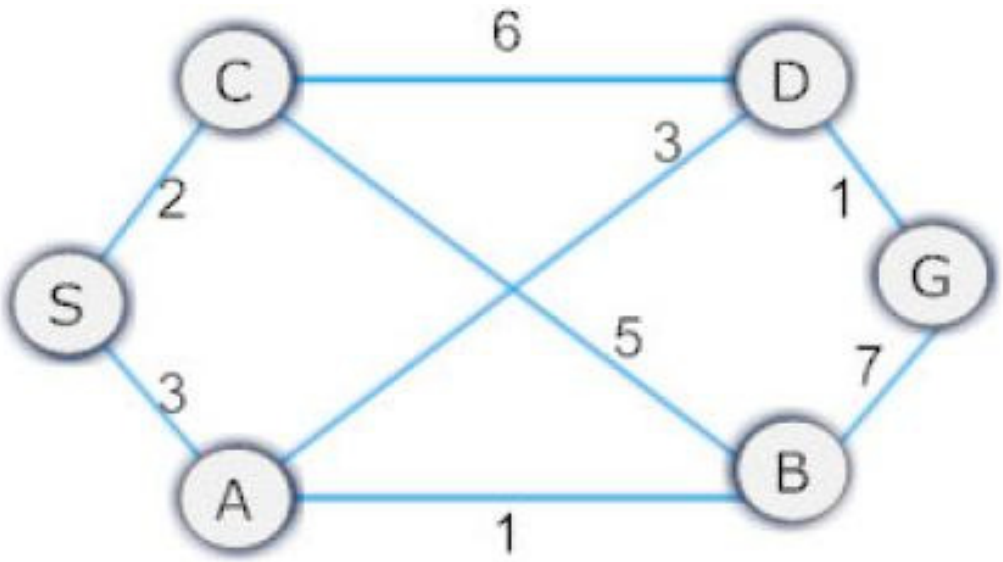
Langkah 2.3 Hapus semua jalur yang ber-*loop.*

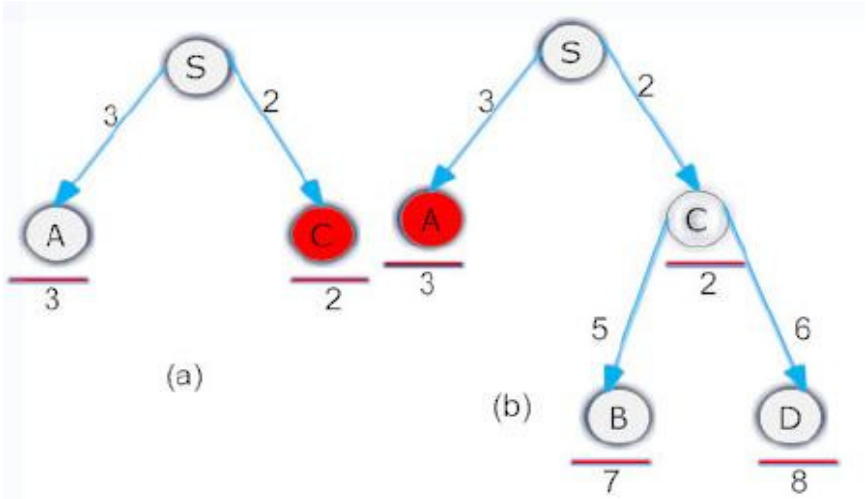
Langkah 2.4 Tambahkan jalur baru yang tersisa, jika ada, ke Q.

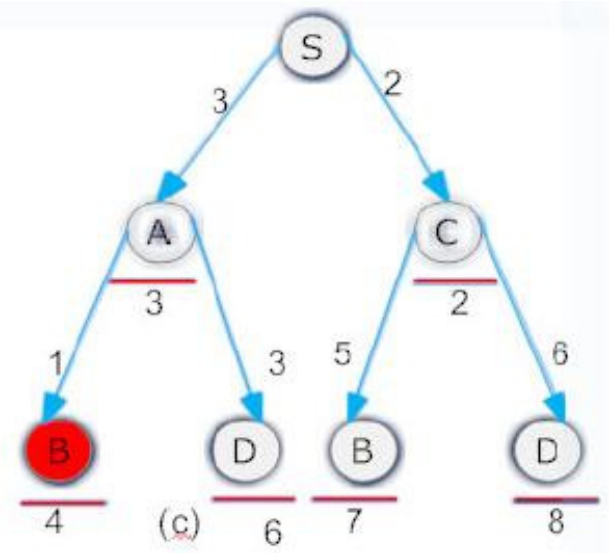
Langkah 2.5 Urutkan Q, jalur berbiaya murah ada di depan.

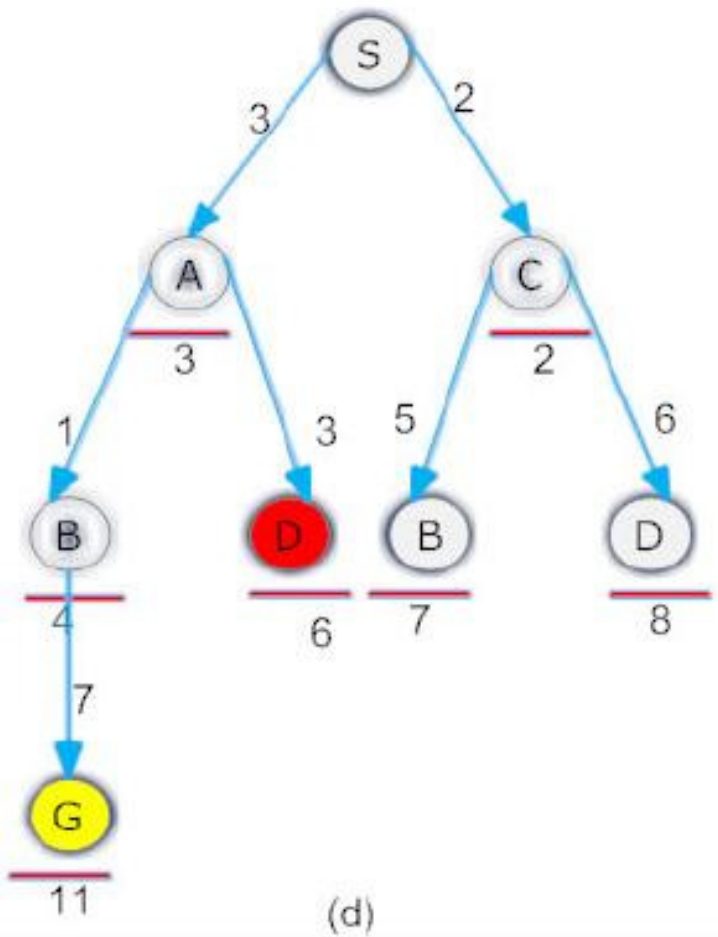
End

Contoh: Mana jalur terbaik dari S menuju G?

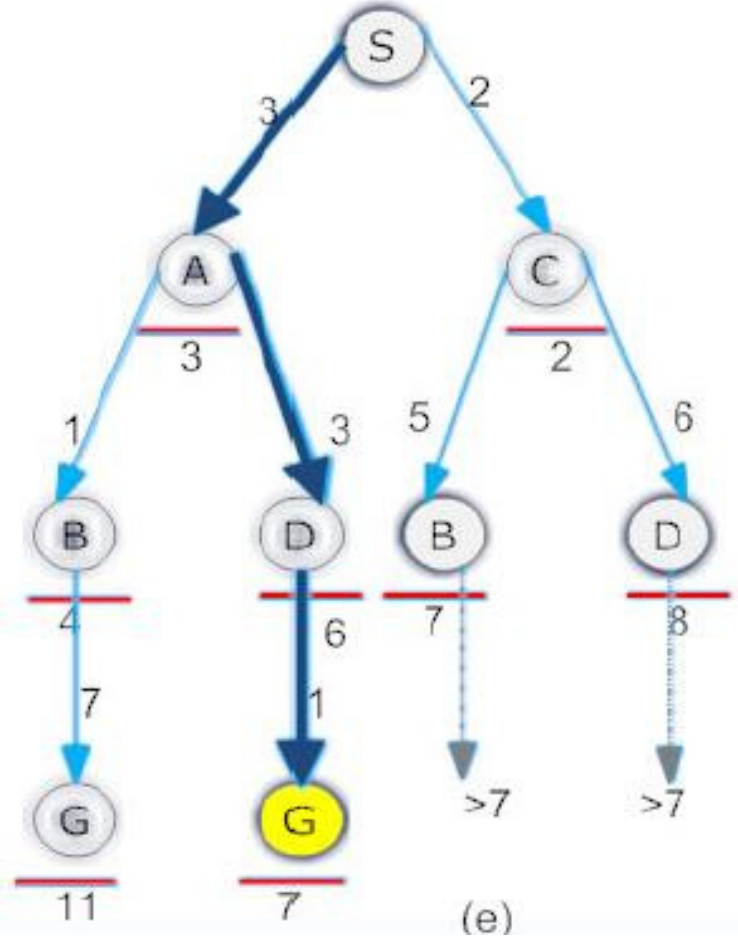



S
3
2
A
C
3
2
(a)
S
3
2
A
C
3
2
5
6
(b)
B
D
7
8


S
3
2
A
C
3
2
1
3
5
6
B
D
B
D
4
(c)
6
7
8


S
3
2
A
C
3
2
1
3
5
6
B
D
B
D
4
6
7
8
7
G
11

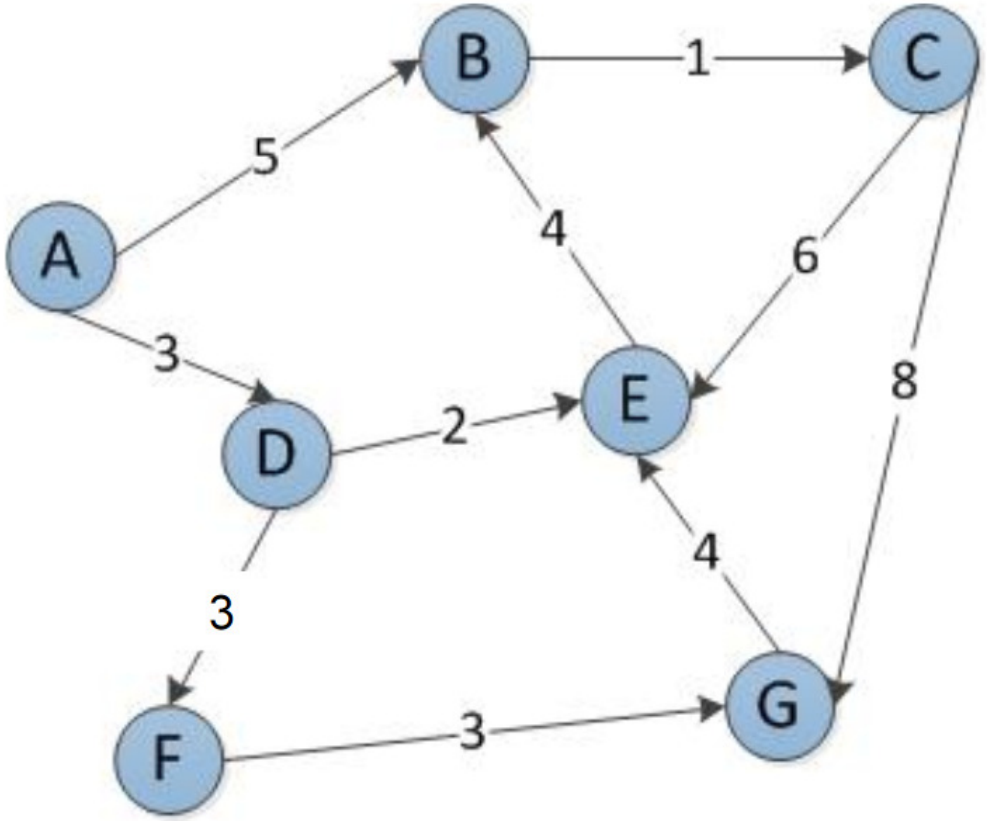
(d)

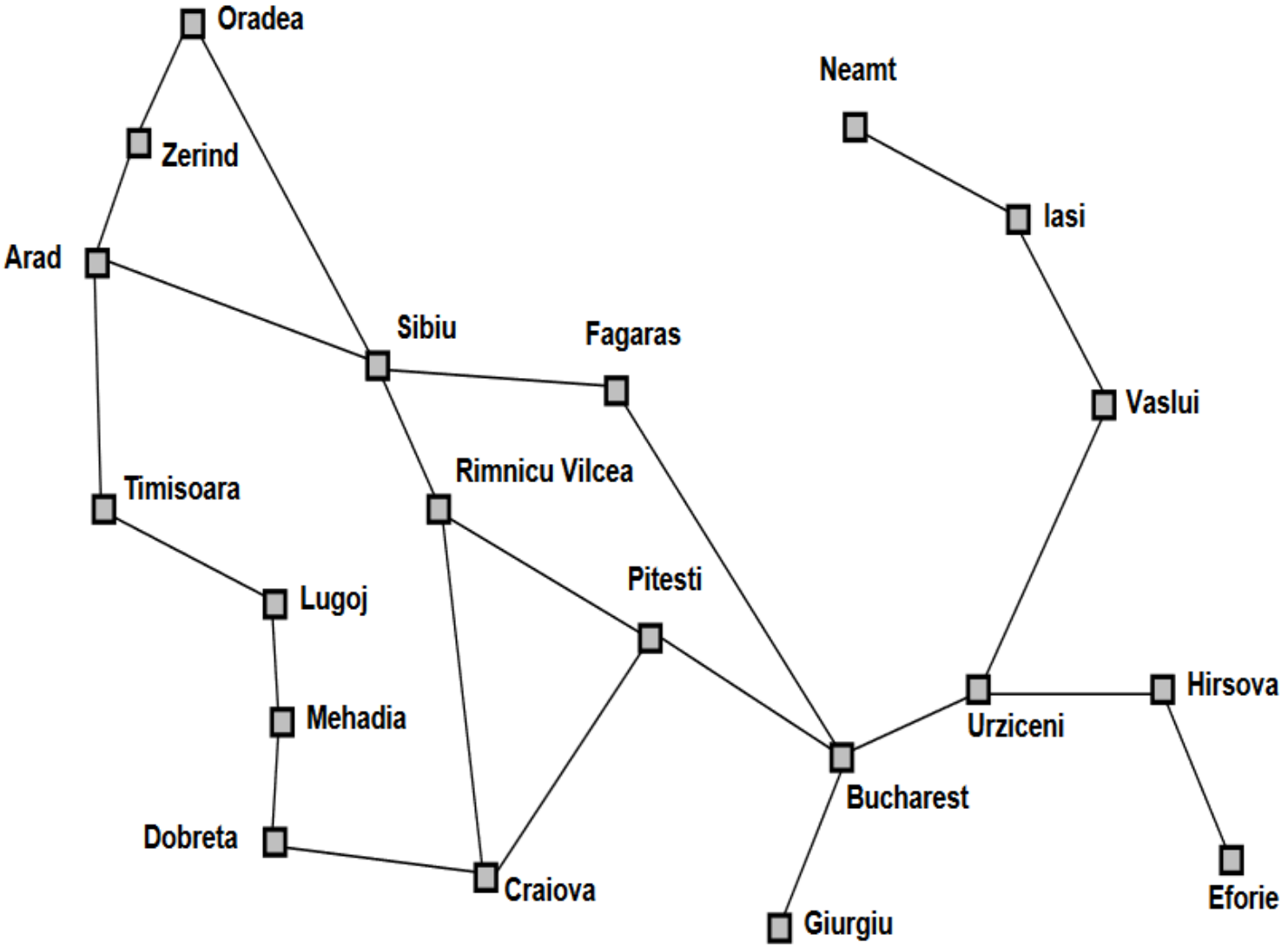

S
3
2
A
C
3
2
1
3
5
6
B
D
B
D
4
6
7
8
7
1
G
G
>7
>7
11
7

(e)

Latihan (hilangkan tanda panahnya)

- A → G
  - DFS
  - BFS
  - B&B




Oradea
Neamt
Zerind
Iasi
Arad
Sibiu
Fagaras
Vaslui
Rimnicu Vilcea
Timisoara
Pitesti
Lugoj
Hirsova
Mehadia
Urziceni
Bucharest
Dobreta
Craiova
Eforie
Giurgiu

Tugas 1

1. Ubah representasi GRAF rute Rumania menjadi representasi TREE
2. Gunakan pohon (*tree)* untuk mencari rute dari Arad menuju Bucharest dengan menggunakan metode pencarian BFS, DFS, dan DFS dengan kedalaman iteratif. Perlihatkan hasil akhir masing-masing metode pencarian, lakukan analisis terhadap hasilnya.
3. [CHALLENGE-20] Buatlah program untuk mengimplementasikan pencarian rute kota Arad ke kota-kota lain di Rumania tersebut dengan menggunakan BFS atau DFS (pilih salah satu)
   - Bahasa pemrograman BEBAS
   - Input = nama kota asal dan nama kota tujuan
   - Output = Rute kota-kota yang dilewati dari asal sampai tujuan sesuai metode

Tugas 1

- Kerjakan CHALLENGE untuk mendapatkan nilai bonus, berguna untuk nilai akhir (membantu mengatrol nilai tanpa dosen harus manipulasi).

- Kerjakan dan submit melalui Classroom dalam bentuk berkas DOCX
- Deadline cek langsung di Classroom
- Beri nama berkas dengan format : **TUGAS1-NIM.docx** (misal : **TUGAS1-123180001.docx**)

- SEGALA BENTUK PLAGIASI AKAN DIBERI NILAI 0 (NOL)