PERTEMUAN 11

Struktur Data & Algoritma

BREADTH FIRST SEARCH (**BFS**) &
DEPTH FIRST SEARCH (**DFS**)

Menjelajah Graf

Di dalam Graf, istilah **traversal** berarti mengunjungi setiap **vertex** dalam graf, untuk mennagani kepetingan seperti itu terdapat dua cara yang umum dipakai, yaitu :

1. *Breadth First Search* (BFS)
2. *Depth First Search* (DFS)

Pengertian BFS dan DFS

1) *Breadth First Search (BFS)* merupakan cara mengunjungi data dalam graph dengan menelusuri setiap simpul yang sederajat terlebih dahulu. Implementasi BFS menggunakan teknik *Queue* (antrean) yang bersifat FIFO.

2) *Depth First Search (DFS)* merupakan cara mengunjungi data dalam graph dengan menelusuri simpul terdalam lebih dahulu. Implementasi DFS menggunakan teknik *Stack* (tumpukan) yang bersifat LIFO.

Algoritma BFS

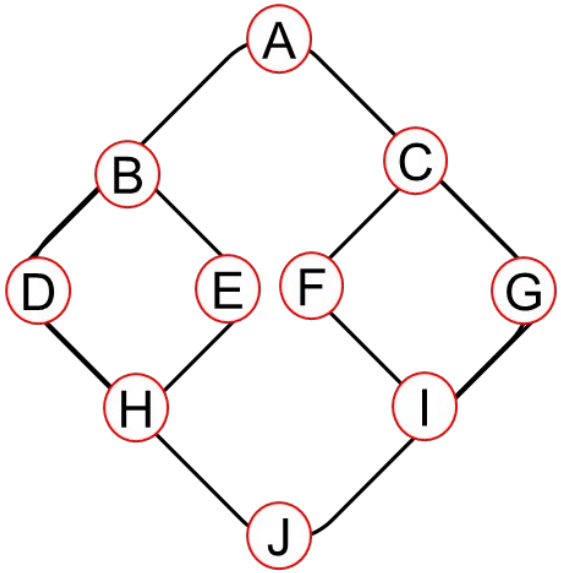
Proses kerja teknik Breadth First Search (BFS):

1) *Antrean harus dalam keadaan kosong.*
2) *Masukan simpul awal ke dalam antrean dan tandai simpul tersebut dengan tanda/ simbol "telah dikunjungi".*
3) *Kosongkan hasil BFS.*
4) *Apabila antrean ada isinya:*
5) *Keluarkan sebuah nilai dari antrean dan tambahkan ke hasil BFS.*
6) *Masukan semua simpul tetangganya yang belum dikunjungi ke dalam antrean.*
7) *Berikan tanda "telah dikunjungi" terhadap setiap simpul yang telah masuk ke dalam antrean.*

.

Algoritma BFS

Contoh: proses kerja teknik BFS untuk mengunjungi setiap simpul dalam graph.



Proses 1: Antrean kosong.

X	X	X	X	X	X	X	X	X	X
A	B	C	D	E	F	G	H	I	J

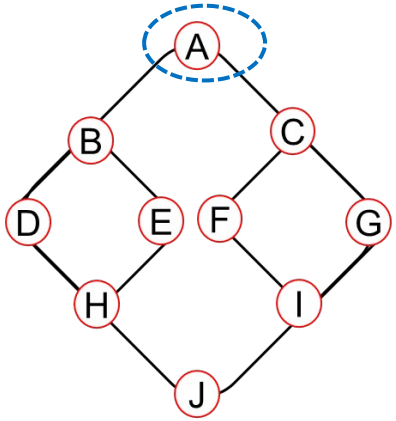
Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Antrean:

Front Rear

Hasil Proses:

Algoritma BFS

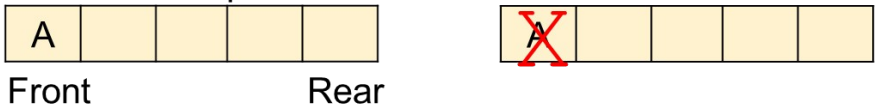


Proses 2: Masukkan simpul awal ke dalam antrean.

√	X	X	X	X	X	X	X	X	X
A	B	C	D	E	F	G	H	I	J

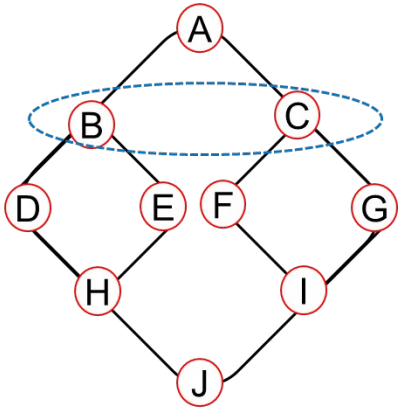
Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Antrean: proses Simpul A ke dalam hasil proses, lalu Simpul A dihapus.



Hasil Proses:

Algoritma BFS



Proses 3: Kunjungi tetangga Simpul A, yaitu Simpul B dan C.

√	√	√	X	X	X	X	X	X	X
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**

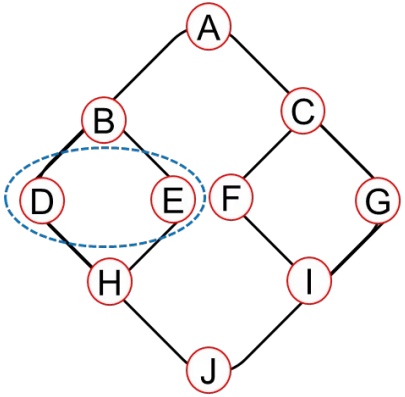
Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Antrean: proses Simpul B lebih dahulu, lalu Simpul B dihapus.

Hasil Proses:

A	**B**								

Proses 4: Kunjungi tetangga Simpul B, yaitu Simpul D dan E.



√	√	√	√	√	X	X	X	X	X
A	B	C	D	E	F	G	H	I	J

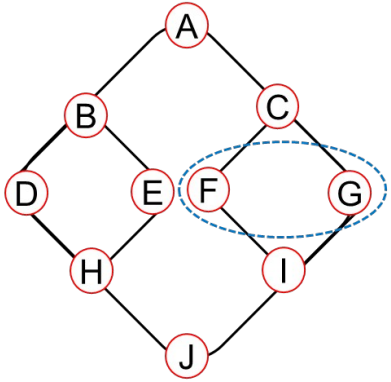
Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Antrean: proses Simpul C lebih dahulu, lalu Simpul C dihapus.

Hasil Proses:

A	B	C							

Proses 5: Kunjungi tetangga Simpul C, yaitu Simpul F dan G.



√	√	√	√	√	√	√	X	X	X
A	B	C	D	E	F	G	H	I	J

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Antrean: proses Simpul D lebih dahulu, lalu Simpul D dihapus.

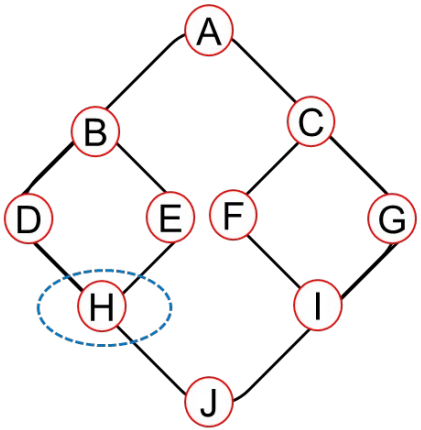
D	E	F	G		→	D	E	F	G	

Front Rear

Hasil Proses:

A	B	C	D						

Proses 6: Kunjungi tetangga Simpul D, yaitu Simpul H.



√	√	√	√	√	√	√	√	X	X
A	B	C	D	E	F	G	H	I	J

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Antrean: proses Simpul E lebih dahulu, lalu Simpul E dihapus.

E	F	G	H	

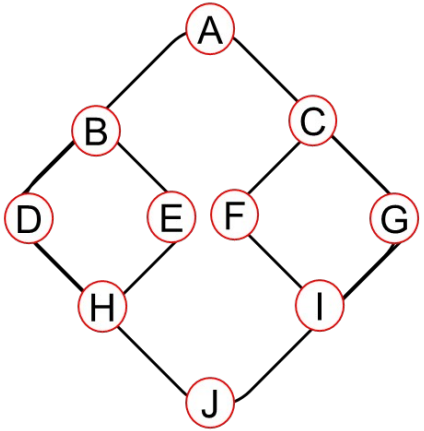
Front Rear

E	F	G	H	

Hasil Proses:

A	B	C	D	E					

Algoritma BFS



Proses 7: Kunjungi tetangga Simpul E yang belum dikunjungi, Simpul H sudah dikunjungi, jadi Simpul E tidak ada tetangga.

√	√	√	√	√	√	√	√	X	X
A	B	C	D	E	F	G	H	I	J

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Antrean: proses Simpul F lebih dahulu, lalu Simpul F dihapus.

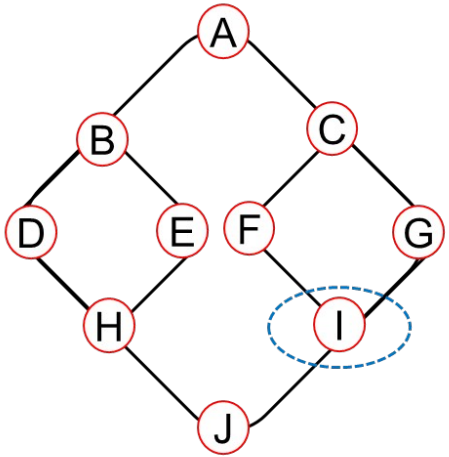
F	G	H		

Front Rear

~~F~~	G	H		

Hasil Proses:

A	B	C	D	E	F				



Proses 8: Kunjungi tetangga Simpul F yang belum dikunjungi, yaitu Simpul I.

√	√	√	√	√	√	√	√	√	X
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Antrean: proses Simpul G lebih dahulu, lalu Simpul G dihapus.

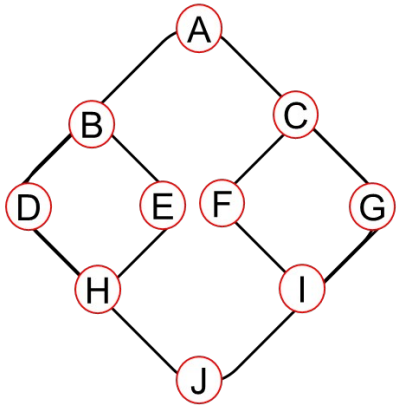
G	H	I		

Front Rear

X

Hasil Proses:

A	**B**	**C**	**D**	**E**	**F**	**G**			



Proses 9: Kunjungi tetangga Simpul G yang belum dikunjungi, Simpul I sudah dikunjungi, jadi tetangga smpul G tidak ada.

√	√	√	√	√	√	√	√	√	X
A	B	C	D	E	F	G	H	I	J

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

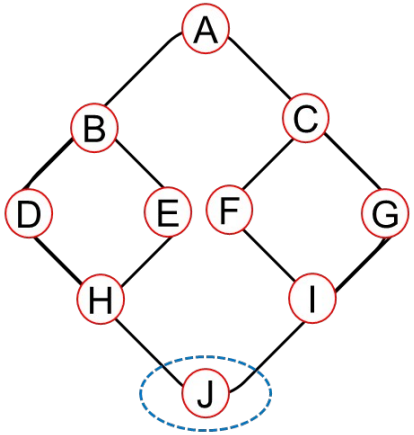
Antrean: proses Simpul H lebih dahulu, lalu Simpul H dihapus.

H	I			

Front Rear

Hasil Proses:

A	B	C	D	E	F	G	H		



Proses 10: Kunjungi tetangga Simpul H yang belum dikunjungi, yaitu Simpul J.

√	√	√	√	√	√	√	√	√	√
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Antrean: proses Simpul I lebih dahulu, lalu Simpul I dihapus.

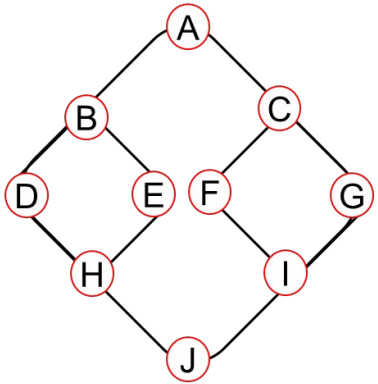
I	J			

Front Rear

X	J			

Hasil Proses:

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	



Proses 11: Kunjungi tetangga Simpul I yang belum dikunjungi, Simpul J sudah dikunjungi, jadi tetangga Simpul I tidak ada.

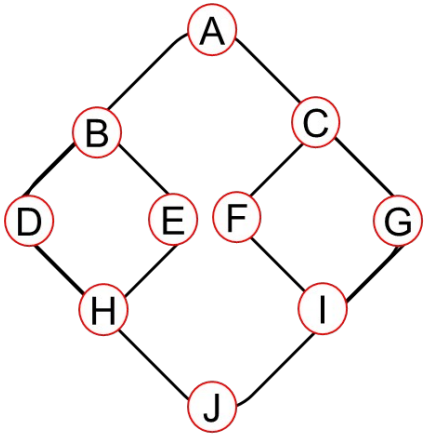
√	√	√	√	√	√	√	√	√	√
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Antrean: proses Simpul J, lalu Simpul J dihapus.

Hasil Proses:

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**



Semua Simpul telah dikunjungi dan semua antrean telah diproses, maka proses selesai.

√	√	√	√	√	√	√	√	√	√
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Antrean: kosong

Front Rear

Hasil Proses:

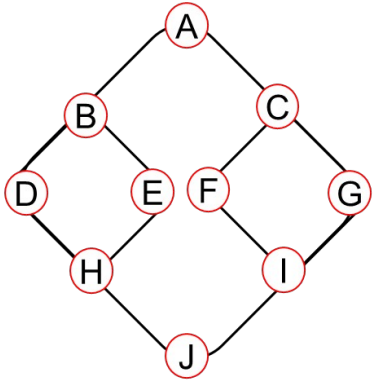
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**

Algoritma DFS

Proses kerja teknik *Depth First Search (DFS)*:

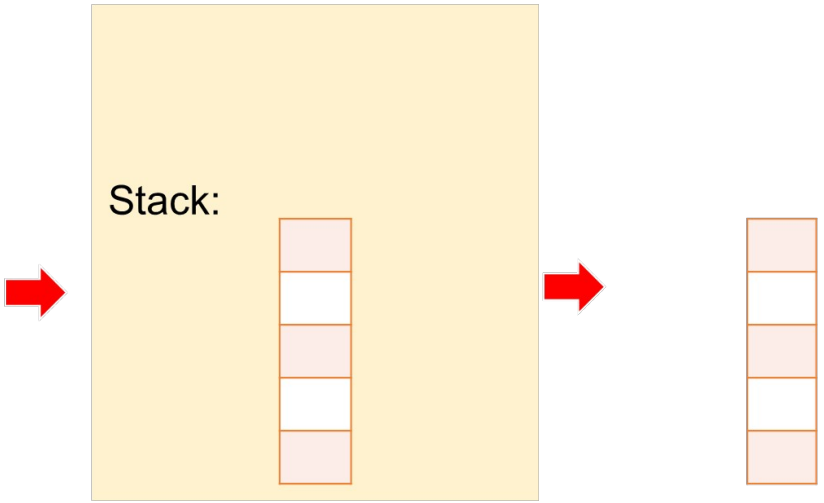
1) Tumpukan harus dalam keadaan kosong.
2) Masukan simpul awal ke dalam tumpukkan dan tandai simpul tersebut dengan tanda/ simbol “telah dikunjungi”.
3) Kosongkan hasil DFS.
4) Apabila tumpukan ada isinya:
   a. Keluarkan *(Pop)* sebuah nilai dari tumpukan dan tambahkan ke hasil DFS.
   b. Masukkan semua simpul tetangganya yang belum dikunjungi ke dalam tumpukan.
   c. Berikan tanda “telah dikunjungi” terhadap setiap simpul yang telah masuk ke dalam tumpukan.

Contoh: proses kerja teknik DFS untuk mengunjungi setiap simpul dalam graph.

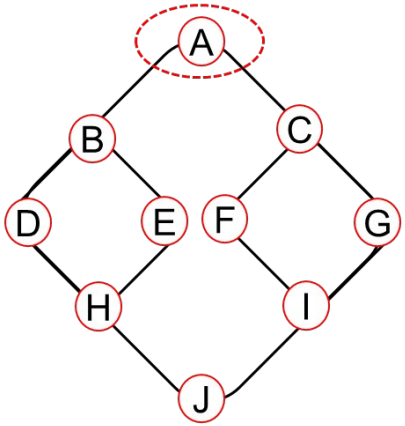


A	X
B	X
C	X
D	X
E	X
F	X
G	X
H	X
I	X
J	X

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

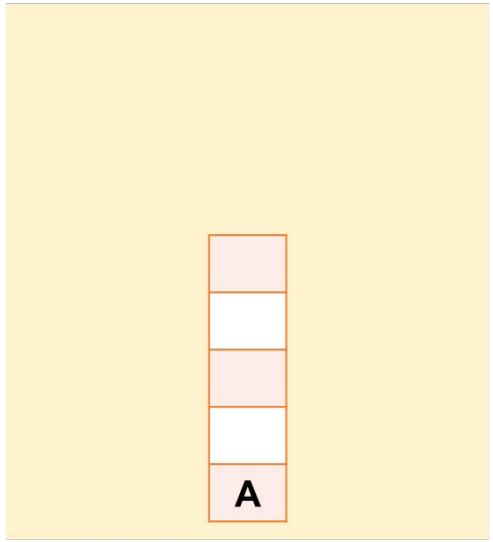


Hasil Proses:



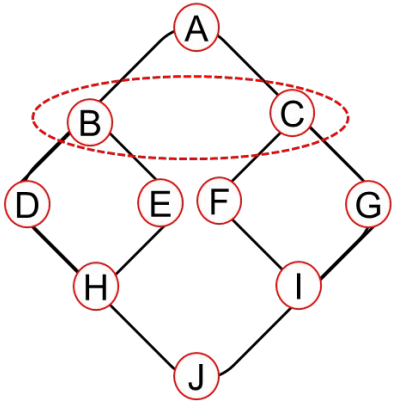
A	√
B	X
C	X
D	X
E	X
F	X
G	X
H	X
I	X
J	X

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi



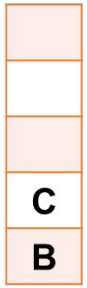
Simpul A telah diproses, lalu dikeluarkan.

Hasil Proses:

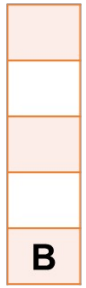


A	√
B	√
C	√
D	X
E	X
F	X
G	X
H	X
I	X
J	X

Proses 2:
Kunjungi tetangga Simpul A, yaitu Simpul B dan Simpul C.

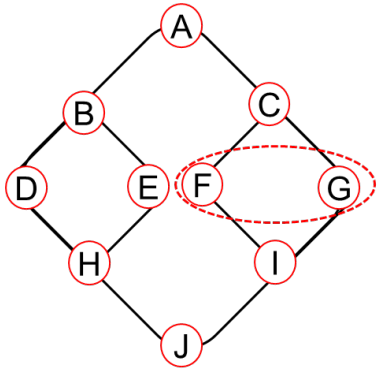


Simpul C diproses lebih dahulu, lalu dikeluarkan.



Hasil Proses:

A	C								



A	√
B	√
C	√
D	X
E	X
F	√
G	√
H	X
I	X
J	X

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Proses 3:
Kunjungi tetangga Simpul C, yaitu Simpul F dan Simpul G.

Stack:

G
F
B

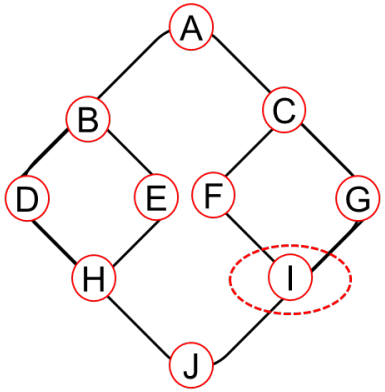
Simpul G diproses lebih dahulu, lalu dikeluarkan.

F
B

Hasil Proses:

A	C	G							

Algoritma DFS

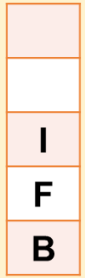


A	√
B	√
C	√
D	X
E	X
F	√
G	√
H	X
I	√
J	X

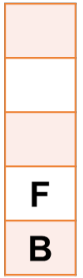
Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Proses 4:
Kunjungi tetangga Simpul G, yaitu Simpul I.

Stack:



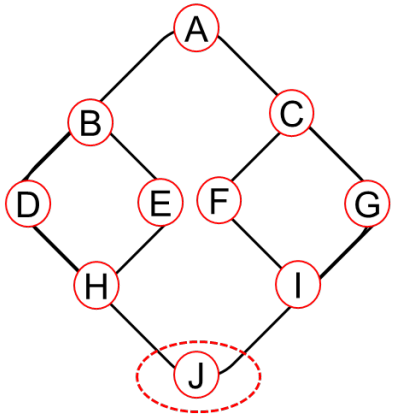
Simpul I diproses lebih dahulu, lalu dikeluarkan.



Hasil Proses:

A	C	G	I						

Algoritma DFS



A	√
B	√
C	√
D	X
E	X
F	√
G	√
H	X
I	√
J	√

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Proses 5:
Kunjungi tetangga Simpul I, yaitu Simpul J.

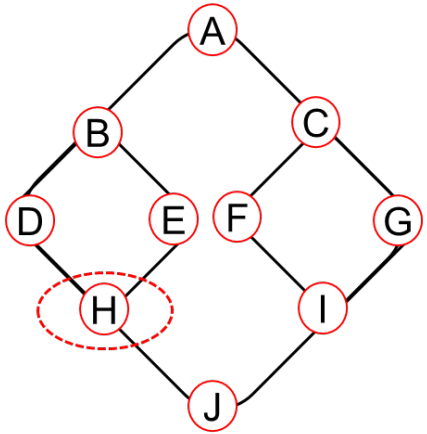
Stack:

Simpul J diproses lebih dahulu, lalu dikeluarkan.

Hasil Proses:

A	C	G	I	J					

Algoritma DFS



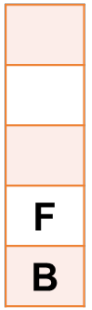
A	√
B	√
C	√
D	X
E	X
F	√
G	√
H	√
I	√
J	√

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Proses 6:
Kunjungi tetangga Simpul J, yaitu Simpul H.

Stack:

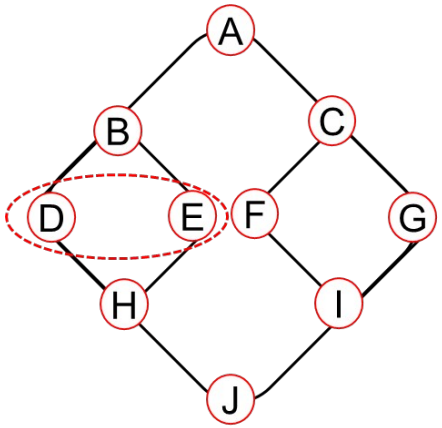
Simpul H diproses lebih dahulu, lalu dikeluarkan.



Hasil Proses:

A	C	G	I	J	H				

Algoritma DFS



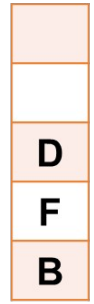
A	√
B	√
C	√
D	√
E	√
F	√
G	√
H	√
I	√
J	√

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Proses 7:
Kunjungi tetangga Simpul H, yaitu Simpul D dan Simpul E.

Stack:

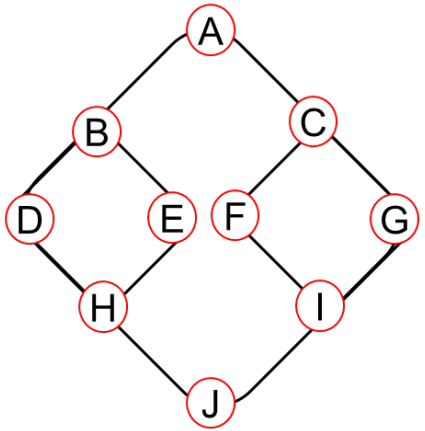
Simpul E diproses lebih dahulu, lalu dikeluarkan.



Hasil Proses:

A	C	G	I	J	H	E			

Algoritma DFS



A	√
B	√
C	√
D	√
E	√
F	√
G	√
H	√
I	√
J	√

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Proses 8:
Kunjungi tetangga Simpul E yang belum dikunjungi, tidak ada.

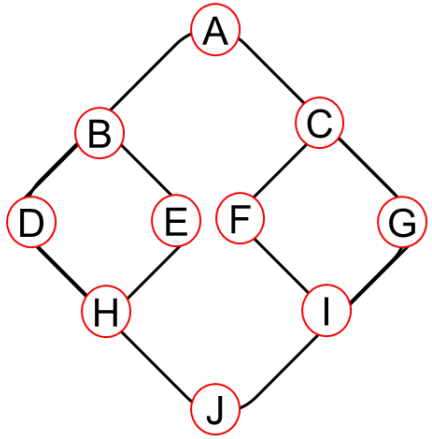
Stack:

Simpul D diproses lebih dahulu, lalu dikeluarkan.

Hasil Proses:

A	C	G	I	J	H	E	D		

Algoritma DFS



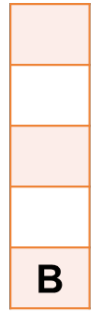
A	√
B	√
C	√
D	√
E	√
F	√
G	√
H	√
I	√
J	√

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Proses 9:
Kunjungi tetangga Simpul D yang belum dikunjungi, tidak ada.

Stack:

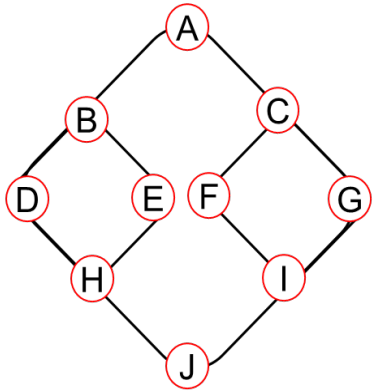
Simpul F diproses lebih dahulu, lalu dikeluarkan.



Hasil Proses:

A	C	G	I	J	H	E	D	F	

Algoritma DFS



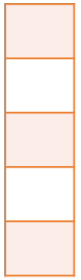
A	√
B	√
C	√
D	√
E	√
F	√
G	√
H	√
I	√
J	√

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Proses 10:
Kunjungi tetangga Simpul F yang belum dikunjungi, tidak ada.

Stack:

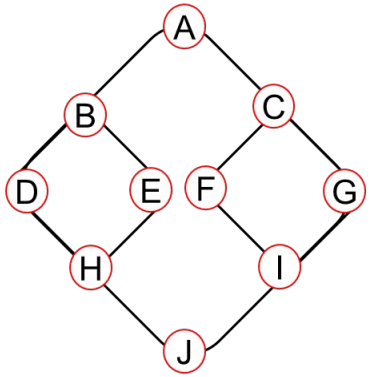
Simpul F diproses lebih dahulu, lalu dikeluarkan.



Hasil Proses:

A	C	G	I	J	H	E	D	F	B

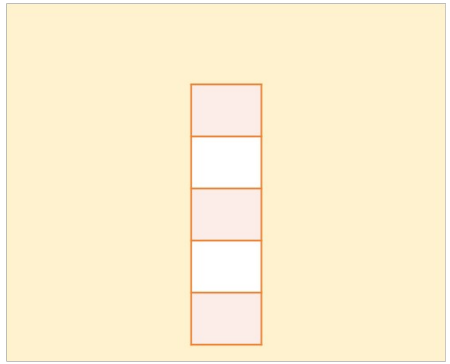
Algoritma DFS



A	√
B	√
C	√
D	√
E	√
F	√
G	√
H	√
I	√
J	√

Tanda X = belum dikunjungi
Tanda √ = sudah dikunjungi

Semua Simpul telah dikunjungi dan semua tumpukan telah diproses dan telah kosong, maka proses selesai.



Hasil Proses:

A	C	G	I	J	H	E	D	F	B

SELESAI