“본 강의 동영상 및 자료는 대한민국 저작권법을 준수합니다. 본 강의 동영상 및 자료는 상명대학교 재학생들의 수업목적으로 제작•배포되는 것이므로, 수업목적으로 내려받은 강의 동영상 및 자료는 수업목적 이외에 다른 용도로 사용할 수 없으며, 다른 장소 및 타인에게 복제, 전송하여 공유할 수 없습니다. 이를 위반해서 발생하는 모든 법적 책임은 행위 주체인 본인에게 있습니다.”

4.2 Depth-first search in undirected graphs

(0) All about search

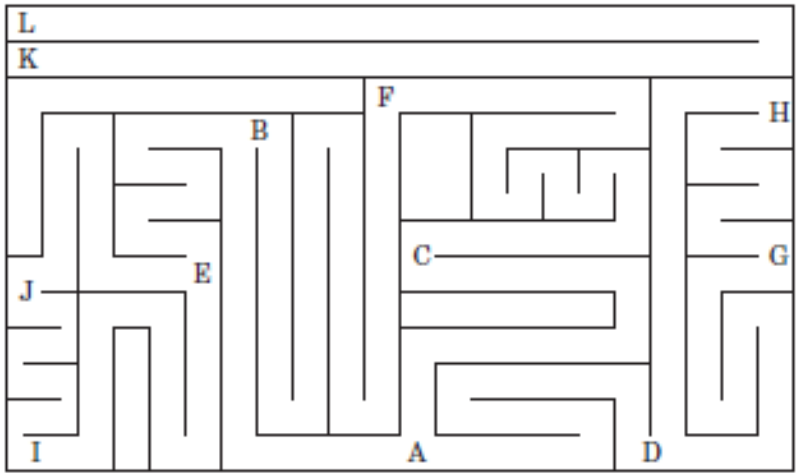
Type		Search	Start	Return	Function	Meaning	Performance
Array		Search (A, x)	A[0] A[m]	T/F or index	Linear search Binary search	Find x in an array A (start from A[0] or A[m])	O(n) O(log n)
Linked list		Search (L, x)	head	T/F or node	Linear search	Find x in a list L (start from head)	O(n)
Tree	Binary Search Tree	Search (T, x)	root	T/F or node	Binary search	Find x in a tree T (start from root)	O(log n)
	Ordinary Tree	Search (T) Traversal	root	List of reachable vertices	Inorder/Preorder/Postorder DFS/BFS	Report all vertices reachable from root node	O(n)
Graph		Search (G, v) Traversal	Any v	List of reachable vertices	DFS/BFS	Report all vertices reachable from v in a graph G (start from v)	O(n) + O(|E|)

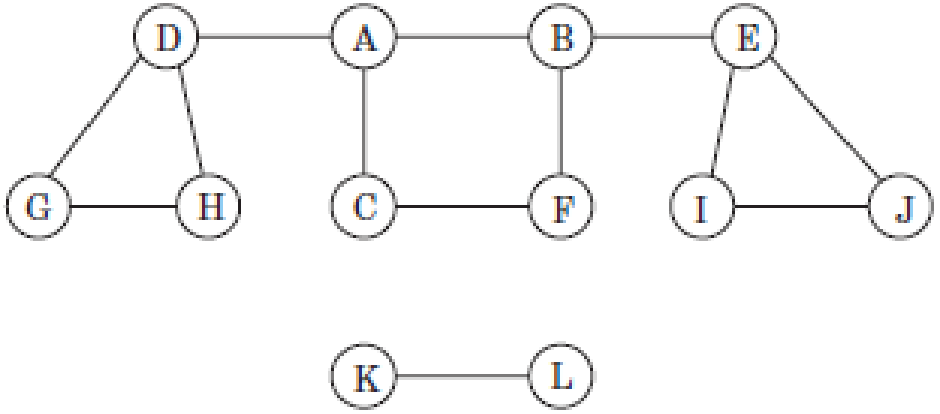
– *Non-searchable data structure: stack, queue, heap*

4.2 Depth-first search in undirected graphs

(1) Search of a graph

– *What parts of the graph are reachable from a given vertex?*





4.2 Depth-first search in undirected graphs

(1) Search of a graph

– Use a recursive call to search a graph

```
void dfs (v)
Input: v ∈ V in G = (V, E)
Output: visit(v) is checked

    visit[v] = True;
//  previsit ( v );
    for each edge (v, u) ∈ E
        if visit[u] == False
            dfs ( u );
//  postvisit ( v );
```

Adjacency matrix → O(n)

```
for ( int i = 0; i < n; i++ )
    if ( adjacency_matrix[v][i] != 0 )
       ……
```

Adjacency list → O(1) ~ O(n)

```
for ( t = v; t != NULL; t = t->next)
    ……
```

4.2 Depth-first search in undirected graphs

(2) Basic strategy of depth-first search

- Visit connected nodes as far as possible
- Use a stack
- Implemented using a recursive call

```
void dfs (G)
Input: G = (V, E)
Output: visit(v) is checked

    for all v ∈ V
        visit[v] = False;
    for all v ∈ V
        if visit[v] == False
            dfs ( v );
```

(3) Time complexity of DFS

```
void dfs (v)
    visit[v] = True;
//  previsit ( v );
    for each edge (v, u) ∈ E
        if visit[u] == False
            dfs ( u );
//  postvisit ( v );
```

dfs(v) → n

e_i: No. of edges incident to vertex i

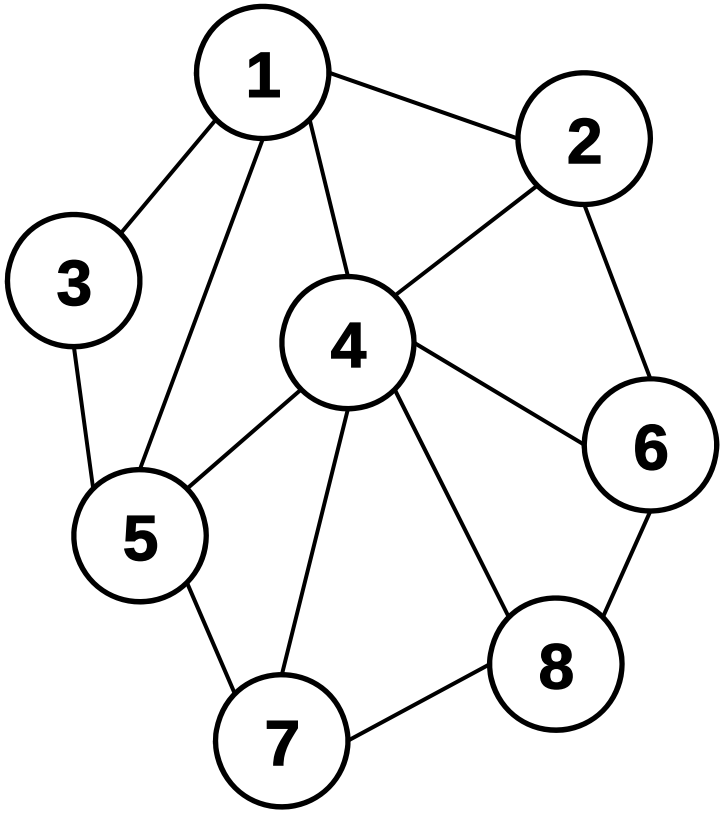
In total, $\sum_{i=1}^{n} e_i = O(E) = O(m)$

```
void dfs (G)
    for all v ∈ V
        visit[v] = False;
    for all v ∈ V
        if visit[v] == False
            dfs ( v );
```

O(n)

In total, O(n + m)

(3) Time complexity of DFS



dfs(G) = dfs(1) + … + dfs(8)

dfs(1) = 4

dfs(2) = 3

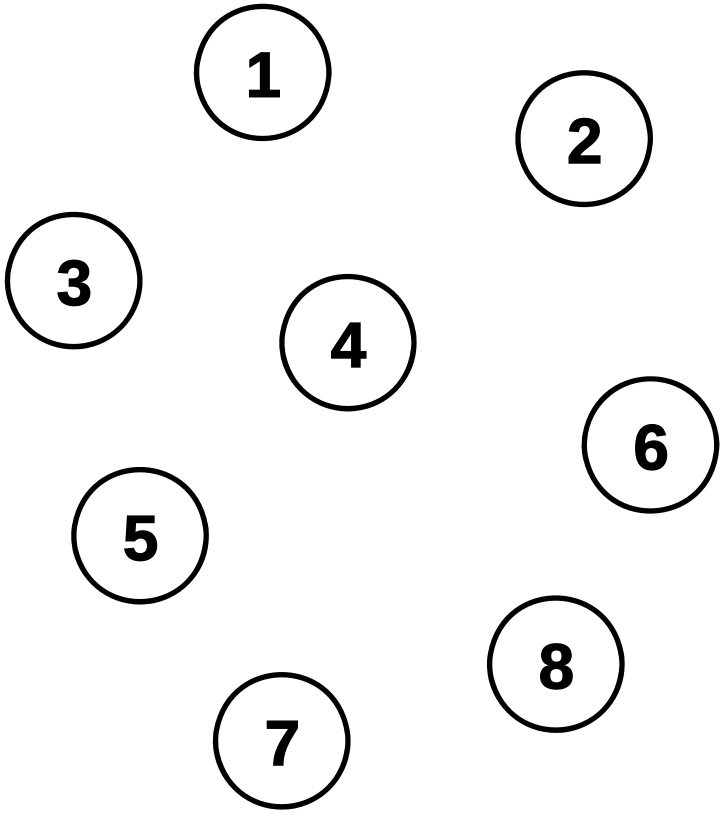
⋮

dfs(8) = 3

+ → dfs(G) = O(m)

```
void dfs (v)
    visit[v] = True;
    for each edge (v, u) ∈ E
        if visit[u] == False
            dfs ( u );
```

(3) Time complexity of DFS



dfs(G) = dfs(1) + … + dfs(8)

dfs(1) = 0

dfs(2) = 0

⋮

dfs(8) = 0

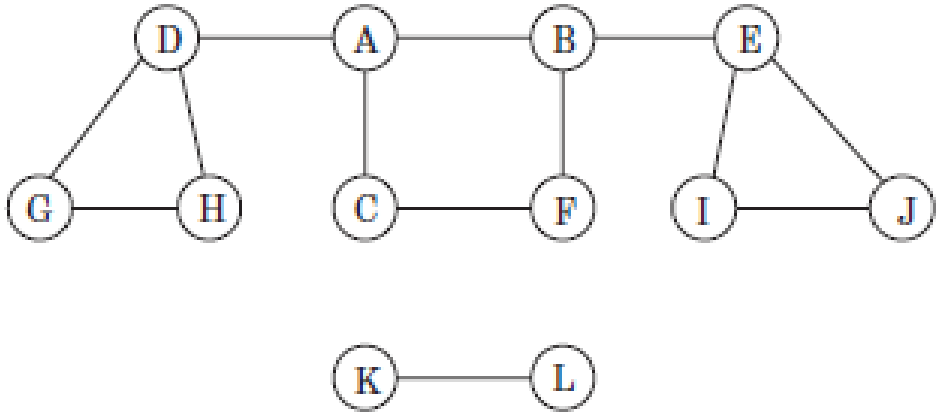
+ → dfs(G) = O(n)

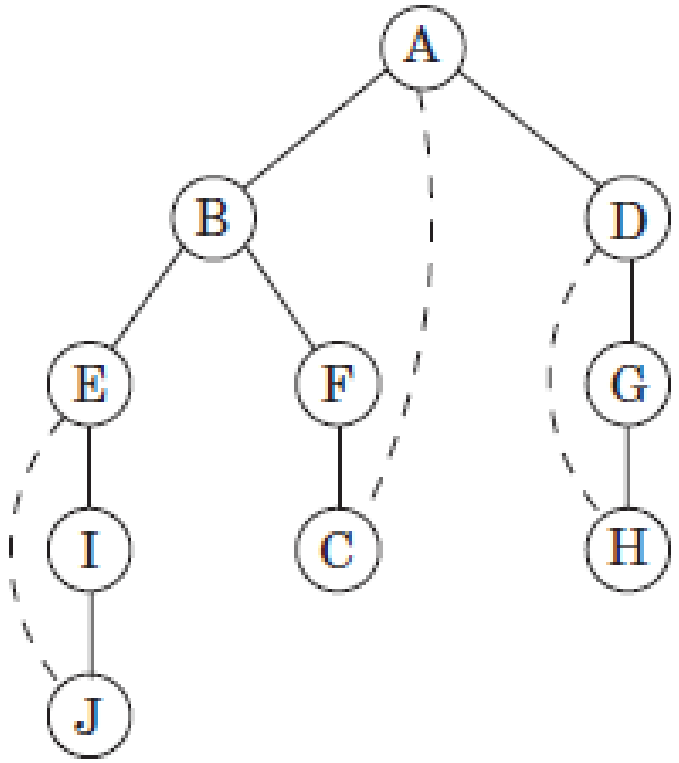
```
void dfs (G)
    for all v ∈ V
        visit[v] = False;
    for all v ∈ V
        if visit[v] == False
            dfs ( v );
```

So, in total, dfs(G) = O(n+m)

(4) Example of depth-first search (1)

– Depth-First Spanning Tree (DFS tree)

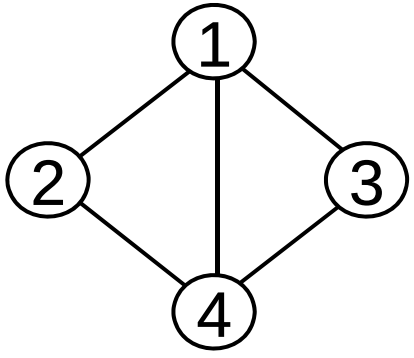




4.2 Depth-first search in undirected graphs

(5) Practical implementation of graph using STL

```
Input: // undirected graph
4 5    // 4: No. of vertices (1 ~ 4)
       // 5: No. of edges
1 2
1 3
1 4
2 4
3 4
```



4.2 Depth-first search in undirected graphs

(5) Practical implementation of graph using STL

```
#include <vector>
#include <algorithm>

vector<int> edge[10001];

void main ( )
{
	scanf("%d %d %d", &n, &m, &s);
	for (i = 0; i < m; i++){
		scanf("%d %d", &u, &v);
		edge[u].push_back(v);
		edge[v].push_back(u);

	}
	for (i = 1; i <= n; i++)
		sort(edge[i].begin(), edge[i].end());
}
```

4.2 Depth-first search in undirected graphs

(5) Practical implementation of graph using STL

```
bool visit[10001];

void initVisit( )
{
      for (int i = 0; i < 10001; i++)
            visit[i] = false;
}

void main ( )
{
      initVisit ( );
      for (i = 1; i <= n; i++) {
            if ( visit[i] == false )
                  dfs ( i );
}
```

4.2 Depth-first search in undirected graphs

(5) Practical implementation of graph using STL

```
void dfs( int v )
{
	if (visit[v] == true){
		return;
	}
	printf("%d ", v);
	visit[v] = true;
	for (int i = 0; i < edge[v].size(); i++){
		dfs(edge[v][i]);
	}
}
```

4.2 Depth-first search in undirected graphs

(6) Previsit and postvisit ordering

- At visiting vertices, mark
  - The first time we visit → previsit
  - The last time we departure → postvisit
- For any nodes u and v, the two intervals [pre(u), post(u)] and [pre(v), post(v)] are either disjoint or one is contained within the other.

```
previsit ( v );
for each edge (v, u) ∈ E
    if visit[u] == False
        dfs ( u );
postvisit ( v );
```

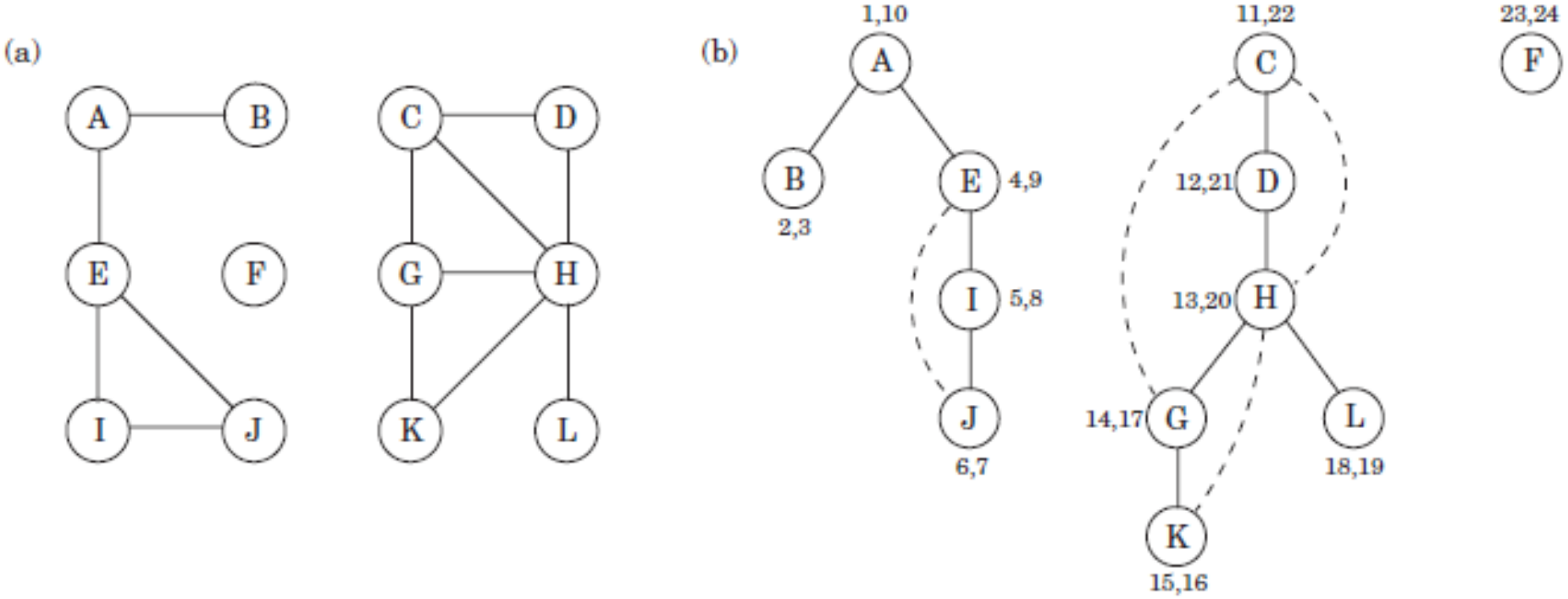
```
procedure previsit(v)
pre[v] = clock
clock = clock + 1

procedure postvisit(v)
post[v] = clock
clock = clock + 1
```

(6) Previsit and postvisit ordering

- Property
  - For any nodes u and v, the two intervals [pre[u], post[u]] and [pre[v], post[v]] are either disjoint or one is contained within the other.

(7) Example of depth-first search (2)

– with previsit & postvisit



(8) Connected components

- Strategy
  - Use a search algorithm to explore all the vertices in a connected component
  - In calling `previsit (v)`,

```
procedure previsit(v)
ccnum[v] = cc
```

  - cc
    - initialized as 0
    - Incremented each time `dfs( v )` is called

4.2 Depth-first search in undirected graphs

다음 중 DFS로 할 수 없는 것은?

(a) 어떤 그래프에서 두 vertex가 연결되어 있음을 알 수 있다.

(b) 어떤 그래프에서 두 vertex 사이의 거리를 알 수 있다.

(c) 어떤 그래프에서 몇 개의 edge가 있는지 알 수 있다.

(d) 어떤 그래프에서 몇 개의 vertex가 있는지 알 수 있다.