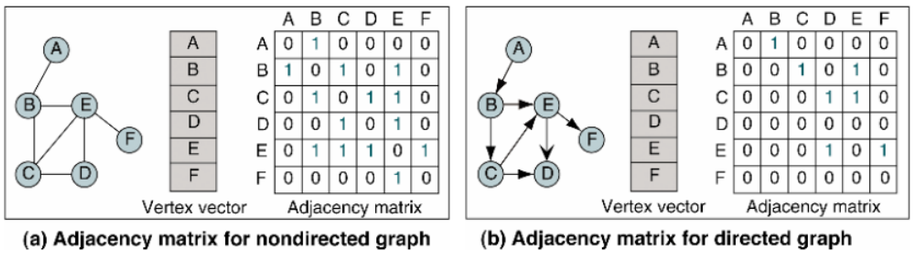
HW 8 관련 (TA 작성)

[필요한 개념]

- Graph Theory 구현 (Adjacency Matrix, Adjacency List)

Representations for Graphs

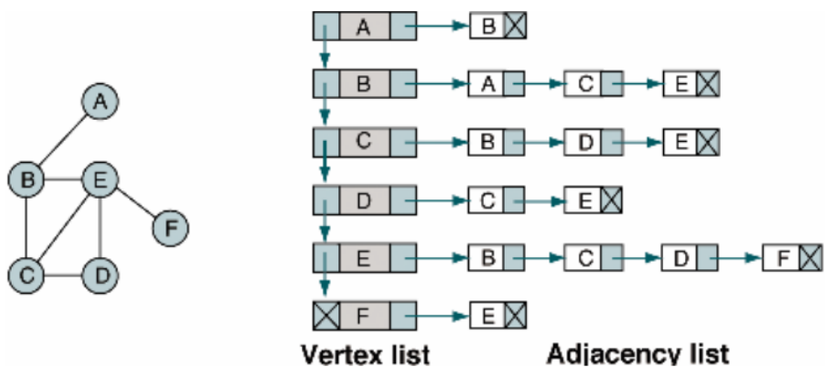
- **Adjacency matrix**: matrix to store edges



(a) Adjacency matrix for nondirected graph (b) Adjacency matrix for directed graph

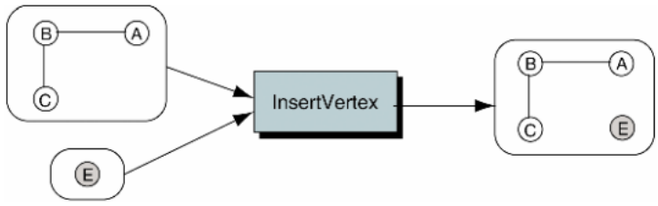
Representations for Graphs

- **Adjacency list**: 2D ragged array to store the edges

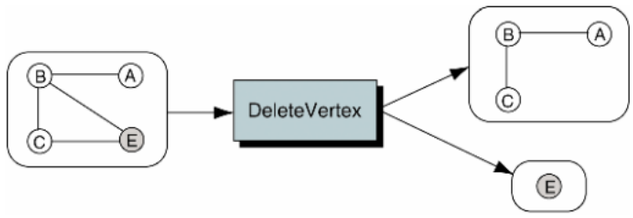


Insertion/Deletion of Vertex

- Inserting a vertex



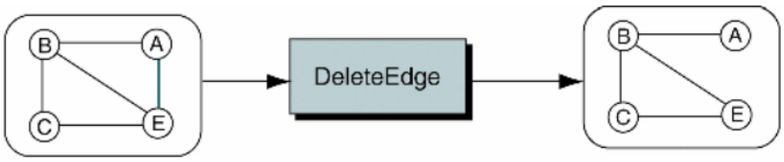
- Deleting a vertex



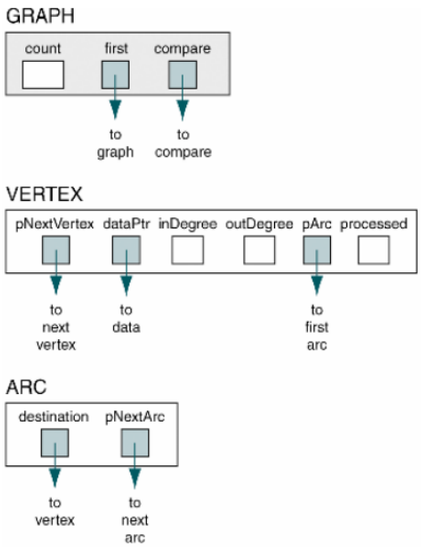
Insertion/Deletion of Edge

- Inserting an edge

- Deleting an edge



Adjacency List Representation

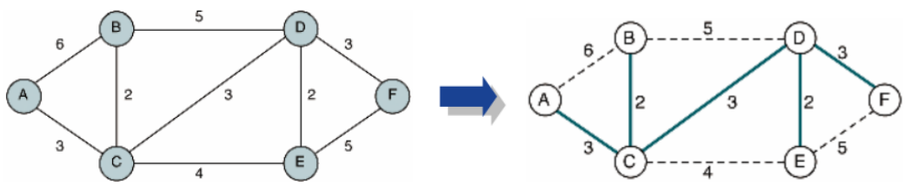



```
typedef struct
    {
     int              count;
     struct vertex* first;
     int  (*compare)
            (void* argu1,
             void* argu2);
    } GRAPH;

typedef struct vertex
 {
  struct vertex*  pNextVertex;
  void*           dataPtr;
  int             inDegree;
  int             outDegree;
  short           processed;
  struct arc*     pArc;
 } VERTEX;

typedef struct arc
 {
  struct vertex*  destination;
  struct arc*     pNextArc;
 } ARC;
```

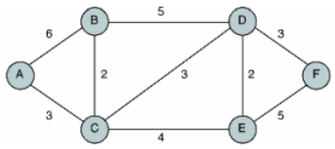
- Depth-First Traversal, Breadth-First Traversal
- 가중치가 있는 Graph 구현
- Minimum (cost) Spanning Tree 알고리즘 (Prim's / or Kruskal's, Solin's)

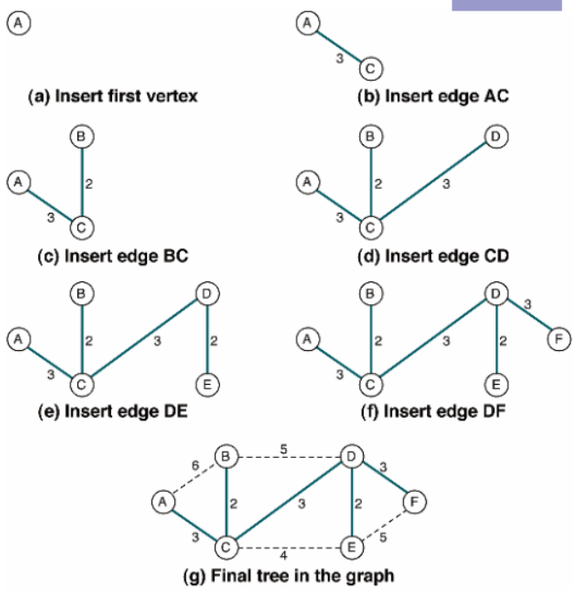
Minimum Spanning Tree

- **Minimum (cost) spanning tree: spanning tree of least cost (sum of weights)**
  - Every vertices are included
  - Total edge weight is minimum possible



Prim's Algorithm





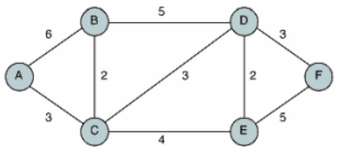
- Shortest Path 알고리즘 (Dijkstra's)

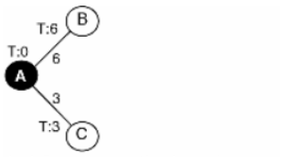
Shortest Path Algorithm

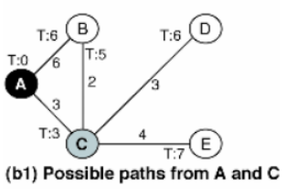
- **Dijkstra's algorithm (source vertex: v)**

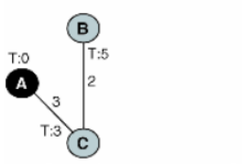
```
int i, u, w;

for(i = 0; i < n; i++){
    found[i] = FALSE;                 // if found[i] == TRUE, i is in S
    distance[i] = cost[v][i];         // v : source vertex
}

found[v] = TRUE;                      // initially S = { v }
distance[v] = 0;
for(i = 0; i < n – 1; i++){
    u = choose(distance, n, found);   // find a vertex with minimum distance
    found[u] = TRUE;                  // add u into S
    for(w = 0; w < n; w++){           // adjust distances to the vertices not in S
      if(!found[w] && distance[u]+cost[u][w] < distance[w])
        distance[w] = distance[u]+cost[u][w];
    }
}
```

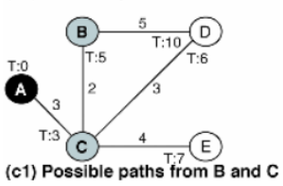
Shortest Path: An Example

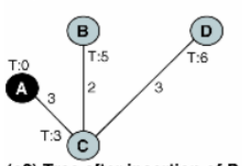




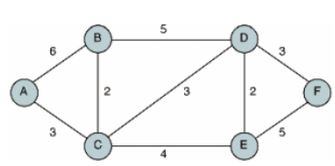


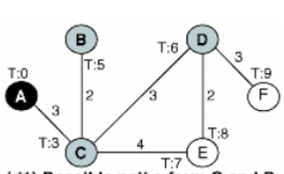


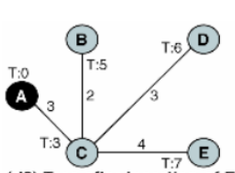


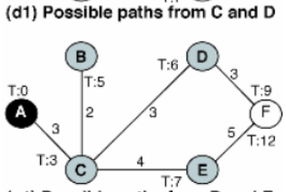


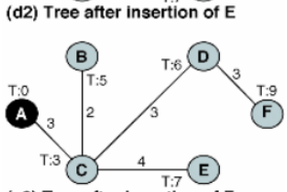
Shortest Path: An Example (Cont.)











[배점 기준]

만점: 70

[교수님 안내사항]

execution example 대로 출력이 되게 해야합니다. 다만, Depth first, Breadth first의 결과는 다를 수 있겠지요.

내부구조는 어떻게 잡든(A, B, C 순서대로 안잡아도 됨) 출력 결과는 예시와 같이 순서대로 나오게 해주세요.

여러가지 방법이 있겠지만, 입력파일을 한번 읽어서 노드를 순서대로 정렬하기 힘들면, 두 번 읽으면 되죠. 처음은 노드를 먼저 파악하고, 파일 다시 처음으로 가서 (fseek()을 쓰면 됩니다.) 두 번째는 연결을 파악하고...

1) Graph ADT 구현 (25)

- Adj Matrix (10), Adj List (15)

2) Depth-First Traversal, Breadth-First Traversal 구현 (10)

3) Adj Matrix 가중치 (5)

4)

- Minimum Spanning Tree 알고리즘 (15)
- Shortest Path 알고리즘 (15)

채점 코멘트

- 각 항목별로 ADT 기준대로 잘 구현해주신 경우 채점이 용이했지만 개인마다 코드 스타일이 너무나 천차만별로 다르기 때문에 감점 기준을 두기가 어려웠습니다. 그렇기 때문에 최대한 '관대한 기준'으로 각 항목별로 채점하고자 하였습니다. 예를 들면 Adj matrix가 동작에 문제없이 잘 구현되어 있으면 15점을 주었습니다. (느낌상 Pass or Fail)