

## 자료 구조 Lab 008 :

lab008.zip 파일 : LabTest.java lab008.java lab.in lab.out lab008.pdf

-----

### 제출

lab008.java 를 **lab\_학번.java** 로 변경하여 이 파일 한 개만 제출할 것.

---

다음은 Cost-Adjacency Matrix를 이용하여 **Directed weighted** Graph에서 All pair shortest path algorithm인 Floyd 알고리즘을 구현하는 내용이다.

이 프로그램에서는 우선 vertex의 개수를 입력하고, 그 다음에 edge를 구성하는 vertex pair를 edge별로 차례로 입력하여 graph를 구성한 후, Floyd 알고리즘을 수행한다. 그 후 사용자의 요청에 따라 주어진 source-destination pair에 대해 경로와 cost를 출력한다.

수행 예는 다음과 같다.

```
선택 sanghwan@PC-: ~/dbox/classes201/ds/lab20/lab20008
sanghwan@PC-:~/dbox/classes201/ds/lab20/lab20008$ java LabTest
Graph > init 5
Graph > edge 0 1 3
Graph > edge 0 3 2
Graph > edge 1 2 4
Graph > edge 1 4 6
Graph > edge 2 3 7
Graph > edge 2 4 3
Graph > edge 3 4 5
Graph > edge 4 0 4
Graph > floyd
Graph > outpath 0 2
Path from 0 2 : 0-1-2 => 7
Graph > outpath 3 2
Path from 3 2 : 3-4-0-1-2 => 16
Graph > outpath 4 3
Path from 4 3 : 4-0-3 => 6
Graph > quit
sanghwan@PC-:~/dbox/classes201/ds/lab20/lab20008$
```

사용자가 사용하는 명령어의 syntax는 다음과 같다. LabTest.java의 main() 함수에 정의되어 있다.

- `init numofnodes`

`numofnodes`는 vertex의 수를 의미하며, 각 vertex는 0부터 `numofnodes - 1`까지의 번호를 가지게 된다.

- `edge v1 v2 cost`

vertex `v1`과 vertex `v2`로 정의된 edge를 그래프에 추가한다. 그 edge의 cost는 이 명령어의 세 번째 파라미터인 `cost`가 된다.

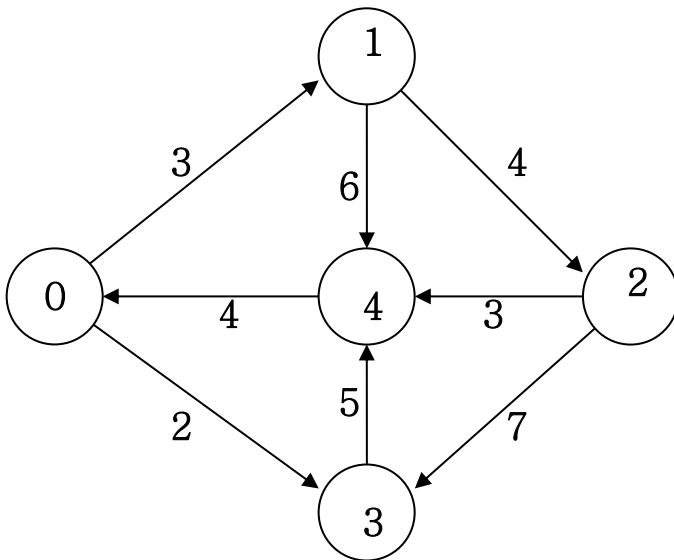
- `floyd`

`floyd` algorithm을 수행하여 내부적으로 각 pair 별로 shortest path length와 Kay 행렬을 유지한다.

- `output v1 v2`

vertex `v1`에서 `v2`로의 경로를 출력하고, 그 distance를 출력한다.

위의 화면과 같이 입력할 경우 내부적으로 다음과 같은 그래프와 cost-adjacency matrix가 구성된다.



위 그래프에 해당하는 cost-adjacency matrix는 아래와 같다.

$$Cost = \begin{bmatrix} 0 & 3 & 0 & 2 & 0 \\ 0 & 0 & 4 & 0 & 6 \\ 0 & 0 & 0 & 7 & 3 \\ 0 & 0 & 0 & 0 & 5 \\ 4 & 0 & 0 & 0 & 0 \end{bmatrix}$$

이 내용을 구현하기 위해 다음 세 함수를 구현해야 한다.

- `void Edge(int v1, int v2, int cost);`

v1과 v2는 한 edge를 구성하는 vertex를 의미한다. 이 함수는 이 edge를 그래프에 추가하는 일을 한다. 클래스 Graph에는 Cost는 2차원 배열이 Cost-Adjacency Matrix를 구성하는데, v1과 v2에 의해 결정되는 이차원 배열 Cost의 entry를 cost로 수정해야 한다. 이 그래프가 **Directed** Graph임에 주의한다.

- `void Floyd();`

교과서 또는 강의 자료에 나온 알고리즘을 구현한다. 클래스 Graph 내에 이차원 배열인 c와 kay를 생성했는데 c[i][j]는 Floyd 알고리즘을 수행된 후에 vertex i에서 vertex j로의 shortest path length가 저장되고, kay[i][j]에는 i와 j 사이의 가장 높은 노드 번호가 저장된다.

- `String outputPath(int i, int j);`

강의 자료에 나온 outputpPath를 구현한다. 초기에 kay[i][j]는 -1로 초기화 되어 있고, 노드 번호는 0에서 numofnodes-1까지 임에 주의한다. 다만 경로를 바로 화면으로 출력하지 않고, 그 경로를 String에 저장하여 return 한다.

## 프로그램 테스트

```
$ diff aaa lab.out
```

또는

```
$ diff -i --strip-trailing-cr -w aaa lab.out
```