자료 구조 HW #2 (30점)

제출일 : 2017년 5월 31일 수요일 (eCampus)

제출물 : 학번.java 파일 하나

Hard copy 제출 필요 없음

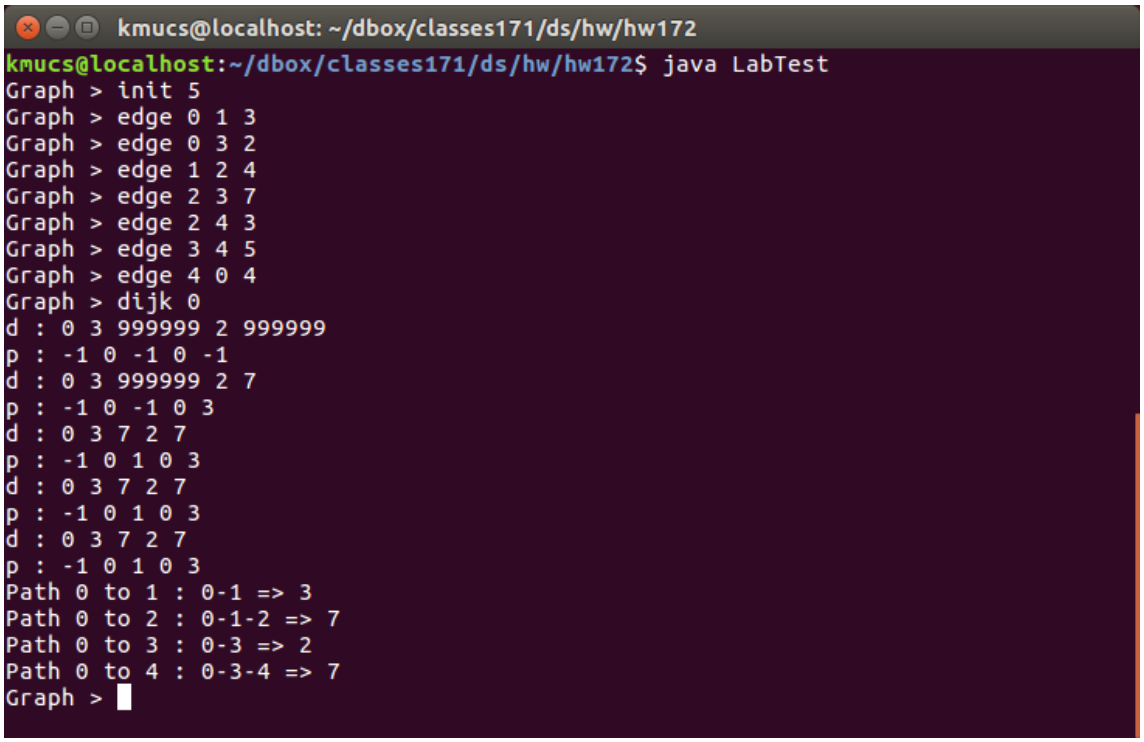
hw2.zip 파일 : LabTest.java hw2.java lab.in lab.out hw2.pdf

제출

hw2.java 를 학번.java 로 변경하여 이 파일 한 개만 제출할 것.

다음은 Cost-Adjacency Matrix를 이용하여 Directed weighted Graph에서 Dijkstra's 알고리즘을 구현하는 내용이다.

이 프로그램에서는 우선 vertex의 개수를 입력하고, 그 다음에 edge를 구성하는 vertex pair 및 cost를 edge별로 차례로 입력하여 graph를 구성한 후, Dijkstra's 알고리즘을 수행한다. 수행 예는 다음과 같다.



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

- init numofnodes

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

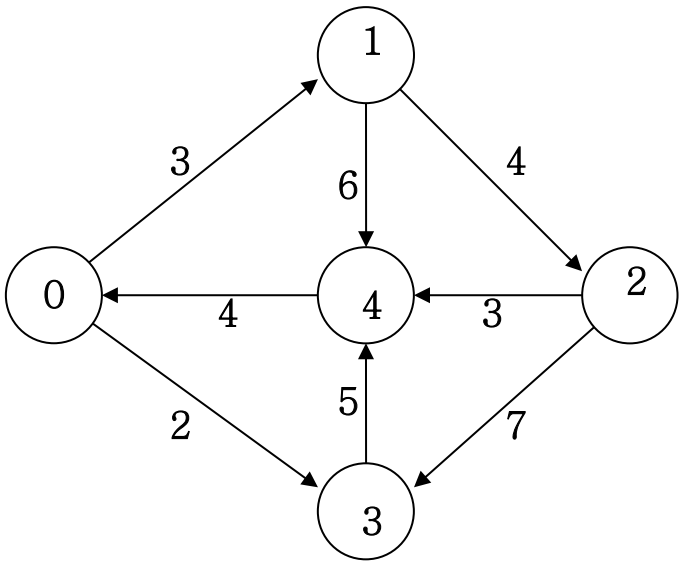
- edge v1 v2 cost

  vertex v1과 vertex v2로 정의된 edge를 그래프에 추가한다. 그 edge의 cost는 이 명령어의 세 번째 파라미터인 cost가 된다.

- dijk src

  Dijkstra's algorithm을 수행하여 내부적으로 d[]와 p[] 배열을 update 시켜 나가는데, 각 스텝별로 변화되는 d[]와 p[]를 보여준다. 그리고 최종적으로 src에서 다른 모든 노드로 가는 경로를 출력한다. 경로와 더불어 "=>" 뒤에 경로의 길이도 같이 출력한다. 예를 들면 "Path 0 to 2 : 0-1-2 => 7" 는 0에서 2로 가는 경로가 0-1-2 이며 경로의 길이는 7임을 보여준다.

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



위 그래프에 해당하는 cost-adjacency matrix는 아래와 같다.

$$Cost = \begin{bmatrix} 0 & 3 & 999999 & 2 & 999999 \\ 999999 & 0 & 4 & 999999 & 6 \\ 999999 & 999999 & 0 & 7 & 3 \\ 999999 & 999999 & 999999 & 0 & 5 \\ 4 & 999999 & 999999 & 999999 & 0 \end{bmatrix}$$

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. Edge가 존재하지 않을 경우에는 cost가 999999가 된다.

- `void Dijk(int src);`

교과서 또는 강의 자료에 나온 Dijkstra's 알고리즘을 구현한다. 필요시 Graph 클래스 내에 추가적인 method나 변수를 선언할 수 있다. 각 스텝별로 변화되는 d[]와 p[]를 보여주는데, 출력 포맷은 위 예제를 참고한다.

- `void ShowAllPath();`

Dijk() 함수를 사용하여 src로부터의 경로를 계산한 후, 다른 모든 노드로의 경로를 출력한다.

프로그램 테스트

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

또는

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