

자료 구조 Lab 009 :

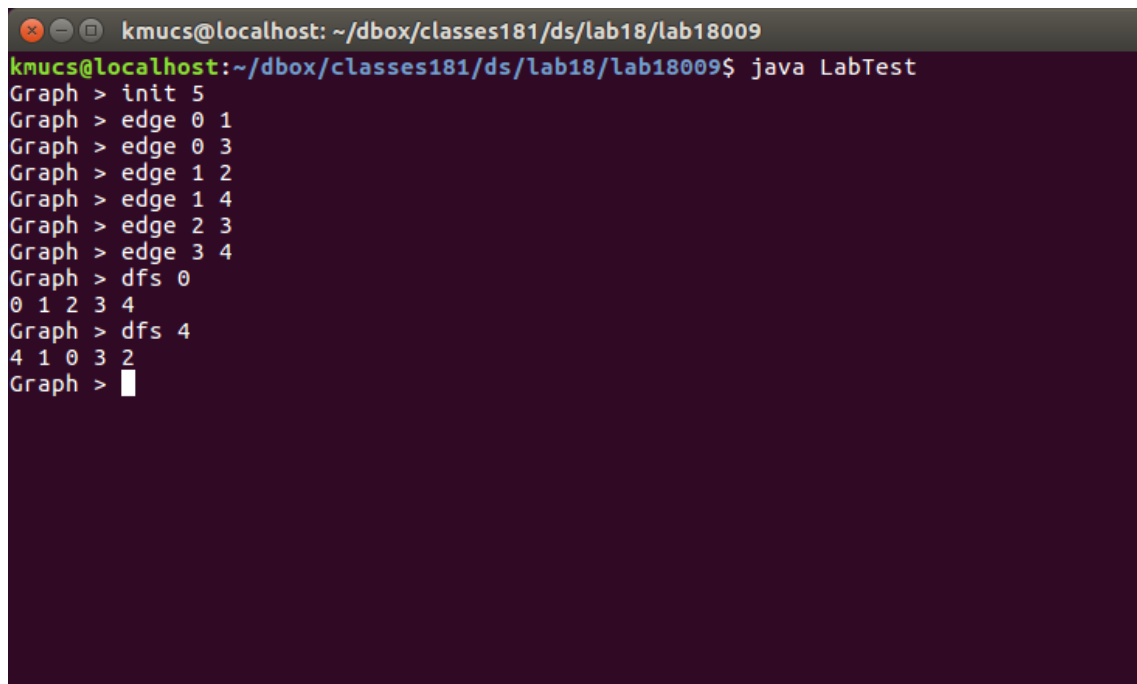
lab009.zip 파일 : LabTest.java lab009.java lab.in lab.out lab009.pdf

제출

lab009.java 를 학번.java 로 변경하여 이 파일 한 개만 제출할 것.

다음은 Adjacency Matrix를 이용하여 Unweighted Undirected Graph를 구현하는 내용이다. 이 프로그램에서는 우선 vertex의 개수를 입력하고, 그 다음에 edge를 구성하는 vertex pair를 edge별로 차례로 입력하여 graph를 구성한 후, Depth First Search를 수행하는 작업을 한다.

수행 예는 다음과 같다.



```
kmucs@localhost: ~/dbox/classes181/ds/lab18/lab18009
kmucs@localhost:~/dbox/classes181/ds/lab18/lab18009$ java LabTest
Graph > init 5
Graph > edge 0 1
Graph > edge 0 3
Graph > edge 1 2
Graph > edge 1 4
Graph > edge 2 3
Graph > edge 3 4
Graph > dfs 0
0 1 2 3 4
Graph > dfs 4
4 1 0 3 2
Graph > 
```

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 e1 e2

vertex e1과 vertex e2로 정의된 edge를 그래프에 추가한다.

- `dfs v`

vertex `v`에서 시작하는 depth first search를 수행하여 그 결과를 출력한다.

이 내용을 구현하기 위해 다음 두 가지 함수를 구현해야 한다.

- `void Edge(int e1, int e2);`

`e1`과 `e2`는 한 edge를 구성하는 vertex를 의미한다. 이 함수는 이 edge를 그래프에 추가하는 일을 한다. 클래스 `Graph`에는 `Adj`라는 2차원 배열이 Adjacency Matrix를 구성하는데, `e1`과 `e2`에 의해 결정되는 `Adj`의 entry를 수정해야 한다. 이 그래프가 **Undirected Graph**임에 주의한다.

- `void Dfs(int v);`

vertex `v`로부터 시작하는 Depth First Search를 수행하는 재귀 함수이다. 클래스 `Graph`의 `mark`라는 integer array를 이용하여 방문했는지 아닌지를 체크하도록 했다. 즉 `mark[i]`가 0이면 vertex `i`가 아직 방문되지 않은 것이고, 1이면 방문된 것이다. DFS에서는 인접 vertex 중에서 하나를 선택해서 `Dfs()`를 call 하게 되는데 일관성을 위해 Vertex `v`의 adjacent vertex 중에서 번호가 가장 작은 것을 선택한다 (그냥 for loop를 사용하면 저절로 해결되는 사항임). 물론 이 vertex는 기존에 방문하지 않은 vertex여야 한다.

프로그램 테스트

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

또는

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