

자료 구조 Lab 008 :

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

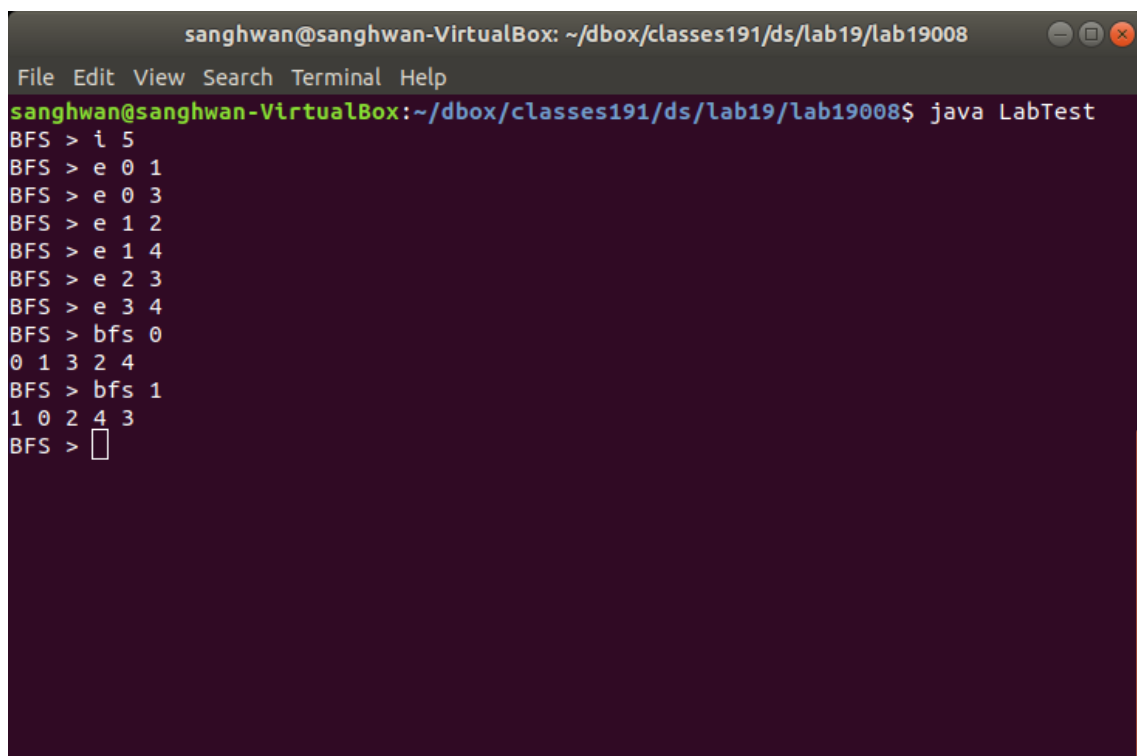
제출

lab008.java 를 학번.java 로 변경하여 이 파일 한 개만 제출할 것.

다음은 Adjacency Matrix를 이용하여 **Unweighted Graph**를 구현하는 내용이다.

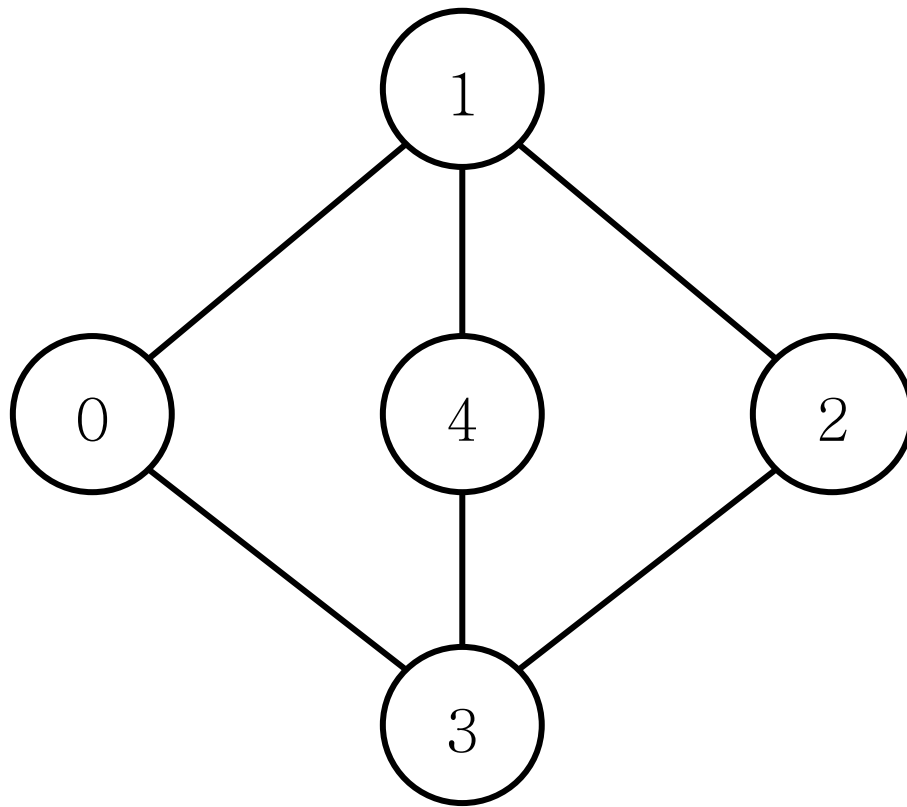
이 프로그램에서는 우선 vertex의 개수를 입력하고, 그 다음에 edge를 구성하는 vertex pair를 edge별로 차례로 입력하여 Graph를 구성한 후, Breadth First Search를 수행하는 작업을 한다.

수행 예는 다음과 같다.



```
sanghwan@sanghwan-VirtualBox: ~/dbox/classes191/ds/lab19/lab19008
File Edit View Search Terminal Help
sanghwan@sanghwan-VirtualBox:~/dbox/classes191/ds/lab19/lab19008$ java LabTest
BFS > i 5
BFS > e 0 1
BFS > e 0 3
BFS > e 1 2
BFS > e 1 4
BFS > e 2 3
BFS > e 3 4
BFS > bfs 0
0 1 3 2 4
BFS > bfs 1
1 0 2 4 3
BFS > 
```

이 예는 아래와 같은 그래프를 입력 받은 후, vertex 0 번과 vertex 1번에서 각각 BFS를 수행한 내용을 보여준다.



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

- `i numofnodes`

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

- `e n1 n2`

vertex `n1`과 vertex `n2`로 정의된 edge를 그래프에 추가한다.

- `bfs v`

vertex `v`에서 시작하는 breadth first search를 수행하여 방문한 노드를 차례로 출력한다.

이 내용을 구현하기 위해 다음 두 가지 함수를 구현해야 한다.

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

`v1`과 `v2`는 한 edge를 구성하는 vertex를 의미한다. 이 함수는 이 edge를 그래프에 추가하는 일을 한다. 클래스 `Graph`에는 `AdjMatrix`라는 2차원 배열이 Adjacency Matrix를 구성하는데, `v1`과 `v2`에 의해 결정되는 `AdjMatrix`의 원소를 수정해야 한다. 이 그래프가 **Undirected Graph**임에 주의한다.

- `void Bfs(int v);`

vertex v로부터 시작하는 Breadth First Search를 수행하는 함수이다. BFS 알고리즘은 Queue를 사용하는데 Java에서 제공하는 Queue Interface를 이용한다. 사용법은 다음과 같다.

```
Queue<Integer> q = new LinkedList<>();
```

관련 웹페이지: method 목록을 참고하여 사용하면 됨.

<https://docs.oracle.com/javase/7/docs/api/java/util/Queue.html>

BFS에서는 인접 vertex를 queue에 Push 해야 하는데 일관성을 위해 인접 vertex들을 번호가 작은 것에서 큰 순서로 queue에 추가시킨다. 물론 이 vertex는 기존에 방문하지 않은 vertex여야 한다.

주어진 .java 파일을 컴파일 하면 수행은 가능하지만 아직 구현이 안된 부분은 비어 있다.

프로그램 결과 테스트

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

또는

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