

# CS2233: Data Structures

## Assignment 5

30th September, 2018

### Problem Statement

- Input: An undirected, unweighted graph  $G = (V, E)$ .
- Goal: Serve the following requests:
  - Is  $\{u, v\} \in E$ ?
  - Breadth-first Search (BFS) of  $G$  from a source vertex  $v$ .
  - Shortest path from a vertex  $u$  to  $v$ .

### Input Format

#### General format:

A graph  $G = (V, E)$  is defined by first providing the number of vertices  $n = |V|$ .

Henceforth you shall assume that the vertex set is  $V = \{1, 2, \dots, n\}$ .

The edge set  $E$  is provided as a list of adjacency lists – each line indicating the adjacency list of a different vertex.

Requests arrive only after the definition of  $G$ .

#### Format in detail:

Each line of the input looks like one of the following:

- ‘N’ followed by number of vertices  $n \in \mathbb{N}$ .
- ‘E’ followed by vertices  $u, v_1, v_2, \dots, v_k \in [n]$  that indicates the adjacency list of vertex  $u$ . The list is given as a space-separated list.
- ‘?’ followed by  $u, v \in [n]$  with a space separating them.
- ‘B’ followed by a  $u \in [n]$
- ‘P’ followed by  $u, v \in [n]$  with a space separating them.

Each of the lines above ends with a ‘\n’ character. All lists are given as elements separated by a space. No commas are used anywhere. All numbers used will fit inside an `int`. End of input is indicated by `EOF`.

## Output Format

- If input line started with ‘N’ or ‘E’, then no corresponding output.
- If input line was “?  $u$   $v$ ”: Output 1 if  $\{u, v\} \in E$ .
- If input line was “B  $u$ ”:  
Output a Breadth-first search of the graph starting from vertex  $u$  as a space-separated list of vertices. The BFS you compute has to *comply with the implementation rules* found in the next section.
- If input line was “P  $u$   $v$ ”:  
If  $v$  is not reachable from  $u$ , output  $-1$ . Else output a shortest path from  $u$  to  $v$  as a space-separated list of vertices starting with  $u$ .

All output lines have to end with a ‘\n’ character.

## Implementation rules

- The input graph  $G = (V, E)$  has to be stored as adjacency lists: use an array  $A$  of length  $|V|$  such that  $A[i]$  (with base index 1) is the head of the adjacency list of vertex number  $i$ . Minor variations due to base index or convenience is allowed.
- Store each vertex’s adjacency list in the same order as provided in the input.
- Computing BFS requires a queue. Implement a queue yourself by using an array or linked list.
- During computation of BFS, if the vertex visited is  $u$ , then enqueue the neighbours of  $u$  in the same order as they appear in the adjacency list.

## Design decisions

- Before deciding the way your nodes look in the adjacency lists, it might be wise to read the other assignment sheets. You might be able to create the nodes in a way that allows you to reuse the input handling routine for other assignments.
- There are only three possible colors for each vertex – White, Grey, Black. So just two bits should suffice to keep track of a vertex color.
- If you are issued consecutive requests of shortest path from the same source node, it is unnecessary to recompute the Breadth-first search from the same source for each request. You should avoid such recomputation.

## Other Remarks

- A good starting point is to program a Queue. Test the queue thoroughly to make sure it does not become the source of bugs later on.
- **Deadline:** 14th October, 2018 .

## Example input

Input:

```
-----  
N 4  
E 1 3 4 2  
E 2 1 3  
E 4 3 1  
E 3 2 1 4  
? 2 4  
B 2  
B 1  
P 2 4  
N 14  
E 1 13 5 4  
E 2 3 13  
E 3 2  
E 4 1  
E 7 9 13  
E 5 10 8 14 1  
E 9 6 7  
E 6 9  
E 8 10 5 14  
E 10 8 5 14  
E 11 12  
E 12 11  
E 13 1 2 7  
E 14 8 5 10  
? 12 11  
? 2 1  
B 7  
P 7 1  
P 7 10  
P 7 12  
P 8 14  
P 8 1  
B 8  
B 11  
-----
```

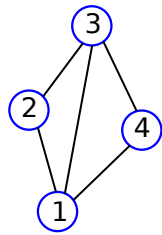
Output:

```
-----  
0  
2 1 3 4  
1 3 4 2  
2 1 4  
1  
0  
7 9 13 6 1 2 5 4 3 10 8 14  
7 13 1  
7 13 1 5 10  
-1  
8 14  
8 5 1  
8 10 5 14 1 13 4 2 7 3 9 6  
11 12  
-----
```

See next page for the graphs given in the above input.

## Example graphs

Graph 1



Graph 2

