CS700: Algorithms and Complexity

Lab Assignment No: 04

Submission Deadline: 07 January 2021

Problem Statement: Bharatiya Rastriya Rajmarg Pradhikaran (NHAI) is a government organization that looks into the construction and maintenance of highways in Bharat which has n cities numbered 0, 1, 2, . . . , n−1. Let $G = (V, E)$ be an undirected graph, where V is the set of cities in Bharat. Two cities u and v share an edge in the graph if and only if there is a highway between the two cities. The Parivahan Mantralaya notices that not all pairs of cities are connected by paths consisting only of highways. So NHAI is assigned the job of constructing new highways so that any two cities become highway-connected (by paths, not edges).

The head office of NHAI is in City 0, and the cities $1, 2, 3, \ldots, n-1$ are arranged in an increasing sequence of distance from City 0 (you do not need to know the exact distances for solving this problem). NHAI engineers decide to build k new highways between City 0 and Cities $u_1, u_2, \ldots, u_k$ with $1 \leq u_1 < u_2 < \cdots < u_k \leq n-1$, that is, to add the new edges $(0, u_i)$ to E for $i = 1, 2, \ldots, k$, so that G becomes a connected graph. In order to minimize the construction cost, it is required to have k as small as possible. Also the highway lengths should be small, so u_1, u_2, . . . , u_k should be chosen as small as possible.
In this assignment, you help NHAI engineers by providing an optimal solution for k and $u_1, u_2, \ldots, u_k$.

Part 1: Write a function ***read_graph*()** to construct and return the highway graph G based upon user inputs. The user first enters n (the number of nodes, that is, cities) and m (the number of edges, that is, highways). The user then enters one by one the two endpoints of the m edges. It is the duty of the user to ensure that the same edge is not added multiple times. You prepare G in an adjacency-list representation. For two (different) cities u and v, the existence of the undirected edge (u, v) implies that v should be in the list of neighbors of u, and u should be in the list of neighbors of v.

Part 2: Write a function ***BFS*(*G*, *u*, *visited*)** to implement a BFS traversal in G starting from the node u. The ***visited*** array should be passed to the function, that is, it should not be a local array in the function. The traversal stops as soon as the BFS queue becomes empty, even if there still remain unvisited nodes.

Part 3: Write a function ***check_connectivity*(G)** to check the connectivity of *G*. It invokes the function *BFS*, and prints a message whether *G* is connected or disconnected by looking at the ***visited*** array after *BFS* returns.

Part 4: Add the minimum number *k* of edges of the form $(0, u_i)$, $i = 1, 2, \ldots, k$, so that *G* becomes connected. Ensure also that u_i are as small as possible. Write a function ***build_highways*(G)** to solve this problem. Notice that *k* is not supplied as input to this function; it will be determined by the input graph *G*.

The *main*() function

- Prepare the graph *G* from user inputs (taken in a file) by calling *read_graph*. Print (in a file) the graph in the format shown in the sample output.
- Assume that *G* is disconnected at this point. Verify this by calling *check_connectivity*.
- Call *build_highways* to make *G* connected.
- Call *check_connectivity* again to verify that *G* is now connected.

N.B: Do not use global/static variables.

Sample output

```
No. of vertices n = 25

No. of edges m = 36

Adding edges:

    (2, 5) (11, 7) (3, 4) (13,12) (8,18) (8,17) (1, 4) (3,1) (8, 0)
    (5, 6) (15,14) (0,18) (8,16) (9, 22) (2, 6) (14, 8) (19, 5) (9, 21)
    (21, 22) (16,18) (24,12) (14,18) (24,13) (16,14) (19,20) (21,23) (8,15)
    (23, 9) (17,18) (16, 0) (16,17) (2, 20) (23,22) (20, 6) (15,18) (0,15)
```

The graph is:

```
City  0  --  8   15  16  18
City  1  --  3   4
City  2  --  5   6   20
City  3  --  1   4
City  4  --  1   3
City  5  --  2   6   19
City  6  --  2   5   20
City  7  --  11
City  8  --  0   14  15  16 17 18
City  9  --  21  22  23
City 10  --
City 11  --  7
City 12  --  13  24
City 13  --  12  24
City 14  --  8   15  16  18
City 15  --  0   8   14  18
City 16  --  0   8   14  17 18
City 17  --  8   16  18
City 18  --  0   8   14  15 16 17
City 19  --  5   20
City 20  --  2   6   19
City 21  --  9   22  23
City 22  --  9   21  23
City 23  --  9   21  22
City 24  --  12  13
```

Not all cities are connected by highways

Building highways:

```
Between City 0 and City 1
Between City 0 and City 2
Between City 0 and City 7
Between City 0 and City 9
Between City 0 and City 10
Between City 0 and City 12
```

All cities are connected by highways

****