CS69001 Computing Laboratory – I

Assignment No: B3

Date: 21–August–2019

Let $G = (V, E)$ be a connected undirected graph, and let s, t be two distinct vertices of G. Each vertex $v \in V$ stores a positive integer $l(v)$. Two robots are initially sitting at vertices s and t. They then enter a loop. Let u and v be the current positions of the two robots (initially $u = s$ and $v = t$). The robots read the two integers $l(u)$ and $l(v)$. The first robot then moves along a *path* in G of length equal to $l(u)$, starting from u. Likewise, the second robot starts from v, and moves along a *path* in G of length equal to $l(v)$. If the new positions of the robots are u' and v', we say that the robots move from (u, v) to (u', v') in one step or iteration. Your task is to find out whether the two robots, starting initially from s and t, can switch their positions, that is, can reach t and s *simultaneously* after the same number of steps. Notice that in a single step, no path of either robot is allowed to contain repeated vertices. However, the paths of each robot in different steps (iterations) may contain repeated vertices. If in some step, a path of the desired length does not exist, then the problem definitely has no solution. Assuming that the problem is solvable, we ask two questions.

1. What is the minimum number of steps (that is, iterations) needed for the robots to switch positions?
2. The effort of each robot is the sum of the lengths of the paths it traverses in all the iterations. What is the minimum combined effort (the sum of the efforts of the two robots) needed for the robots to change their positions from (s, t) to (t, s)?

You consider two situations. Let (u, v) be the positions of the robots at the end of an iteration.

(i) It is allowed to have $u = v$ at the end of one or more iterations.

(ii) It is never allowed to have $u = v$.

Part 1: Declare and print a graph

Define a suitable data type to store a graph $G = (V, E)$ (directed or undirected) in the adjacency-list format. The structure should store the number n of vertices of G, the number m of edges of G, and n headers to the linked lists of neighbors of the vertices. In this assignment, each vertex v is needed to store the path length $l(v)$, so your graph should also contain an integer array of size n. Assume that $V = \{0, 1, 2, \ldots, n-1\}$.

Write a function *prngraph* to print a graph G in the format shown in the sample output. You first print n and m. Then, for each vertex v, you print its weight followed by its neighbors.

Part 2: Generate the one-step reachability graph G_R

The user enters the input undirected graph $G = (V, E)$ in the *main* function. From this, you generate a graph $G_R = (V_R, E_R)$ as follows. We have $V_R = V$. Let $u \in V_R = V$. If a robot sits at u at some point of time, it can reach a vertex v from u in one step if there is a u, v path of length $l(u)$. If and only if this is the case, include the edge (u, v) in E_R. For each $v \in V$, copy $l(v)$ from G to G_R. Write a function *genrgraph* to generate G_R from G. Notice that G_R is a directed graph (even though G is undirected).

Part 3: Generate the one-step simultaneous reachability graph G_S

You need to track the simultaneous movements of the two robots. This is captured by a bigger graph $G_S = (V_S, E_S)$, and can be generated from G_R. Take $V_S = V_R \times V_R = V \times V$. Renumber the pair (u, v) of vertices of G_S as $nu + v$, so $V_S = \{0, 1, 2, \ldots, n^2 - 1\}$. Include the edge $((u, v), (u', v'))$ in E_S if and only if $(u, u') \in E_R$ and $(v, v') \in E_R$ (so E_S is essentially $E_R \times E_R$). Finally, store $l(u, v)$ in G_S as $l(u) + l(v)$. Write a function *gensgraph* to generate G_S from G_R. Notice that G_S is again a directed graph.

Part 4: Minimize the number of steps

The initial position of the robots is (s, t), and their final position is (t, s). Run a BFS in G_S from (s, t) . If (t, s) is ever reached, the minimum number of steps is the level of (t, s) in the BFS tree. Print the unique path from (s,t) to (t,s) in the BFS tree, its length, and its cost (combined total effort of the robots on this path). If (t,s) is never reached from (s,t), then the problem has no solution; report that.

Your BFS function should be able to handle both the situations (i) and (ii) described above. We start with the assumption that $s \neq t$. In situation (i), your BFS is allowed to visit nodes (u,u). In situation (ii), you initially mark all nodes (u,u) as visited so you never try to revisit them in the BFS loop. Pass a flag to your BFS function to indicate which situation you are dealing with.

Part 5: Minimize the total combined effort

While BFS traversal can minimize the number of steps, it may fail to give the correct solution when the goal is to minimize the total combined effort of the robots. You need to run a single-source-shortest-path (SSSP) algorithm in G_S with (s,t) acting as the source. Implement Dijkstra's algorithm to this end. This algorithm requires edges to be weighted, but we stored weights in vertices. This is not a problem. For a directed edge $e = ((u,v),(u',v')) \in E_S$, the weight is $weight(e) = l(u,v) = l(u) + l(v)$, and is dependent solely on the starting vertex (u,v).

As in Part 4, pass a flag to your SSSP function to indicate which of the situations (i) and (ii) you are dealing with. In situation (ii), you do not consider vertices of the form (u,u) even though G_S stores information about these vertices.

After you find a shortest $(s,t) \to (t,s)$ path, print its cost and also its length (number of steps). If in the end, the distance of (t,s) from (s,t) remains ∞, then there is no $(s,t) \to (t,s)$ path in G_S.

The *main*() function

- Read G from the user. The user enters n, m, the n path lengths, and the undirected edges (u,v) of G, in that order. It is the duty of the user to supply data for a connected graph. Print G.
- Generate G_R from G. Print G_R.
- Generate G_S from G_R. Print G_S.
- Read s and t from the user. Again assume that the user supplies $s \neq t$.
- Call you BFS function in both situations (i) and (ii), and report the results.
- Call you SSSP function in both situations (i) and (ii), and report the results.

Submit a single C/C++ source file. Do not use global/static variables. You may use STL queues and stacks.

Sample outputs

```
n = 5
m = 4
+++ Path lengths: 3 1 3 2 1
+++ Edges:
    ( 2, 4) ( 0, 3) ( 1, 2) ( 0, 1)

+++ Input graph
    Number of vertices: 5
    Number of edges   : 4
    0   (3) ->   1,  3
    1   (1) ->   0,  2
    2   (3) ->   1,  4
    3   (2) ->   0
    4   (1) ->   2

+++ Reachability graph
    Number of vertices: 5
    Number of edges   : 6
    0   (3) ->   4
    1   (1) ->   0,  2
    2   (3) ->   3
    3   (2) ->   1
    4   (1) ->   2

+++ Simultaneous reachability graph
    Number of vertices: 25
    Number of edges   : 36
    0   (6) ->  24
    1   (4) ->  22, 20
    2   (6) ->  23
    3   (5) ->  21
    4   (4) ->  22
    5   (4) ->  14,  4
    6   (2) ->  12, 10,  2,  0
    7   (4) ->  13,  3
    8   (3) ->  11,  1
    9   (2) ->  12,  2
    10  (6) ->  19
    11  (4) ->  17, 15
    12  (6) ->  18
    13  (5) ->  16
    14  (4) ->  17
    15  (5) ->   9
    16  (3) ->   7,  5
    17  (5) ->   8
    18  (4) ->   6
    19  (3) ->   7
    20  (4) ->  14
    21  (2) ->  12, 10
    22  (4) ->  13
    23  (3) ->  11
    24  (2) ->  12

s = 0
t = 2

+++ Minimum number of steps
--- Allowing robots to share a node
    Steps: (0,2) (4,3) (2,1) (3,0) (1,4) (2,2) (3,3) (1,1) (2,0)
    Number of steps = 8
    Total effort    = 32
--- Disallowing robots to share a node
    Steps: (0,2) (4,3) (2,1) (3,2) (1,3) (0,1) (4,2) (2,3) (3,1) (1,2) (0,3) (4,1) (2,0)
    Number of steps = 12
    Total effort    = 48

+++ Minimum total combined effort
--- Allowing robots to share a node
    Steps: (0,2) (4,3) (2,1) (3,0) (1,4) (2,2) (3,3) (1,1) (2,0)
    Number of steps = 8
    Total effort    = 32
--- Disallowing robots to share a node
    Steps: (0,2) (4,3) (2,1) (3,2) (1,3) (0,1) (4,2) (2,3) (3,1) (1,2) (0,3) (4,1) (2,0)
    Number of steps = 12
    Total effort    = 48
------------------------------------------------------------------------------------------
```

```
n = 5
m = 5
+++ Path lengths: 2 1 3 1 3
+++ Edges:
    ( 0, 4) ( 1, 3) ( 0, 2) ( 0, 1) ( 1, 2)

+++ Input graph
    Number of vertices: 5
    Number of edges   : 5
    0   (2) ->   1,  2,  4
    1   (1) ->   2,  0,  3
    2   (3) ->   1,  0
    3   (1) ->   1
    4   (3) ->   0

+++ Reachability graph
    Number of vertices: 5
    Number of edges   : 12
    0   (2) ->   1,  2,  3
    1   (1) ->   0,  2,  3
    2   (3) ->   3,  4
    3   (1) ->   1
    4   (3) ->   1,  2,  3

+++ Simultaneous reachability graph
    Number of vertices: 25
    Number of edges   : 144
    0   (4) ->  18, 17, 16, 13, 12, 11,  8,  7,  6
    1   (3) ->  18, 17, 15, 13, 12, 10,  8,  7,  5
    2   (5) ->  19, 18, 14, 13,  9,  8
    3   (3) ->  16, 11,  6
    4   (5) ->  18, 17, 16, 13, 12, 11,  8,  7,  6
    5   (3) ->  18, 17, 16, 13, 12, 11,  3,  2,  1
    6   (2) ->  18, 17, 15, 13, 12, 10,  3,  2,  0
    7   (4) ->  19, 18, 14, 13,  4,  3
    8   (2) ->  16, 11,  1
    9   (4) ->  18, 17, 16, 13, 12, 11,  3,  2,  1
    10  (5) ->  23, 22, 21, 18, 17, 16
    11  (4) ->  23, 22, 20, 18, 17, 15
    12  (6) ->  24, 23, 19, 18
    13  (4) ->  21, 16
    14  (6) ->  23, 22, 21, 18, 17, 16
    15  (3) ->   8,  7,  6
    16  (2) ->   8,  7,  5
    17  (4) ->   9,  8
    18  (2) ->   6
    19  (4) ->   8,  7,  6
    20  (5) ->  18, 17, 16, 13, 12, 11,  8,  7,  6
    21  (4) ->  18, 17, 15, 13, 12, 10,  8,  7,  5
    22  (6) ->  19, 18, 14, 13,  9,  8
    23  (4) ->  16, 11,  6
    24  (6) ->  18, 17, 16, 13, 12, 11,  8,  7,  6

s = 0
t = 2

+++ Minimum number of steps
--- Allowing robots to share a node
    Steps: (0,2) (3,4) (1,1) (2,0)
    Number of steps = 3
    Total effort    = 11
--- Disallowing robots to share a node
    Steps: (0,2) (2,4) (4,1) (2,0)
    Number of steps = 3
    Total effort    = 15

+++ Minimum total combined effort
--- Allowing robots to share a node
    Steps: (0,2) (3,3) (1,1) (2,0)
    Number of steps = 3
    Total effort    = 9
--- Disallowing robots to share a node
    Steps: (0,2) (1,3) (0,1) (2,0)
    Number of steps = 3
    Total effort    = 10
```