CS 341, Spring 2019 A. Jamshidpey, E. Zima

ASSIGNMENT 4

DUE: Monday, July 15, 6 PM. DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.

Please read `http://www.student.cs.uwaterloo.ca/~cs341` for general instructions and policies. Also read Assignment section of course outline for clarification of what "justify" means.
Note: All logarithms are base 2 (i.e., $\log x$ is defined as $\log_2 x$).
Note: For all algorithm design questions, you must give the algorithm, argue the correctness, and analyze time complexity.

1 Warmup... BFS, DFS, MSTs, shortest paths ...

1.1. [4 marks] **BFS-tree versus MST**. Give an example of a connected, weighted, undirected graph G such that the BFS spanning tree of G is never the same as a minimum weight spanning tree of G, regardless of where you start BFS and how the adjacency lists are ordered. Justify the correctness of your counter-example.

1.2. [5 marks] **The same BFS and DFS.** Prove that if the BFS and DFS spanning trees of a connected undirected graph G from the same start vertex s are equal to each other, then G is a tree.

1.3. [5 marks] Use the shortest path algorithms you learned in class to give an efficient algorithm for finding the weight of a minimum weight cycle in a directed weighted graph. The graph may have negative weight edges but has no negative weight cycle. Clearly indicate which algorithms you are using, give pseudocode and argue correctness (you may assume correctness of the algorithms presented in class). Clearly state and justify the run time.

2 Change of weight after MST is found ...

[10 marks] Suppose we have an edge weighted undirected graph $G = \{V, E\}$ in adjacency list representation and a minimum spanning tree T. Let the weight function be $w : E \to \mathbb{R}^{\geq 0}$. One edge $e = (u, v)$ changes its weight and we want to update the minimum spanning tree. Give an efficient (linear time in the size of the input) algorithm to update T. You may assume that T is stored in adjacency list representation. Prove correctness of your algorithm and analyze running time.

3 Single source shortest path at the cost of BFS ...

[10 marks] Given a graph where every edge has weight equal to 1 or 3, and a source vertex, find the shortest path from source vertex to every other vertex. You cannot use Dijkstra's algorithm for this, as this particular case of the problem can be done at the cost of BFS. Give an efficient algorithm that has worst case running time complexity $\Theta(|V| + |E|)$.

4 Internet to every house.

4.1. [6 marks] There are n houses to be connected to the internet in a very sparse forest neighbourhood. There are two options for every household i:
1) to put an ethernet link to a neighbour j, which costs c_{ij} dollars, or
2) to get a router, which costs r_i dollars (price for the routers can be different depending on the location).

A house will be connected to the internet if either (1) there is some path from it along ethernet links that leads to a house with the router or (2) there is a router in the house.

Design an efficient algorithm that, given the array $r[1..n]$ and the array $c[1..n, 1..n]$ as input, finds the minimum cost to connect every house to the internet. Note that not every two houses can be connected by a link, in which case the entry $c_{ij} = \infty$.

4.2. [20 marks] Implement your algorithm from the previous question in `C` or `C++`.

Input and output The input consists of $m + 2$ lines. The first line consists of two integers n (the number of houses) and m (the number of possible ethernet connections). The following m lines have 3 integer numbers i, j, c_{ij} each, and represent the cost c_{ij} of an ethernet link from house i to house j. The last line contains n numbers $r_1, \dots, r_n$ which represent the cost of sending and installing a router in house i.

The output needs to be one line consisting of 2 two numbers: the minimal cost of connecting all the houses to the internet, and the number of routers required.

Sample input
```
  4     4
  1     2    1000
  1     4    2000
  2     4    5000
  3     4    4000
 1000  2500  3000  500
```

Sample output
```
 5500  3
```

Submission guidelines Submit one zip file through Marmoset: `https://marmoset.student.cs.uwaterloo.ca/` which must only contain your code file named routers.c/routers.cpp. You will see two sample public tests, but the main tests will be secret. Your final score will be the score of your best submission. We will compile/run your program in a Linux environment using the command "`gcc -std=c11 routers.c -o`" for `C` or "`g++ -std=c++14 routers.cpp -o`" for `C++`. There is a time limit for each test case, so make sure you have the most efficient implementation.