Homework 10

Dushan Terzikj

due 21 May, 2018, 23:55 hours

Problem 1

Your friend (who hasn't taken this course) asks you for help on implementing an algorithm for finding the shortest path between two nodes u and v in a directed graph (possibly containing negative edge weights). She proposes the following algorithm:

1. Add a large constant to each edge weight such that all weights become positive.
2. Run Dijkstra's algorithm for the shortest path from u to v.

Prove or disprove the correctness of this algorithm to find the shortest path (note that in order to disprove, you only need to give a counterexample).

Solution: This might work in several cases, but it is **not** correct. Consider the following counterexample:

$$a \xrightarrow{-3} b \xrightarrow{5} c \tag{1}$$

$$a \xrightarrow{-3} b \xrightarrow{1} d \xrightarrow{3} c \tag{2}$$

The shortest path would be the (2) and the weight is 1. In order to use Dijkstra with negative edges, let us add 3 (since -3 is the smallest negative edge) to all the edges:

$$a \xrightarrow{0} b \xrightarrow{8} c \tag{3}$$

$$a \xrightarrow{0} b \xrightarrow{4} d \xrightarrow{6} c \tag{4}$$

Now the shortest path would be (1) whose weight is 8. This happens because we add a constant on each edge, therefore the weight of the shortest path depends on how many edges we use. The real result can be done by subtracting the number of edges times the constant, but Dijkstra algorithm does not do that. Optimization for this to work can be done, however the current problem statement does not state any optimization to be done.

Problem 2

Solution: Please check file *algorithm2.h*. If you want to use the *main*() function I provided, please follow the input and output structure.

Input

N⟶number of cities
M⟶number of roads (edges)
then M lines follow in the form of A_i B_i C_i, meaning an edge from city A_i to B_i with weight C_i.
P Q⟶your city and your friend's city

Output

The city m where they should meet

Problem 3

1. N/A
2. Please check the file *p3.cpp*. If you want to use the *main*() function I provided, please follow the input and output structure below.

   Input

   N$\longrightarrow$the number of students
   then N lines follow with one integer X_i, which means that student i wants to pick on student X_i.

   Output

   A list where the students are sorted by picking order.

Problem 4

a) **Solution:** Obviously, we can make a graph out of the puzzle board. We can do it in a way that each cell on the board is a node, therefore $V = \{0, 1, ..., n^2 - 1\}$. Let $B[i, j]$ be an entry on the board B and let the board have dimensions $n \times n$. Also consider a function $f(i, j) = in + j$, where $f : n \times n \to V$, i.e., maps coordinates i, j to the index of the node in V. Then each node $v \in V$, where $f(i, j) = v$ will have neighbouring edges with the nodes $M = \{f(i + B[i, j], j), f(i - B[i, j], j), f(i, j + B[i, j]), f(i, j - B[i, j])\}$, if, of course, nodes in M actually are in interval $[0, n^2 - 1]$.

Mathematically:

$$V = \{0, 1, ..., n^2 - 1\}$$
$$E = (v \times M)$$

where $v \in V$ and elements in M are in interval $[0, n^2 - 1]$.

b) Please check *p4.cpp*.

c) Please check *p4.cpp*.