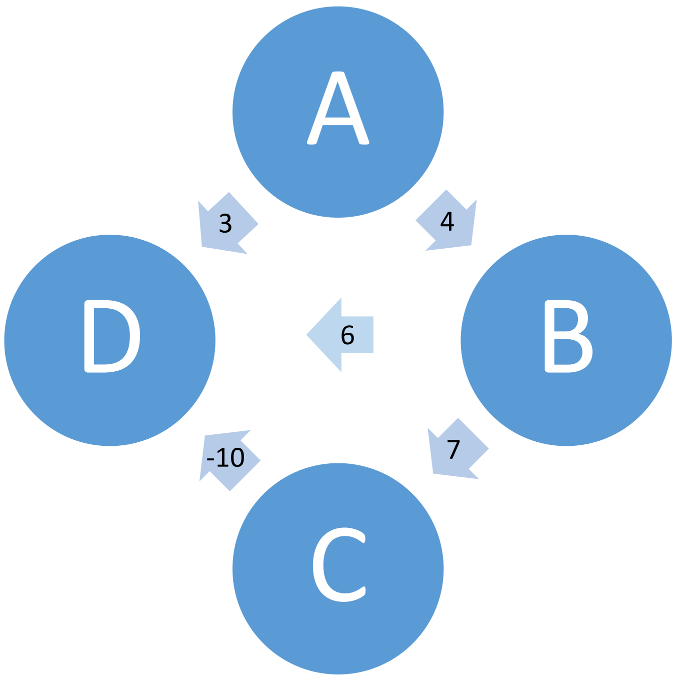
Algorithms and Data Structures

Homework 11

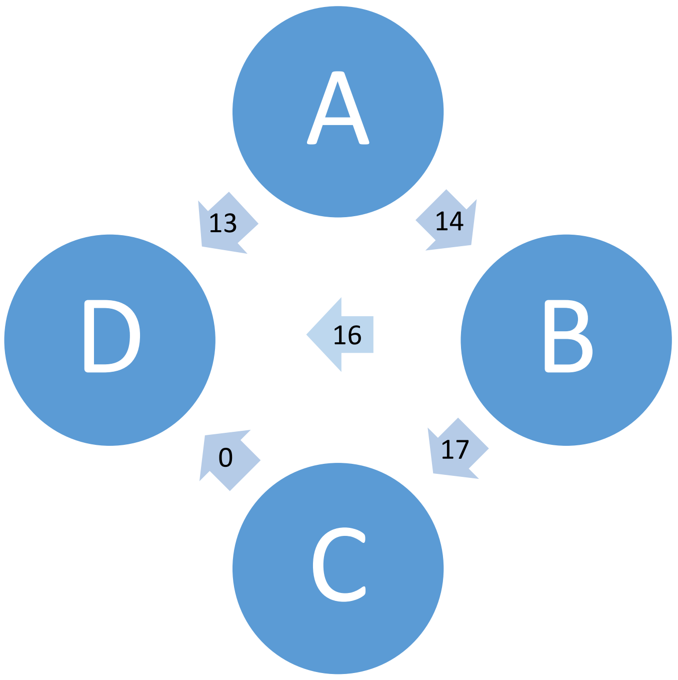
Dragi Kamov

21 May 2018

Problem 1



Before adding a constant to each weight, the shortest path would be A-B-C-D. If we add 10 to each edge weight we get:



And after this addition our Dijkstra's algorithm will give us that the shortest path is directly from A-D.

Problem 2

The solution for this problem can be found in “problem2.py”. I wrote a main function where I am asking the user first to enter the number of cities, and then I am asking the user to enter how many connections he is going to enter. After that the user is entering the number of the city that we can move from, the number of the city to which we are moving and the lastly the time needed to pass that distance. After entering all those connections the algorithm is printing the number of the city that they would have to meet in order to have the optimal meeting point.

Problem 3

a.

b. The solution for this problem can be found in “problem3.py”. I wrote a main function where I am asking the user to enter the number of middle school students and later on the user is entering which student is picking on which one starting from the 0th student.

Problem 4

a. We can represent this problem as a graph problem and we can do this by making every cell of the board a node. We would have n^2 nodes because of the $n \times n$ board. We would access every cell of the board, *mat[i][j]* with this function *f(i,j)=i*n+j.* The neighboring edges would be:

- *f(i+mat[i][j],j)*
- *f(i-mat[i][j],j)*
- *f(i,j+mat[i][j])*
- *f(i,j-mat[i][j])*

And for checking if a node is in the board we would have to check if *i* and *j* are smaller than the size of the board *n*.

b. The solution for this can be found in “problem4.py”. Here I made my own test case and tried it out. In order to execute it please uncomment the main() call.

c. The solution for this can be found in “problem4.py”. Here I made my own test case and tried it out. In order to execute it please uncomment the main() call.