Ben-Gurion University

Faculty of Engineering Sciences

Department of Electrical and Computer Engineering

Introduction to algorithms and data structures

Home assignment #2

15/6/22

Semester B / 2022

Instructions

These are the guidelines for submitting this home assignment.

1. The home assignment is submitted **only** through the moodle website! Do not send by email.
2. You should submit a pdf and one .c\ c++ file.
3. The file name should be ID1_NUMBER_ ID2_NUMBER.c.
4. There are three problems you have to provide solutions to.
5. You do not have to check the input. You can assume the input is correct.
6. The inputs are entered by hand. NOT from a file! You can use redirect values from a text file into your program from the command-line (Google it).
7. The first input will be the number of problem you want to solve.
8. The program will print the solution to the screen. For floating point answers, print only the first two digits after the point.
9. Each problem will be tested with 10 different inputs. If your solution is not correct, or your algorithm times-out (more than 1 second) you will get 0 for the test input. Otherwise, 1.
10. The assignment is in pairs. Each pair must submit only one file with the ID numbers. Deadline is in moodle website.

Programing task

Problem 1

Iceland trip

Vik the puffin is planning a long road trip around the circle road in Iceland, during which he wants to visit all the landmarks along a path of length L. The tank of Vik's car can take up to F units of fuel and for every unit of distance covered, his car consumes a unit of fuel. Using Google maps, Vik knows how far each of the N gas stations are from the beginning of the path and the price per fuel unit each station offers. At the starting point he has T units of fuel in his car.

1 Task

Write a program that will accept the above information and will calculate the minimum amount of money Vik needs to spend on gas. If the journey is impossible to make, it should print -1.

2 Input

The first line contains four space separated integers:

N (0 < N < 50001): The total number of gas stations

F (0 < F < 1000001): The units of fuel Vik's car can take

T (0 <= T <= F): The units of fuel Vik's car has at the beginning of the trip

L (0 < L < 1000000001): The path length of the landmarks he plans to visit

Each of the following N lines will contain two integers: the first one, D_i (0 <= D_i <= L) corresponds to the distance of the station from the starting point, and the second one, C_i (1 <= C_i <= 1,000,000) represents the cost per fuel unit for that station.

Note: You may assume that the trip will be on a straight line where all gas stations are spread on this line at the positions specified by their D_i values.

3 Output

The minimum amount of money to be spent or with -1 in case the trip is not feasible.

Note: There is a newline character at the end of the last line of the output.

4 Sample input

```
4 20 6 34
4 40
18 15
10 7
20 12
```

5 Sample output

348

6 Explanation

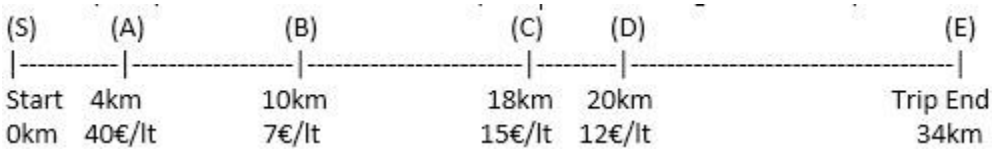
The first line of the input is 4 20 6 34 which means that:

a. There are in total N=4 gas stations on the route
b. The (max) fuel capacity of Vik's car is F=20 liters
c. The tank currently has T=6 liters of gas
d. Vik wants to travel L=34 kms in total

Then the details for the 4 gas stations are provided in the form Di Ci, where Di is the distance of this gas station from the starting point and Ci is the cost per liter of gas:

5 40
18 15
10 7
20 12

For simplicity assume that the whole trip is done in a straight line as depicted below:



Obviously, Vik does not have enough fuel for all 34 kms, so he needs to refuel. The cheapest gas station is the one labeled (B) above, however Vik does not (initially) have enough fuel in his tank to reach (B), since B-S = 10 and he has T=6. So he needs to add an extra 4 liters from gas station A, so that he can the make it until gas station B to get as much (cheap) as he can in order to make his 34 km journey. Thus, he pays (i) 4lt * 40€/lt = 160€ and now he can make it until (B). Since until this moment he has only traveled 10 kms, he needs gas for another 34-10=24kms. Normally he would want to refuel his car with 24 liters (since B is the cheapest gas station) but since his (max) fuel capacity is F=20 liters he will only take 20 liters and thus pay (ii) 20 lt * 7 €/lt = 140€. He knows however that up to point (B) he has only traveled 10kms and he needs to travel another 24kms to reach his goal, whereas he has gas for 20kms. So he would have to stop at a later gas station (after he has traveled at least 4kms) to refuel another 4 liters of gas so that he could complete the whole 34 kms journey. Since he now has quite some gas, he may decide whether he wants to refuel at (C) or at (D) and since (D) is cheaper, it is more than 4kms away from (B) and is within reach (based on his gas in the tank) he will choose to refuel another 4 liters at (D) and thus pay (iii) 4lt*12€/lt=48€). After that he can successfully reach the end point of his trip.

Problem 2

The transplant list

The organ bank has an organ transplant list that prioritizes patients according to some algorithm. The algorithm is: $(85 - AGE) + (\frac{3}{TTP}))$, where AGE is the age of the patient, and TTP is the time to perish unless an organ is provided (value provided by the treating doctor). The transplant list is a dynamic queue that helps the organ bank to decide which patient will get a donated organ.

1 Task

Write a program that manages a queue of patients. The program can add new patients to the queue, remove patients from the queue and update patients in the queue.
Removing a patient from the queue can occur for two reasons:

1. Patient no longer needs a transplant.
2. An organ arrived and the first patient in the queue is removed.

2 Input

First line contains the maximum number of patients in queue, N, an integer K, and the number of tasks M.

5<=N<50,000

1<K<N - Used for printing the ID of the Kth patient in the queue.

1<M<= N

Each line after is a task index with its arguments if necessary. Argument can be a person's ID, or person's ID and age and TTP values.

The person ID is a 9-digit integer.

T - task index. An integer between 1-5.

T = 1:

Add a patient to the queue. **Line have the following construct:**

1 ID_OF_PATIENT AGE TTP

T = 2:

Update patient in the queue. You can assume patient will be somewhere in the queue. **Line have the following construct:**

2 ID_OF_PATIENT AGE TTP

T = 3:

Remove patient from the queue. **Line have the following construct:**

3 ID_OF_PATIENT

T = 4:

Hallelujah!! an organ arrived for transplant. Remove first patient in the queue. **Line have the following construct:**

4

T = 5:

Print the Kth ID_OF_PATIENT in the queue, and exit the program. This operation will always be the last to come, and will only occur once. **Line have the following construct:**

5

3 Output

Print the ID of the Kth patient in the queue. If queue is smaller than K, print 0.

4 Sample input

```
10 2 8
1 222000212 60 1
1 300100000 32 0.2
1 300011002 39 0.03
1 200000000 25 0.2
3 200000000
2 300100000 32 0.03
4
5
```

5 Sample output

```
222000212
```

6 Sample explained

1. 222000212(28.0)
2. 300100000(68.0) <-222000212(28.0)
3. 300011002(146.0) <-300100000(68.0) <-222000212(28.0)
4. 300011002(146.0) <-200000000(75.0)<-300100000(68.0) <-222000212(28.0)
5. 300011002(146.0) <-300100000(68.0) <-222000212(28.0)
6. 300100000(153.0)<-300011002(146.0) <-222000212(28.0)
7. 300011002(146.0) <-222000212(28.0)
8. 222000212

Problem 3

Farthest away from enemies

On the battlefield, the king must be present to manage the war. Therefore, the king is the most important and should be the farthest away possible from the enemies. Assume the battlefield is a M by N grid, and each cell in the grid can be either 1 (a cell occupied by an enemy) or 0 (a cell occupied by friendly forces). The king will be positioned in the 0 cell that is farthest from all 1's.

1 Task

Given a Matrix of size N*M filled with 1's and 0's, the task is to find the maximum distance from a 0-cell to its nearest 1-cell.

Note: Only horizontal and vertical movements are allowed in the matrix. Note: If the matrix is filled with only 0's or only 1's, return -1.

2 Input

The first line is a pair of space separated integers:

Number of rows N - $1 \leq N \leq 2000$.

Number of columns M - $1 \leq M \leq 2000$.

The following N lines will have M space separated values of 0 and 1.

3 Output

The farthest distance from a 0 to the closest 1. If the matrix is filled with only 0's or only 1's, return -1.

4 Sample input

```
3 3
1 0 0
0 0 0
0 0 0
```

5 Sample output

4

6 Sample explained

Cell number (2, 2) is at the farthest distance of 4 cells from the only 1-cell (0, 0).

Theoretical questions

1. Given an unsorted array A with n different elements and two natural numbers i and j where i+j < n. The procedure *get(i,j)* returns all the elements between the i-th largest number and i+j largest number (i.e the i, i+1, i+2, … , i+j largest numbers).

    a. Describe the procedure and analyze the time complexity when no preprocessing of the data is allowed. The algorithm needs to be as efficient as possible.

    b. Describe the procedure and analyze the time complexity when a preprocessing phase of the data is allowed, note that i and j are not given at the preprocessing phase. The preprocessing needs to be efficient as possible, and the time complexity of *get(i,j)* is O(j). Describe the preprocessing and the get procedure.

    c. Describe the procedure and analyze the time complexity when a preprocessing phase of the data is allowed, such that i is given at the preprocessing phase. The preprocessing needs to run in time O(n), and the time complexity of *get(i,j)* is O(j log j). Describe the preprocessing and the get procedure.

2. Given the input $x_1, x_2, \dots, x_n \in \{0, 1, \dots, n^2-1\}$ describe an algorithm based on Hashing that returns the element that appears the most in the data, for example if the input is (1,4,3,5,1,7,1) the procedure will return 1. The time complexity has to be $O(n^2)$ in the worst case and O(n) in the average case for all inputs.

3. Given n points on a plane $(x_1, y_1), \dots, (x_n, y_n)$.

    a. Describe a data structure that holds all the points and a procedure *Closer(a, b)*, that will return all the points (x_i, y_i) in the DS that are closer to (0,0) compare to (a, b). The time complexity needs to be O(k + log(n)) where k is the number of points that are closer. Note that there is no time constraints on building the data structure.

    b. Describe a data structure that holds all the points (x_i, y_i) and a procedure Farther(a, b), that will return the number of all the points in the DS that $x_i > a$ and $y_i > b$. The time complexity needs to be $O(\log^2(n))$.

4. Given a graph where all the edges have weight of 1 or 2, describe an efficient algorithm that finds the shortest path from a start node s to all other nodes.

5. Given a graph G with n nodes and m non-negative weighted edges.
    a. Describe an algorithm that calculates the shortest path from a start node s to all other nodes by the value of the multiplication of the edges weight.
    b. Given a node v in the graph, describe an efficient algorithm that finds the shortest circle that contains the node v.