Ben-Gurion University

Faculty of Engineering Sciences

Department of Electrical and Computer Engineering

Introduction to algorithms and data structures

Home assignment #1

15/5/22

Semester B / 2022

Instructions

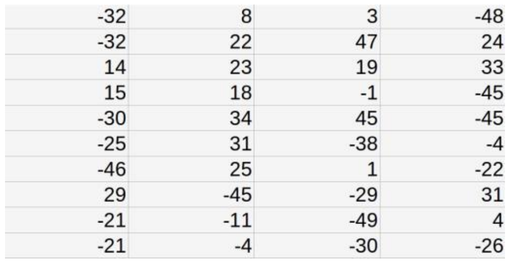
These are the guidelines for submitting this home assignment.

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 two 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 program will print the solution to the screen. For floating point answers, print only the first two digits after the point.
7. 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.
8. The assignment is in pairs. Each pair must submit only one file with the ID numbers. Deadline is in moodle website.
9. The inputs are entered by the user. You can use redirect values from a text file into your program from the command-line (Google it).
10. The first input will be the number of problem you want to solve.
11. A template file is attached. Don't add any other libraries! Don't use scanf_s! Make sure compiles with Visual Studio.

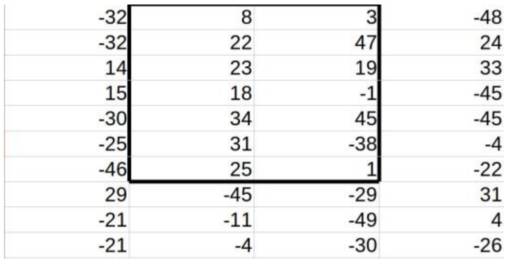
Problem 1

Best bounding box

In image processing a large number of algorithms are used to create a mask from an image based on some feature. The mask can be a binary image, a probability image, or an image field with some metric that proposes where the object is in the image. For example,



-32	8	3	-48
-32	22	47	24
14	23	19	33
15	18	-1	-45
-30	34	45	-45
-25	31	-38	-4
-46	25	1	-22
29	-45	-29	31
-21	-11	-49	4
-21	-4	-30	-26



-32	8	3	-48
-32	22	47	24
14	23	19	33
15	18	-1	-45
-30	34	45	-45
-25	31	-38	-4
-46	25	1	-22
29	-45	-29	31
-21	-11	-49	4
-21	-4	-30	-26

Figure 1.1: Original image (on the right) and with bounding box (on the left).

1 Task

Write a program that finds the best fit for the bounding box, by finding the greatest sum of the pixels within the selected box.

2 Input

The first line contains two space separated integers which are the ROWS (1-200) and COLS (1-300) of the image.
Each of the following ROWS lines will contain COLS integers.

3 Output

The sum of the integers inside the bounding box that produce the largest possible sum.

4 Sample input

```
10 4
-32 8 3 -48
-32 22 47 24
14 23 19 33
15 18 -1 -45
-30 34 45 -45
-25 31 -38 -4
-46 25 1 -22
29 -45 -29 31
-21 -11 -49 4
-21 -4 -30 -26
```

5 Sample output

```
237
```

6 Explanation

The box from row 1 col 2 to row 7 col 3. As seen in the figure above.

Problem 2

Rick’s Destination

Rick tries to find his way across the galaxy. He has built a spaceship that can reach any point in the galaxy by going only in a straight line. The universe is in infinite so the line goes from $[-\infty,\infty]$. The engine is magnificent and **always** accelerates! This means that at the first time step the spaceships travels one space distance, the second time step the spaceship travels two space distance, and at the i-th time step the spaceship travels i space distance.

But Rick blundered, and the engine can only stop once. But luckily the spaceship has an option to go forward or backwards.

Help Rick to get to his destination in the minimum amount of steps.

1 Task

Write a program that gets a target destination, and outputs the number of steps required to reach this target.

2 Input

A target destination. An integer number between $target \in [-40,40]$.

Note - from -35 to 35, brute force approaches will work!

3 Output

The number of steps required to reach the destination.

4 Sample input 1

4

5 Sample output 1

3

6 Explanation

Rick starts at 0, and after rst step he goes to -1, (0, -1). After second step he goes to (-1, 1). After third step he reaches destination (1, 4).

7 Sample input 2

5

8 Sample output 2

```
5
```

9 Explanation

Starting at 0 and performing the following steps: $(0,1) \rightarrow (1,3) \rightarrow (3,6) \rightarrow (6,10) \rightarrow (10,5)$.

Theoretical questions

1. Find the complexity of the function func1(A, B), given that 'A' and 'B' are both n-length arrays. Show your proof.

```
func1(A, B):
        n=length(A)
        RES = new array(n)
        for i = 0 to n-1:
                RES[i]= func2(A[i], B)
        return RES

func2(a, B):
        n = length(B)
        s = 0, e = n-1, res = -1
        while res = -1:
                if (a==B[s])
                        res = s
                if (a==B[e])
                        res = e
                e = e-1
                s = s+1
                if(s>=e)
                        res = s
        return res
```

2. In this task you are asked to implement a two-priority queue using pseudo-code (see question 1 for a pseudo-code example). The two-priority queue is a data structure that is defined to hold a data element <priority, data>. The priority can be either High or Low, and the data is arbitrary. The queue should implement the Enqueue(element) and Dequeue(element) functions in the following way:
   a. If the queue contains elements with High priority then the order of extraction is LIFO among the High priority elements as in the stack pop function, meaning that the last High priority element will be dequeued.
   b. If the queue contains only elements with Low priority then the order of extraction is FIFO as the case of an ordinary queue, meaning that the first Low priority element will be dequeued.

   Answer the following sections.

   a. Show how to implement the two-priority queue with two stacks without time restrictions.
   b. Show how to implement the two-priority queue with two stacks where each Enqueue and Dequeue operation are done in amortized time of O(1).

3. In this task you are asked to implement a data structure that holds Strings and allows the following functions:
    a. Create ($X_1, \ldots, X_n$) – which creates the data structure with time complexity of O(n).
    b. Insert(X) – which insert a new string to the structure with time complexity of O(log(n)).
    c. Del_Median() – which extracts the String in the $\lceil n/2 \rceil$ position, in lexicographic order with time complexity of O(log(n)). Assume that each comparison between Strings takes O(1).