Homework #2

Green Correction Date: 2023/10/25 0:00
Pink Correction Date: 2023/10/23 19:00
Orange Correction Date: 2023/10/22 23:00
Blue Clarification Date: 2023/10/21 11:00
Red Clarification Date: 2023/10/19 16:30
Due Time: 2023/11/03 23:59
Contact TAs: `ada-ta@csie.ntu.edu.tw`

Instructions and Announcements

- There are **four programming problems** and **two hand-written problems**
- **Programming.** The judging system is located at `https://ada-judge.csie.ntu.edu.tw`. Please log in and submit your code for the programming problems (i.e., those containing "Programming" in the problem title) by the deadline. NO LATE SUBMISSION IS ALLOWED.
- **Hand-written.** For other problems (also known as the "hand-written problems"), you should upload your answer to **Gradescope**. For each sub-problem, please label (on Gradescope) the corresponding pages where your work shows up, **or you may get some penalty for each problem you did not label** . NO LATE SUBMISSION IS ALLOWED.
- **Collaboration policy.** Discussions with others are strongly encouraged. However, you should write down your solutions **in your own words**. In addition, for **each and every** problem you have to specify the references (e.g., the Internet URL you consulted with or the people you discussed with, or **screenshot of the full conversation** if you are using GPT) on the **first** page of your solution to that problem and tag the corresponding page in Problem 0 on Gradescope. You may get zero points due to the lack of references.
- **Tips for programming problems.** Here are some C++ only tips. For this homework, you might want to check out sections about cin/cout, vectors, pairs, references, and range for. Reference is IMHO the most important since it rids you of C pointers. (Yeah!!!)
- **Instruction on explaining your algorithm with pseudocode.** If you decide to use pseudocode to describe your algorithm, please try to make your pseudocode human-readable (not C/C++ compilable), and your pseudocode should be less than 30 lines. Pseudocode without explanation or not human-readable may lead to a deduction of points for the corresponding problem. You can check out the textbook for some examples of easy-understanding pseudo code.

Problem 1 - Socializing (Programming) (10 points)

Problem Description

As a student, Grouper lives a simple campus life. Each day, it arrives at school, chats with trees in the morning, participates in some events in the afternoon, and goes home. The events may require Grouper to socialize with others.

Grouper's **social energy** can be described as an integer S (possibly negative). At the beginning of each day (before any events), Grouper may increase or decrease S by 1 through chatting with trees (the ones performing photosynthesis, not the rotating and balancing ones), or leave S unchanged.

There are N days in the new semester, and the **social index** of the day i can be described as an integer a_i (possibly negative), representing the social effort Grouper has to spend on the events that day. If S, Grouper's social energy, is equal to x on the i-th day, then it gets $x \cdot a_i$ points of **happiness** that day.

Grouper's social energy S starts with 0 at the beginning of day 1. To avoid losing itself, Grouper would like to have S to be 0 at the end of day N. What is the maximum sum of happiness score Grouper can get?

Input

The first line of each input file contains an integer T, representing the number of testcases.

For each testcase, there are two lines. The first line contains a single integer N representing the number of days in the semester. The second line contains N integers separated by whitespaces, representing a_i for $i \in [1, N]$.

Constraints:

- $1 \le N \le 10^6$
- $\sum N \le 10^6$
- $|a_i| \le 10^6$

Output

For each testcase, output a line containing a single integer, representing the maximum sum of happiness score Grouper can get that semester.

Test Group 0 (0 %)

- Sample input.

Test Group 1 (10 %)

- $a_i \ge 0$

Test Group 2 (30 %)

- $\sum N \le 5000$. (That is, $\sum N \cdot \max |S|$ is not too big.)

Test Group 3 (60 %)

- No additional constraints.

Sample Input 1

```
5
1
7122
2
-334 -556
4
10 -3 -3 -3
5
10 -3 -3 -3 -3
10
-5 8 8 1 7 -9 4 -10 3 -7
```

Sample Output 1

```
0
334
13
16
57
```

Hint

- For the sample inputs, S before each day's events changes as below:
  1. S stays zero.
  2. $S : -1 \to 0$
  3. $S : 1 \to 0 \to -1 \to 0$
  4. $S : 1 \to 0 \to -1 \to -1 \to 0$
  5. $S : 1 \to 2 \to 3 \to 2 \to 1 \to 0 \to -1 \to -2 \to -1 \to 0$
- Consider d_i, the change of S on day i. How does d_i contibute to the total answer?
- Have you had enough with prefix sums? Well, here are some suffix sums. Have fun!

Extra questions for those interested

- Will the maximum sum of happiness score ever be < 0 ? Why or why not?

Problem 2 - XXLee and USB (Programming) (10 points)

Problem Description

Oh no! The IPMI is broken! XXLee has to manually reinstall the systems on all the servers in the server room.

There are N servers in the server room, indexed from 1 to N. The server room can be viewed as a 2D plane, and the i-th server is located at (x_i, y_i) on the plane.

To complete the reinstallation, XXLee has to plug into each server a USB drive containing the required ISO image. Additionally, XXLee has to obey the following rules:

- Initially, XXLee has two USBs located at $(0, 0)$.
- At the i-th step, XXLee moves one of the USBs to the i-th server, where $i \in [1 \cdots N]$.
- In the end, XXLee moves both USBs back to location $(0, 0)$.

As USBs are heavy, XXLee needs to consume $(x_j - x_i)^2 + (y_j - y_i)^2$ cups of bubble tea before moving one USB from (x_i, y_i) to (x_j, y_j). What is the minimum number of cups of bubble tea XXLee needs to consume before completing the reinstallation?

Input

The first line of each input file contains an integer T, representing the number of test cases.

For each test case, the first line contains a single integer N representing the number of servers.

The next N lines contain two integers separated by a single space, the two number on the i-th line represents x_i and y_i respectively.

Constraints:

- $1 \le N \le 5000$
- $\sum N \le 5000$
- $|x_i|, |y_i| \le 10^6$

Output

For each testcase, output a single integer representing the minimum number of cups of bubble tea XXLee needs.

Test Group 0 (0 %)

- Sample input.

Test Group 1 (10 %)

- $y_i = 0 \ \forall i \in [1, N], x_i < x_{i+1} \ \forall i \in [1, N-1]$

Test Group 2 (60 %)

- $\sum N \le 500$

Test Group 3 (30 %)

- No additional constraints.

Sample Input 1

```
3
3
1 0
2 0
3 0
6
4 1
4 -1
3 1
3 -1
2 1
2 -1
4
1000000 1000000
-1000000 -1000000
1000000 -1000000
-1000000 1000000
```

Sample Output 1

```
12
44
16000000000000
```

Hint

- In the sample input:
  1. It's best to just use one USB and leave another one at $(0, 0)$.
  2. Moving the nearest USB to the next server isn't always the best strategy.
  3. Beware of integer overflow!

Extra questions for those interested

- This problem can be solved with $O(N)$ memory. How?

Problem 3 - Kiwi's Kiwi Factory (Programming) (15 points)

Problem Description

***Kiwi** enrolled in a course named Object Oriented Programming. As a hard-working student, he decided to study some related topics before this semester started. He learned about a well-known design pattern - the factory method pattern, and decided to implement it by starting up a Kiwi factory. After half a semester, **Kiwi**'s Kiwi Factory became the largest kiwi factory in ADA Kingdom.*

Kiwi's Kiwi Factory produced N cans of kiwis this month. The i-th can has a capacity of c_i. Some cans are empty, while others have kiwis in them. **Kiwi** has a non-negative integer sequence $s_1, s_2, \ldots, s_N$. For $1 \le i \le N$, if $s_i = 0$, that means the i-th can is empty. Otherwise, the i-th can has a kiwi of size s_i in it. A kiwi cannot be put into a can with a capacity less than its size, i.e., $\forall 1 \le i \le N,\ c_i \ge s_i$.

After putting all the kiwis into the cans, **Kiwi** realized that the kiwi in the g-th can is a golden kiwi! **Kiwi** thinks his clients may request him to put the golden kiwi into a certain can. To do this, **Kiwi** can perform the following operation any number of times:

1. Choose a can with a kiwi in it.
2. Choose an **empty** can.
3. Move the kiwi from the first chosen can into the empty can.

Of course, the moved kiwi should be smaller than or equal to the capacity of the empty can. After the operation, the first chosen can becomes empty.

Kiwi is curious about how many operations it will take, for $1 \le k \le N$, to move the golden kiwi from its initial can to the k-th can.

For example, let $N = 5, c = [2, 3, 5, 7, 10], s = [0, 2, 3, 7, 5]$ and $g = 2$. If **Kiwi** would like to move the golden kiwi (which is in the 2-nd can initially) into the 4-th can, he can follow the steps:

1. Move the kiwi in the 2-nd can into the 1-st can, now the golden kiwi is in the 1-st can.
2. Move the kiwi in the 3-rd can into the 2-nd can.
3. Move the kiwi in the 5-th can into the 3-rd can.
4. Move the kiwi in the 4-th can into the 5-th can.
5. Move the kiwi in the 1-st can (which is the golden kiwi) into the 4-th can.

Input

The first line contains a positive integer T, representing the number of testcases.

For each testcase:

The first line contains two positive integers N, g, representing how many cans **Kiwi**'s Kiwi Factory made, and the kiwi in the g-th can is the golden kiwi.

The second line contains N positive integers $c_1, c_2, \ldots, c_N$, representing the capacity of each can.

The third line contains N non-negative integers $s_1, s_2, \ldots, s_N$, representing the size of the kiwi in each can. $s_i = 0$ means that the i-th can is empty.

- $2 \le N \le 5 \times 10^5$
- $\sum N \le 5 \times 10^5$
- $1 \le g \le N$
- $\forall 1 \le i \le N,\ 1 \le c_i \le 10^9$
- $\forall 1 \le i \le N,\ 0 \le s_i \le c_i$
- $\exists 1 \le i \le N,\ s_i = 0$, that is, at least one of the kiwi cans is empty at first.
- $s_g > 0$

Output

For each testcase, output N integers $t_1, t_2, \ldots, t_N$ in one line, representing the minimum number of operations required to move the golden kiwi from its initial can to the i-th can is t_i. If it is impossible to move the golden wiki to the i-th can, output t_i as -1.

Test Group 0 (0 %)

- Sample Input

Test Group 1 (10 %)

- $\sum N \le 500$

Test Group 2 (30 %)

- $\sum N \le 10000$

Test Group 3 (60 %)

- No additional constraints

Sample Input 1

```
3
5 2
2 3 5 7 10
0 2 3 7 5
10 4
1 1 3 5 6 7 7 8 9 10
0 1 0 4 5 3 7 6 9 9
5 3
4 8 7 3 6
3 7 6 0 4
```

Sample Output 1

```
1 0 3 5 4
-1 -1 -1 0 3 2 3 3 -1 -1
-1 5 0 -1 3
```

Hint

- An empty can with what property is more useful?
- When we are finding "a more useful can", can we know some t_i at the same time?
- The modified version of this problem may not be much harder than the original one: There are Q queries, where each query gives g, k. Please answer how many operations it will take to move the kiwi initially in the g-th can to the k-th can for each query.

Problem 4 - Alexander and Forest (Programming) (15 points)

Problem Description

Alexander won a lottery from Juice Corporation. The chief of Juice Corporation, Papaya, had heard that Alexander loves forests (which was fake news, as Alexander actually loves one-dollar coins). Papaya then decided to change the top prize of the lottery from one trillion dollars to a land of size 3 meter $\times$ n meter, which was suitable for creating a forest.

When Alexander heard the news about the rearrangement of the prizes, he was very upset about the loss of his precious one-dollar coins. Soon, he noticed that although he could not obtain one-dollar coins directly, he could plant some trees and sell the wood to get coins.

However, some of the land was not fertile enough to grow trees. Also, if any two trees were too close to each other, the growth rate of both trees would be limited. In order to acquire as many coins as possible, Alexander needed to find the maximum number of trees he could plant given these conditions.

Input

The first line contains two integers n, k representing the width of the land and the constraint of distance, respectively. Each of the next 3 lines contains a string S_i of length n that consists of either character `'#'` or `'.'`. The j-th character of S_i represents the condition of cell (i, j) of the land. Cell (i, j) is usable if $S_{i,j}$ = `'.'` and unusable if $S_{i,j}$ = `'#'`.

- $1 \le n \le 10^7$
- $5 \le k \le n^2 + 4$
- $1 \le i \le 3$
- $1 \le j \le n$
- $S_{i,j} \in \{$`'.'`, `'#'`$\}$

Output

Output the maximum of the number of trees Alexander can plant under the given constraints.

- You can only plant one tree in the middle of a usable cell. You cannot plant any tree in an unusable cell.
- The square of the distance of any two trees should be larger than or equal to k.

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (10 %)

- $\forall i, j,\ S_{i,j}$ = `'.'`

Test Group 2 (10 %)

- $k \ge 10$

Test Group 3 (80 %)

- No additional constraints.

Test Group 4 (0 %, for those who are interested in general case)

- $1 \le k \le n^2 + 4$

Sample Input 1

```
5 20
#....
.....
....#
```

Sample Output 1

```
2
```

Sample Input 2

```
5 5
##.##
#####
.....
```

Sample Output 2

```
3
```

Sample Input 3

```
3 8
..#
...
..#
```

Sample Output 3

```
1
```

Problem 5 - Nathan at Small Circle Village (Hand-Written) (25 points)

Note: In all the following problems, you should briefly explain the correctness of your algorithm and explain its time complexity. You are allowed to use pseudocode to describe your algorithm, but it should be less than 30 *lines.*

Nathan loves bubble tea

Nathan lives in Small Circle Village, a village that is very famous for bubble tea. Every villager in Small Circle loves to drink bubble tea, and Nathan is no exception. As an addict to bubble tea, Nathan would drink many cups of bubble tea every day. In Small Circle Village, there are several bubble tea shops selling different bubble teas. This gives Nathan a thought-provoking problem every day: how should he buy bubble tea for himself to drink?

One day, while thinking about the problem, Nathan walked past a familiar street of Small Circle Village. He remembered that there were N bubble tea shops arranged in order on the street. The bubble tea sold by the i-th shop cost c_i dollars and would give h_i happiness to Nathan after drinking it based on experience. Nathan couldn't wait to drink bubble tea, so he gave up thinking and decided to buy all bubble tea on the street. (You may suppose that Nathan would drink at most one cup of bubble tea from each shop for all the problems below. Also, you may assume that all c_i and h_i are positive integers.)

(a) (3 points)

Assume that Nathan had M dollars with him, given the cost of each bubble tea $c_1, c_2, \dots c_N$ and the happiness Nathan can obtain from each bubble tea $h_1, h_2, \dots h_N$. Please design an $O(NM)$ algorithm to find out the maximum happiness Nathan can obtain after buying some bubble tea to drink.

Nathan and his good days

Nathan had luckily won a lottery recently, his good days finally came! With this unexpected windfall, the cost of bubble tea was no longer a problem for him! However, Fysty, being a friend of Nathan, advised him against excessive spending on bubble tea and suggested that he limit himself to drinking no more than K cups every day. Trusting in Fysty's wisdom, Nathan resolved to exercise self-restraint and needed Fysty's advice.

(b) (2 points)

Given the happiness that the N cups of bubble tea $h_1, h_2, \dots, h_n$, please design an $o(N^2)$ algorithm to find the maximum happiness Nathan can obtain after buying at most K cups of bubble tea to drink. That is, find the sum of the top K largest elements in $\{h_i\}$.

Nathan feeling EMO

Nathan had been overwhelmed with the assignments from Numerical Analysis of Security Applications and Digital System Analysis. The pressure was so intense that he became frustrated and would easily feel EMO if he missed consecutive L cups of bubble tea. More formally, he would feel EMO if he didn't buy any bubble tea from $k, k+1, ..., k+L-1$-th shops for some k.

In this part, you can obtain the points from both subproblems **(c - 1)** and **(c - 2)** by only writing a solution to subproblem **(c - 2)**. Please tag your all your work on subproblem **(c - 1)** and subproblem **(c - 2)** in **Problem 5(c)** on Gradescope.

(c - 1) (4 points)

Given $N, K, L, \{h_i\}$, where

- N is the number of bubble tea shops
- K is the limit on the number of bubble tea
- L is the limit on Nathan that he need to buy a cup of bubble tea in every L consecutive shops
- $\{h_i\}$ is the N happiness value each bubble tea $h_1, h_2, \ldots h_N$ provide.
- You can assume that Nathan has at least one way to buy bubble tea, meaning there is no need to handle unsolvable edge cases.

Please design an $O(NKL)$ algorithm to find the maximum happiness Nathan can obtain after buying at most K cups to drink without feeling EMO.

(c - 2) (4 points)

Please design an $O(NK)$ algorithm to find the maximum happiness Nathan can obtain after buying at most K cups to drink without feeling EMO.

Nathan and Fysty

After Nathan successfully bought K cups of bubble tea without feeling EMO, he suddenly discovered that after he became frustrated, the maximum happiness he could obtain had decreased compared to the previous situation. This heart-breaking fact almost overwhelmed him. Fortunately, his friend Fysty, who specializes in swaps, can "swap" the two bubble tea shop to increase Nathan's happiness.

Fysty can swap the position of two bubble tea shop before Nathan start buying bubble tea. That is, given the happiness of N bubble tea provides $h_1, h_2, \ldots h_n$, Fysty can choose two indexes x, y and swap the values of h_x, h_y. However, since the relationship between Nathan and Fysty was not that good, Fysty is willing to do this operation for at most T times.

(d) (4 points)

given $N, K, L, \{h_i\}, T$, where

- N is the number of bubble tea shops
- K is the limit on the number of bubble tea
- L is the limit on Nathan that he need to buy a cup of bubble tea in every L consecutive shops
- the N happiness value each bubble tea provides $h_1, h_2, \ldots h_N$ provide.
- $T = 1$, that is Fysty is only willing to do the swap one time.

- You can assume that Nathan has at least one way to buy bubble tea, meaning there is no need to handle unsolvable edge cases.

Design an $O(NKL)$ algorithm to find the maximum happiness Nathan can obtain after buying at most K cups to drink with Fysty's help.

In this part, you can obtain the points from both subproblems **(e - 1)** and **(e - 2)** by only writing a solution to subproblem **(e - 2)**. Please tag your all your work on subproblem **(e - 1)** and subproblem **(e - 2)** in **Problem 5(e)** on Gradescope.

(e - 1) (5 points)

Given $N, K, L, T, \{h_i\}$, where

- N is the number of bubble tea shops
- K is the limit on the number of bubble tea
- L is the limit on Nathan that he needs to buy a cup of bubble tea in every L consecutive shops
- T is the number of swaps Fysty is willing to do
- $h_1, h_2, \ldots h_N$ is the happiness that the N bubble tea provides, **satisfying** $h_i \neq h_j$ **if** $i \neq j$
- You can assume that Nathan has at least one way to buy bubble tea, meaning there is no need to handle unsolvable edge cases.

Assume H is the maximum happiness Nathan can obtain from buying bubble tea without the limit L. Design an $O(NL^2T + N \log N)$ algorithm to tell if it is possible for Nathan to obtain the same happiness H with the help (swaps) from Fysty.

(e - 2) (3 points)

Assume H is the maximum happiness Nathan can obtain from buying bubble tea without the limit L. Design an $O(NLT + N \log N)$ algorithm to tell if it is possible for Nathan to obtain the same happiness H with the help (swaps) from Fysty.

Problem 6 ADA FTP server (25 points)

Note: In all of the following problems, you need not only to construct the algorithm but also to prove the correctness. Proving the correctness includes proving the complexity.

Download homework quickly

To enhance students' homework experience, we have implemented a robust ADA FTP server, allowing students to download their assignments. This server is tasked with serving n students by transmitting files to them. Each student i has a network download speed of d_i and aims to download the file f_i with a file size of s_i. The ADA FTP server's network transmitting speed is denoted as u. The download speed of student i is $\min(d_i, u)$, which is the bottleneck between the FTP server and the student.

Let c_i represent the completion time for student i to finish downloading. Suppose the ADA FTP server's service begins at timestamp a, and student i completes downloading at timestamp b. The completion time for student i is denoted as $c_i = b - a$. For simplicity, we set $a = 0$, meaning that the ADA FTP server starts at timestamp 0.

Given the students' eagerness to complete their homework promptly, we aim to design an algorithm to minimize the average completion time, denoted as $c_{avg} = \frac{1}{n}\sum_{i=1}^{n} c_i$. It's important to note that the ADA FTP server can only serve one student at a time.

(a) (3 points)

Now ADA FTP server transmitting speed $u = 6$. Suppose there two student with $d_1 = 3$, $d_2 = 15$ and $s_1 = 36,\ s_2 = 30$.Compute the c_1, c_2 and the average completion time c_{avg} if (1) the ADA FTP server first serves student 1 (2) the ADA FTP server first serves student 2

(b) (4 points)

Design a greedy algorithm that minimizes c_{avg} in $O(n \log n)$ time complexity. In this case, once the server starts transmitting file f_i, it cannot be interrupted and cannot serve other students until it completes transmitting file f_i.

(c) (4 points)

We have upgraded the ADA FTP server, allowing it to suspend the current file transmission and resume the previous transmission at a later time. For example, if we have a file f_i with size $s_i = 5$, the server can first transmit 2 units to the student, then switch to transmit another file f_i', and later resume transmitting the remaining 3 units of f_i. Design a greedy algorithm that minimizes c_{avg} in $O(n \log n)$ time complexity.

(d) (4 points)

Continuing from problem (c), let's consider a scenario where all students are not connected to the ADA FTP server initially. Instead, each student c_i connects to the server at a specific time t_i. Before time t_i, the ADA FTP server is unaware of the file f_i and cannot transmit any file to c_i. Design a greedy algorithm that minimizes c_{avg} with a time complexity of $O(n \log n)$.

Arrange the ADA FTP server

Now, one ADA FTP server is insufficient to cater to all students' needs. Hence, we have established N ADA FTP servers. Each ADA FTP server, denoted as server i, operates at a specific transmission speed u_i and is positioned at location x_i. Unfortunately, these servers emit strong electromagnetic waves, leading to malfunctions in nearby servers. We have a distance judgment function denoted as F. If $F(i, j) = True$, it implies that servers i and j will interfere with each other.

Our objective is to identify a subset of ADA FTP servers to activate, aiming to maximize the total transmission speed. In other words, we want to find a set $A \subseteq \{1, \dots, N\}$ such that for any pair (a_i, a_j) where $a_i, a_j \in A$, the condition $F(a_i, a_j) = False$ holds. The goal is to maximize the sum of transmission speeds, given by $\sum_{a_i \in A} u_{a_i}$.

It is guarantee that $x_i < x_{i+1}$, $i \in \{1, 2 \dots N - 1\}$. That is, the sequence is strictly increasing.

(e) (5 points)

The ADA FTP server will only affect neighbor ADA FTP server. That is $F(a, b) = (|a - b| \leq 1)$. Design an algorithm that runs in $O(n)$ time complexity, and provide a proof of its correctness.

(f) (5 points)

Now, server i's electromagnetic waves can affect other servers within a radius of r_i. In other words, each ADA FTP server is associated with a value r_i. That is $F(a, b) = (|x_a - x_b| \leq \max(r_a, r_b))$. Design an algorithm that runs in $O(n \log n)$ time complexity and provide a proof of its correctness. In this problem, It is guarantee that $r_i < r_{i+1}$, $i \in \{1, 2 \dots N-1\}$. That is, the sequence is strictly increasing.