Homework #2

Red Correction Date: 2021/11/2 18:00

Due Time: 2021/11/9 14:20

Contact TAs: `ada-ta@csie.ntu.edu.tw`

Instructions and Announcements

- There are **four programming problems** and **two hand-written problems**.
- **Scoring policy.** Although the total score in this homework is 102, you'll get 100 if your original score exceeds 100.
- **Programming.** The judge system is located at `https://ada-judge.csie.ntu.edu.tw`. Please login 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** as demonstrated in class. 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 URL of the website you consulted or the people you discussed with) on the first page or comment in code of your solution to that problem. You may get zero points due to the lack of references.
- **Tips for programming problems.** Since the input files for some programming problems may be large, please add
  - `std::ios_base::sync_with_stdio(false);`
  - `std::cin.tie(nullptr);`

  to the beginning of the main function if you are using `std::cin`.

Problem 1 - Glass Bridge 2.0 (Programming) (12 points)

Problem Description

Improved from the fifth game in *Squid Game (2021)*, Glass Bridge 2.0 is another deadly game played on an $n \times m$ grid, where contestants have to reach the goal cell by jumping from a square glass to another. To better describe the environment, we use (i, j) to denote the cell on the i-th row and on the j-th column. The starting cell is located on $(1, 1)$ while the goal one is located on (n, m). For each of the remaining cells, the cell (i, j) satisfies **exactly one** of the following states:

- The cell is placed with a square glass with a given risk value, since the glass may be smashed when you step on it. To make the game more exciting, there may be a bundle of banknotes on that piece of glass. Considering the revenue and risk, the **expected profit** could be modeled as $a_{i,j}$ if one jumps onto the glass.
- The cell is empty, since a previous contestant has smashed the glass there. In other words, the cell is now impassable – if one steps on this cell, she/he will fall into a horrible abyss immediately.

You, as the next contestant to move on, are going to reach the goal cell by jumping from one cell to another. A jump from (x, y) to (x', y') is valid if **all** of the following conditions hold:

- The cell (x', y') is placed with a square glass.
- $x \leq x' \leq n$
- $y \leq y' \leq m$
- $1 \leq (x' - x) + (y' - y) \leq k$

Currently standing on the starting cell, $(1, 1)$, you would like to first realize if it is possible to reach the goal cell with those currently existing glasses. If so, please come up with a jumping path such that S, the sum of expected profit over all cells on your path, is maximized.

Input

The first line of the input contains 1 integer T – the number of test cases. Then, in each test case:

- The first line is an empty line. This may make it more friendly for you to distinguish between test cases with your naked eyes.
- The second line contains 3 integers n, m, k, denoting the size of the grid and your jumping ability.
- Then, n lines follow, the i-th of which contains m strings or integers, the j-th of which is:
  - 0, if $(i, j) = (1, 1)$ or (n, m).
  - $a_{i,j}$, if the cell (i, j) is placed with a square glass.
  - "`X`" (without quotation marks), if the cell (i, j) is empty.

Constraints

- $1 \leq T \leq 100$
- $1 \leq n, m \leq 10^6$
- $2 \leq nm \leq 10^6$
- $\sum nm \leq 10^6$
- $1 \leq k \leq n + m - 2$
- $-456, 456, 456, 456 \leq a_{i,j} \leq 456, 456, 456, 456$
- You may obtain up to 10.2 points from this problem even if your solution is not fully accepted.

Test Group 0 (0 %)

- Sample Input

Test Group 1 (15 %)

- $n, m \leq 30$
- $k = 1$

Test Group 2 (25 %)

- $n, m \leq 30$

Test Group 3 (45 %)

- $\Big(n, m \leq 30\Big)$ or $\Big(T = 1$ and $n, m \leq 400\Big)$

Test Group 4 (15 %, Bonus Subtask)

- No additional constraint

Output

For each test case:

- In the first line, please output a string s. $s =$ `Passable` if it is possible for you to reach the goal cell with those currently existed glasses. Otherwise, $s =$ `Impassable`.
- If $s =$ `Passable`, you have to specify your jumping path. In the second line, please output S, the maximized sum of the expected profit. In the third line, please output L, the length (including the starting and goal cells) of your path planned. Then L lines follow, the i-th of which should contain 2 integers, x_i, y_i, denoting the i-th coordinate in your path. Note that you **don't** have to minimize the length of your path. If there are multiple solutions, you may print any.

Hint

1. Since each input includes several independent test cases, please carefully clear all results of the current test case before dealing with the next one.
2. In Sample Input 1, the first and second test cases are exactly the same. However, the Sample Output comes up two different paths to answer the same test case. This is just a demonstration on the special scoring method.
3. Test Group 4 may be a challenging subtask. This could be viewed as a bonus subtask since you are still able to obtain 50 points from programming problems even without solving it. Nevertheless, you are encouraged to give it a try because it's solvable without any advanced data structure or algorithm – what you need is just an ingenuity!

Sample Input 1

```
5

3 3 1
0 0 0
-1 0 -1
X -1 0

3 3 1
0 0 0
-1 0 -1
X -1 0

3 3 1
0 -7 X
X X 20
0 10 0

3 3 2
0 -7 X
X X 20
0 13 0

5 7 9
0 X X X X X X
X X X X X X X
X X X -456456456456 X X X
X X X X X X X
X X X X X X 0
```

Sample Output 1

```
Passable
-1
5
1 1
1 2
2 2
3 2
3 3
Passable
-1
5
1 1
1 2
2 2
2 3
3 3
Impassable
Passable
13
4
1 1
1 2
2 3
3 3
Passable
-456456456456
3
1 1
3 4
5 7
```

Behind the Scene

Some wise words from YP, one of the lead TAs, upon knowing the Test Group 4 in this problem:

Problem 2 - ADA Removal Game (Programming) (15 points)

Problem Description

Consider an array A containing N elements. For each operation, you can choose 2 or 3 contiguous elements, whose pairwise greatest common divisor is greater than 1, and remove them from the array. After removal, the remaining parts of the array are concatenated. Meanwhile, you earn points by doing this operation, and the point is calculated by adding up the greatest common divisor of each pair of adjacent elements you removed.

Formally, consider an array A of length N, you can choose two integers i, k such that $1 \leq i \leq i+k-1 \leq N, k = \{2, 3\}$ where $\gcd(A_x, A_y) > 1, \forall i \leq x \leq y \leq i+k-1$. Remove $A_i, \ldots, A_{i+k-1}$ from the array and get $\sum_{p=i}^{i+k-2} \gcd(A_p, A_{p+1})$ points. After the operation, the new array becomes $A_1, \ldots, A_{i-1}, A_{i+k}, \ldots, A_N$ of length $N - k$.

You can perform the above operation multiple times until the array is empty. Please find the maximum total points you can receive to eliminate the entire array, and determine whether it is possible to do so.

For example, array $[2, 3, 12, 6, 4]$ can be totally eliminated by removing $(3, 12, 6)$ and $(2, 4)$ in order, which gets 11 points. Or, it can also be removed by $(3, 12)$ and $(2, 6, 4)$, which gets 7 points. There is no other way to remove the entire array. Hence, the answer should be the greater one 11.

Input

The first line contains an integer indicating N, where $2 \leq N \leq 500$.

The second line contains N space-separated positive integers a_i, where $2 \leq a_i \leq 10^9$.

Test Group 0 (0 %)

- Sample Input

Test Group 1 (10 %)

- $N \leq 10$

Test Group 2 (20 %)

- $2|a_i, \forall i$

Test Group 3 (30 %)

- $N \leq 100$

Test Group 4 (40 %)

- No other constraints.

Output

Please output an integer indicating the maximum points. If it is impossible to eliminate the entire array, output -1.

Sample Input 1

```
5
2 3 12 6 4
```

Sample Output 1

```
11
```

Sample Input 2

```
5
10 9 3 10 10
```

Sample Output 2

```
23
```

Sample Input 3

```
6
2 3 6 12 6 4
```

Sample Output 3

```
13
```

Sample Input 4

```
5
2 3 8 6 4
```

Sample Output 4

```
-1
```

Problem 3 - IOICamp (Programming) (15 points)

Problem Description

IOICamp is a competitive programming training camp held by NTU CSIE students. Xiao Feng, an IOICamp staff, is full of enthusiasm to dedicate himself to making it better.

One day, Baluteshih, the general coordinator of IOICamp, assigned N tasks to Xiao Feng. Concretely, the i^{th} task consists of x_i units of work. Each unit of work costs one day for Xiao Feng to complete. Besides, Xiao Feng can not do two different tasks in one day simultaneously. Because of Baluteshih's requirements, Xiao Feng can only work on the i^{th} task between ${s_i}^{\text{th}}$ day and ${e_i}^{\text{th}}$ day (inclusive).

To encourage Xiao Feng to finish as much work as possible, Baluteshih gives him some benefits. For the i^{th} task, if Xiao Feng completes y_i units of work, he will get $p_i \cdot y_i$ bonus from Baluteshih. Note that Xiao Feng can still get the bonus from Baluteshih even if he didn't finish all units of work in the one task. Please help Xiao Feng to calculate the maximum bonus he could receive from his kindly boss, Baluteshih :).

Input

The first line of the input contains only one integer N, denoting the number of tasks assigned to Xiao Feng.

In the following N lines, the i^{th} line contains four integers s_i, e_i, x_i, p_i, describing the information of the i^{th} task.

constraints

- $1 \leq N \leq 3000$.
- $1 \leq s_i \leq e_i \leq 10^9$.
- $1 \leq x_i \leq e_i - s_i + 1$.
- $1 \leq p_i \leq 10^9$.

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (15 %)

- $s_1 \leq s_2 \leq \ldots \leq s_N$.
- $e_1 \leq e_2 \leq \ldots \leq e_N$.
- $p_i = 1, \forall 1 \leq i \leq N$.

Test Group 2 (40 %)

- $p_i = 1, \forall 1 \leq i \leq N$.

Test Group 3 (15 %)

- $s_1 \leq s_2 \leq \ldots \leq s_N$.
- $e_1 \leq e_2 \leq \ldots \leq e_N$.
- $x_i = e_i - s_i + 1, \forall 1 \leq i \leq N$.

Test Group 4 (30 %)

- No other constraints.

Output

Print one integer denoting the maximum bonus Xiao Feng could receive if he arranges the work optimally.

Sample Input 1

```
3
1 3 2 1
1 5 1 1
2 4 1 1
```

Sample Output 1

```
4
```

Sample Input 2

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

Sample Output 2

```
55
```

Sample Input 3

```
5
9 10 1 5
5 15 6 7
4 6 2 8
1 6 1 3
3 9 1 1
```

Sample Output 3

```
67
```

Sample Input 4

```
10
317828572 952962709 511194031 474210
139065667 594136128 184836056 727043
145449199 856665845 135232964 221941
185367317 719253355 508496356 303732
286924029 536237215 174723858 743784
448407424 788782769 294918233 970051
128701901 369779350 133590454 996886
268148730 724234276 442825804 255091
658359136 999211180 190588357 715619
114934339 328552693 120729904 373197
```

Sample Output 4

```
741483180481768
```

Hint

1. It might be useful to solve **Test Group 2** before solving the rest of the problem. If you encounter some difficulties, please try to solve it first.
2. It is recommended to use C++ Standard Template Library (STL) data structures, such as `std::stack`, `std::queue`, `std::priority_queue`, `std::list`, `std::vector`, and `std::set`. They can reduce your coding complexity.

Problem 4 - Restricted Candies (Programming) (10 points)

Problem Description

The definition of an alternating sequence is slightly different from that of in Homework 1.

Baluteshih brings N candies to his friend, Waynetu. Those candies are lined on the table. Each candy has its own sweetness indicated by an integer. The sweetness of the candies from left to right are a_1, a_2, ..., a_N, respectively.

Baluteshih assigns Waynetu an interesting mission. He asks Waynetu to remove some candies from the table, so that the remaining candies on the table are *alternating*. Formally, we say that a sequence of candies with sweetness b_1, b_2, ..., b_k is *alternating* if $\boldsymbol{b_i \times b_{i+1} < 0}$ holds for all $1 \leq i < k$.

With the aforementioned rule, Waynetu hopes to maximize the sum of the sweetness of the candies on the table.

However, Ltf0501 comes and interrupts the mission. Because he wants to make this mission more interesting, he gives Waynetu an additional restriction, that is, **Waynetu needs to leave exactly k candies on the table**. Moreover, for each k between 1 to N, Waynetu needs to determine whether it is possible to leave exactly k candies on the table; if so, Waynetu also need to maximize the sum of the sweetness for that k value.

Please help Waynetu find the maximum possible sum of exactly k remaining candies' sweetness for each k between 1 to N.

Input

The first line contains two integers T, representing the number of test cases, and $flag$ ($flag \in \{0, 1\}$), which will be described in the output section.

Each test case includes two lines: the first line contains an integer N ($1 \leq N \leq 10^5$), and the second line contains N integers a_1, a_2, ..., a_N ($|a_i| \leq 10^9, \boldsymbol{a_i \neq 0}$).

It is guaranteed that the sum of N within the same input file does not exceed 10^5.

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (20 %)

- $flag = 1$, $a_1 > 0$ and $a_N > 0$.
- $\sum N \leq 1000$.

Test Group 2 (20 %)

- $\sum N \leq 1000$.

Test Group 3 (30 %)

- $flag = 1$, $a_1 > 0$ and $a_N > 0$.

Test Group 4 (30 %)

- No additional constraints.

Output

For each test case, please print N integers in a line, separated by spaces. The i^{th} number represents the maximum possible sum of the sweetness when Waynetu leaves exactly i candies on the table. If it is impossible to leave exactly i candies on the table, just output 0 for the i^{th} number.

If $flag = 1$, then you can get Accepted when you have at least $\lfloor \frac{N}{2} \rfloor$ correct answers for each test cases.

Sample Input 1

```
1 0
5
3 -1 6 -7 4
```

Sample Output 1

```
6 5 8 2 5
```

Sample Input 2

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

Sample Output 2

```
3 0 0
3 1 2 -2
```

Sample Input 3

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

Sample Output 3

```
0
5 0 0
0 0 0 0
```

Hint

1. Since each input includes several independent test cases, please carefully clear all results of the current test case before dealing with the next one.
2. In Sample Output 3, you will get Wrong Answer if $flag = 0$. The Sample Output is just a reminder for the special scoring method.
3. It is recommended to use C++ Standard Template Library (STL) data structures, such as `std::stack`, `std::queue`, `std::priority_queue`, `std::list`, `std::vector`, and `std::set`. They can reduce your coding complexity.

Problem 5 - Toyz's Dog (Hand-Written) (30 points)

You, Asiagodtone, as known as Toyz's dog, notices that someone smelled strange, so you decide to trace him. With your big fat nose, you keep tracking that strange smell. After a long trip, you finally arrive at a smoky place. There is a castle surrounded by a moat, and the only path to get into the castle consists of $N-1$ floating stones in front of the gate. Each stone will sink to the deep when you step on it. You have to choose some stones as a route so that you can go into the castle and then catch the odd-smelling guy. When leaving, you have to go through all the remaining stones to prevent the enemies in the castle from chasing you. That is, you could only step on each stone once and you should let all of them sink to the bottom after leaving the castle.

Subproblem (a)

Assuming that the initial place is S_0 and the castle is located at S_N. The $N-1$ stones are numbered as S_1 to S_{N-1}. When you jump from S_a to S_b, you spend $E(a,b)$ energy. It does **NOT** guarantee that $E(a,b) \leq E(a,c) + E(c,b)$. Note that you are too fat to turn around your body on the stone. For example, $N = 3$, your possible path can be:

$\{S_0 \to S_1 \to S_2 \to S_3 \to S_0\}$

$\{S_0 \to S_1 \to S_3 \to S_2 \to S_0\}$

$\{S_0 \to S_2 \to S_3 \to S_1 \to S_0\}$

$\{S_0 \to S_3 \to S_2 \to S_1 \to S_0\}$.

Now, you want to design a dynamic programming algorithm to find the smallest cost of the entire trip in $O(N^2)$ time complexity. Note that your should let $dp(i,j)$ as the minimal cost of the trip, where your last point on the way you go is S_i and your first point on the way you left is S_j and then minus $E(i,N) + E(N,j)$. Under $dp(i,j)$ state, you should already choose S_0 to $S_{max(i,j)}$ to your path no matter the stone is on the way go or left.

(1) (6%) Please recursively define $dp(i,j)$ and prove your correctness and time complexity.

(2) (6%) With the recursive function you defined in question (1), please solve the problem in $O(N)$ space complexity.

(3) (6%) Please get the optimal path in the same space complexity as question (2).

Subproblem (b)

After the capture, you surprisingly find out that the man is your master, Toyz. However, you still send him to the jail for the justice. Unfortunately, his friend, R-code, helps Toyz escape from the jail and then flee back to his castle. This time, he set a smoke bomb at each stone on the way to the castle. Whenever you stand on a stone S_n, the bomb can be triggered and causes D_n damage on you. Luckily, you find a switch that can turn off all the bomb trap in the castle, so you will be safe when leaving the castle.

Assuming that you have initial Health Points H, when you stand on S_n on the way to castle, your Health Points would be reduced by D_n. Under this additional constraint, please design a dynamic programming algorithm to find the smallest cost of the entire trip, as in the subproblem (a), while keeping your Health Points greater than zero.

(1) (6%) Please modify your definition of dp function to solve the problem in $O(HN^2)$ time.

(2) (6%) Please prove the correctness and the time complexity of your definition.

Problem 6 - Howard's Desiring Order of Courses (Hand-Written) (20 points)

*In problem 6, please **briefly** explain your solution in text. **Do not** use pseudo code, or you will receive penalty.*

Howard Wang, as known as rap god, started his first semester in the CSIE department last autumn. When he was 18, he set a goal for his college life, "Having a girlfriend." However, due to COVID-19 pandemic and lack of courage, he is still not familiar with most of his classmates, let alone making any female friend. L.Y.P., who is his best friend and a natural charmer, gave him an advice, "How about enrolling in courses with the maximum number of girls?". Following the advice, Howard started to arrange his timetable of the upcoming semester. He collected essential information all CSIE courses and marked his preference of them. As Howard's friend, please help him find the **maximum satisfying value** he can reach by arranging a desiring order of those courses.

Assuming that there are n courses in the CSIE department, which are C_1 to C_n. For each course C_i, Howard marked his preference of the course with P_i. Moreover, the maximum satisfying value is defined as the maximum number for that could be formed by **concatenating some of the course preference values (in digits) in any order**. For example, if there are three courses $C_1,\ C_2,\ C_3$ in CSIE, which $P_1 = 2$ *and* $P_2 = 40$ *and* $P_3 = 2$, the maximum satisfying value will be 4022.

Lastly, there are some assumptions as follows, which are used in the following problems.

Assumption 1 *Preference of the course is always a single digit number, that is* $\forall\ 1 \leq i \leq n,\ P_i \in [0,\ 9]$.

Assumption 2 *Preference of the course is always a small integer, that is* $\forall\ 1 \leq i \leq n,\ P_i \in [0,\ 1000]$.

Assumption 3 *The maximum satisfying value should be a multiple of 3.*

Please answer the following problems.

(1) (2%) Given preference value of 6 courses $P = [1,\ 0,\ 9,\ 81,\ 4,\ 12]$, can you find the maximum satisfying value under no assumption and under **Assumption 3**, respectively?

(2) (2%) Please give a $O(n\log(n))$ algorithm to find the maximum satisfying value under **Assumption 1**.

(3) (2%) Please give a $O(n\log(n))$ algorithm to find the maximum satisfying value under **Assumption 2**.

(4) (4%) Please give a $O(n)$ algorithm to find the maximum satisfying value under both **Assumption 1 and 3** (If you get full points on this problem, you will automatically gain full points of (2).)

There are still problems on the next page!

W.H.Y., Howard's another friend, decided to further help him by his rich experience. "How about considering some courses from the college of management? There are more girls and elites!", said W.H.Y.. Howard accepted his advice and also collected information of courses in college of management. To balance his daily life, he also collected n courses from college of management, which are E_1 to E_n. Same as the first time, Howard also marked his preference of each course from college of management E_i with a preference value M_i. For easier understanding, there are now **total** $2n$ **distinct courses from CSIE and college of management, each has** n **courses, which are** C_1 **to** C_n **and** E_1 **to** E_n, respectively. For those courses in CSIE, each has a preference value P_i; for those in college of management, each has a preference value M_i. Same as the first part, you're also asked to find the **maximum satisfying value** he can reach by arranging a desiring order of those courses. In case you forget what is maximum satisfying value, the maximum satisfying value is defined as the maximum number for that could be formed by **concatenating some of the course preference values (in digits) in any order**.

However, the time is limited. Howard can only put **at most total** k **courses in his desiring order** of courses. Moreover, he performed sorting this time, which **preserves the relative order of the courses from the same department**. That is, in the desiring order, the courses from CSIE should have the same relative order as C_1 to C_n, and the courses from management should have the same relative order as E_1 to E_n. To simplify the problem, the preference of courses is always a **single digit number**, that is $\forall\ 1 \leq i \leq n,\ \{P_i,\ M_i\} \in [0,\ 9]$.

For example, if $n = 3$, $k = 4$, the preference value of courses are $P_i = [3,\ 6,\ 4]$, $M_i = [8,\ 5,\ 7]$, the maximum satisfying value will be 8764.

(5) (2%) Assuming $n = 5$, $k = 5$, given preference value of courses $P_i = [3,\ 4,\ 6,\ 5,\ 0]$, $M_i = [9,\ 0,\ 5,\ 8,\ 3]$, can you find the maximum satisfying value?

(6) (8%) Please give a $O(kn^2)$ algorithm to find the maximum satisfying value given P_i and M_i.