

Homework #2

Due Time: 2017/11/09 (Thu.) 17:20

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

Instructions and Announcements

- There are **four programming problems** and **two handwritten problems**.
- **Programming.** The judge system is located at <https://ada-judge.csie.org>. 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.**
- **Handwriting.** For other problems (i.e., those containing “Handwriting” in the problem title), please submit your answers to the instructor at the beginning of the 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 Internet URL you consulted with or the people you discussed with) on the first page of your solution to that problem. You may get zero point for problems due to the lack of references.
- Top-graded solutions/codes may be published as references for your fellow classmates.

Problem 1 - Sum (Programming) (10 points)

Problem Description

In mathematics, a sequence of positive real numbers s_1, s_2, \dots is called *superincreasing* if each element in the sequence is greater than the sum of all previous elements in the sequence:

$$s_{n+1} > \sum_{i=1}^n s_i$$

For example, $(1, 3, 6, 13, 27, 52)$ is a superincreasing sequence, but $(1, 3, 4, 9, 15, 25)$ is not.

Given a superincreasing sequence s_1, \dots, s_n and a positive integer k , please find a subsequence of s_1, \dots, s_n with the sum equal to k .

Input

The first line contains an integer T , indicating the number of test cases. Each test case consists of two lines. The first line contains two integers n, k , and the second line contains n integers indicating s_1, \dots, s_n .

- $1 \leq T \leq 10000$
- $1 \leq s_i \leq 2^{63}$
- $1 \leq k \leq 2^{64} - 1$

Subtask1 (20 %)

- $1 \leq n \leq 10$

Subtask2 (80 %)

- No other constraint.

Output

For each test case, output one integer m , indicating the length of the subsequence. Then one line with m integers, indicating the subsequence. If there are several possible subsequences, you can output any of them. If it's impossible to sum to k , output -1 .

Sample Input

```
3
4 10
1 2 4 8
2 10
1 5
4 15
1 2 4 8
```

Sample Output

```
2
2 8
-1
4
1 2 4 8
```

Note

If you are confusing about the limitation of n , you should think about the definition of *superincreasing*.

Problem 2 - Longest Similar Subsequence (Programming) (10 points)

Problem Description

A *subsequence* is a sequence that can be derived from another sequence by deleting some elements (maybe none) without changing the order of the remaining elements.

A sequence C is a *common subsequence* of two sequences A, B if and only if C is a subsequence of A and C is also a subsequence of B . The longest common subsequence (LCS) of two sequences A, B is the longest sequence that is a common subsequence of A and B .

A sequence C is a *similar subsequence* of two sequences A, B if and only if C is a subsequence of A and we can replace at most K elements in C such that C is a subsequence of B . The longest similar subsequence (LSS) of two sequences A, B is the longest sequence that is a similar subsequence of A and B .

Hanhan loves the longest similar subsequence. He has two integer sequences, A and B and wants to know the length of the longest similar subsequence of A and B . Please help him to find the answer.

Input

The first line contains three space-separated integers N, M, K , where N is the length of sequence A , M is the length of sequence B , K is the constant in the definition of a similar sequence. The second line contains N space-separated integers indicating the elements of A , that is, A_1, A_2, \dots, A_N . The third line contains M space-separated integers indicating the elements of B , that is, B_1, B_2, \dots, B_M .

- $1 \leq N, M \leq 2 \times 10^3$
- $0 \leq K \leq 20$
- $0 \leq A_i, B_j \leq 1021$

Subtask 1 (40 %)

- $K = 0$, the problem is reduced to longest common subsequence (LCS).

Subtask 2 (60 %)

- No other constraint

Output

Output one line containing an integer indicating the length of the longest similar subsequence (LSS) of A and B .

Sample Input 1

```
2 2 0
1 0
2 1
```

Sample Output 1

```
1
```

Sample Input 2

2 2 1
0 1
2 1

Sample Output 2

2

Problem 3 - Fishing (Programming) (20 points)

Problem Description

Eddy is a fisherman. He is simply good at fishing.

One day, he finds a nice place to fish. There are n fishponds in one line. Moving from the i -th fishpond to the j -th fishpond would cost $|i - j|$ units of time.

Initially, Eddy can get F_i fish in the i -th fishpond. In the next turn at the same fishpond, the amount of fish he can get is decreased by D_i . Notice that Eddy will not get negative amount of fish. Each turn of fishing takes Eddy 1 unit of time.

For example, if $F_1 = 10, F_2 = 5, D_1 = 2, D_2 = 3$ and Eddy fishes in the each fishpond three turns, he will get $10 + 8 + 6 + 5 + 2 + 0 = 31$ fish.

Since Eddy has only k units of time to fish, could you tell him how to get the maximum amount of fish?

At the beginning, Eddy is at the first fishpond.

Input

The first line contains an integer T , indicating the number of test cases. The first line of each test case contains two integers n, k . Each of the next n lines contains two integers F_i, D_i .

- $1 \leq T \leq 10$
- $1 \leq n \leq 1000$
- $1 \leq k \leq 10000$
- $0 \leq F_i, D_i \leq 10^9$

Subtask1 (40 %)

- $1 \leq n \leq 50$
- $1 \leq k \leq 500$

Subtask2 (60 %)

- No other constraint.

Output

Output the way to get maximum amount of the fish for each test case using the following format.

The first line contains two integers a, m ($m \leq 10 \times n$), indicating the maximum amount of fish and the total number of the operations Eddy does. Each of the following m lines indicating one operation as below:

- **move** x , indicating Eddy moves to the x -th fishpond.
- **fish** y , indicating Eddy does y ($y > 0$) turns of fishing in the current fishpond.

Sample Input

```
2
2 6
10 2
5 3
3 10000
10 2
5 3
100 1
```

Sample Output

```
33 3
fish 4
move 2
fish 1
5087 9
fish 4
move 2
fish 3
move 1
fish 100
move 3
fish 100
move 2
fish 100
```

Note**First sample test case**

1. Eddy spent 4 units of time at the first fishpond and got 28 fish.
2. Eddy spent 1 unit of time to moved to the second fishpond.
3. Eddy spent 1 unit of time at the second fishpond and got 5 fish.

Finally, Eddy got 33 fish and the total time he spent is 6 units.

Second sample test case

1. Eddy spent 4 units of time at the first fishpond and got 28 fish.
2. Eddy spent 1 unit of time to moved to the second fishpond.
3. Eddy spent 3 units of time at the second fishpond and got 7 fish.
4. Eddy spent 1 unit of time to moved to the first fishpond.
5. Eddy spent 100 units of time at the first fishpond and got 2 fish.
6. Eddy spent 2 units of time to moved to the third fishpond.
7. Eddy spent 100 units of time at the the third fishpond and got 5050 fish.
8. Eddy spent 1 unit of time to moved to the second fishpond.
9. Eddy spent 100 unit of time at the second fishpond and got 0 fish.

Finally, Eddy got 5087 fish and the total time he spent is 312 units.

Problem 4 - Balanced Parentheses (Programming) (20 points)

Problem Description

Hanhan loves seeing balanced parentheses: that is, each opening symbol '(' has a corresponding closing symbol ')' and the pairs of parentheses are properly nested.

The string of "()" is balanced. If A and B have balanced parentheses, then (A) and AB also represent balanced parentheses. For example, "(()()())", "((((())))", and "(()((()())))" are balanced parentheses. But, "(((((((())", "())", "(()()())" are not.

Hanhan receives a string of parentheses. But, some positions of the string are lost. Hanhan wants to fix it and makes it balanced again.

More specifically, Hanhan receives a string composed of '(', ')', and '?'. Hanhan needs to replace each '?' with '(' or ')' such that the string of parentheses becomes balanced. However, replacing each '?' with '(' or ')' has cost: Replacing i -th '?' with '(' costs l_i while replacing i -th '?' with ')' costs r_i . Hanhan wants to know the minimum possible cost to fix the string into balanced parentheses. Please help Hanhan find the minimum cost or tell him it's impossible to fix.

Input

The first line contains a string S indicating the string received by Hanhan. If there are K '?'s in the S , there will be K lines followed. For each following lines, each contains two space-separated integer l_i, r_i indicating the cost replacing i -th '?' with '(' or ')', respectively.

- $1 \leq |S| \leq 10^6$, length of S won't exceed 10^6
- S contains only '(', ')', and '?'
- $0 \leq l_i, r_i \leq 10^9$

Subtask 1 (20 %)

- $|S| \leq 10^2$

Subtask 2 (30 %)

- $|S| \leq 5 \times 10^3$

Subtask 3 (50 %)

- No other constraint

Output

Output one line containing one integer indicating the minimum possible cost if it's possible to fix the string. Otherwise, output "Impossible" in one line.

Sample Input 1

```
(??)
10 0
0 10
```

Sample Output 1

```
0
```

Sample Input 2

```
(??  
10 0  
0 10
```

Sample Output 2

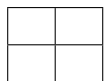
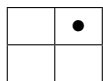
```
Impossible
```

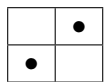
Hint

After replacing '(' with $+1$ and ')' with -1 , does the prefix sum of the resulted sequence have some relation with the balanced parentheses?

Problem 5 - Dynamic Programming (Handwriting) (20 points)

- (1) (30%) Given a sequence S of positive numbers, find a non-decreasing subsequence with the maximum sum. For example, if $S = \{2, 2, 3, 1, 9, 3, 4\}$, the answer should be $16(2+2+3+9 = 16)$. Use dynamic programming to solve this problem and show that the time complexity is $O(n^2)$.
- (2) Given $n, m (n \leq m)$ that represent an $n \times m$ grid, we want to put some objects on the grid satisfying the following rules:
1. Each grid can at most have one object. In other words, each grid can have zero or one object.
 2. At most one object can be in any 2×2 grid. For example,


 and
 
 are valid.


 is invalid.

How many valid ways we can put objects into the grid? Provide a dynamic programming solution with time complexity $O(4^n \times n \times m)$.

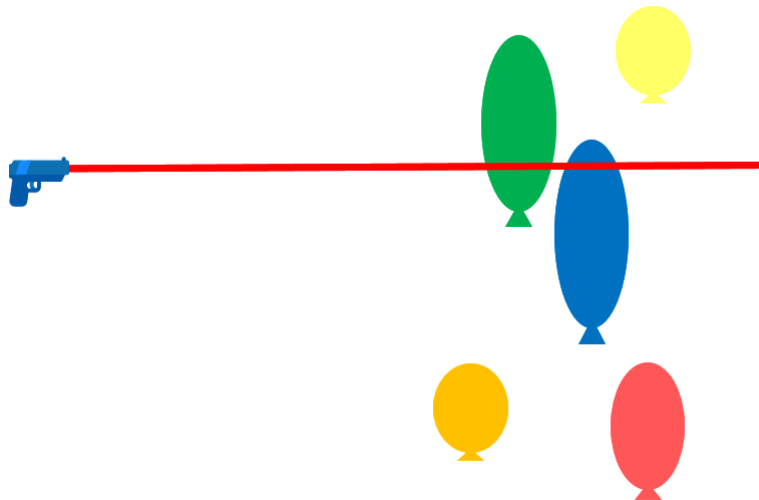
- (a) (35%) Define the subproblem and the transition function (i.e., relationships between subproblems) of your dynamic programming solution.
- (b) (35%) Explain the time complexity of your solution.

Problem 6 - Greedy Algorithm (Handwriting) (20 points)

Bob is a CSIE student and he has been single for almost 20 years. He was really eager to find ways to have a relation. However, what he did was always in vain. After learning the mood changing theory during last homework, he finally made a great progress. He chatted with Alice, the girl he had a crush on, and had a great time. Today, Bob and Alice decided to go to a night market together.

They came to a shooting stall in the night market, where there were many balloons scattering vertically with stationary positions. Bob was given a laser gun to shoot all of them without bullet limit. To impress Alice, Bob wanted to use the minimum number of shots to break all balloons.

Suppose that the gun can only shoot in a horizontal direction and will break all balloons covering the same horizontal line. Thus, Bob just needs to decide the y-position of the gun.



You will be given a sequence S of n positions of balloons: $S = \{b_0, b_1, \dots, b_{n-1}\}$, where each balloon b_i has its lowest position and its highest position, denoted by $[y_i^l, y_i^h]$.

Given the lowest position y^l and the highest one y^h of each balloon, the i -th balloon will burst if the gun shoots at y when $y_i^l \leq y \leq y_i^h$.

Please help Bob design a greedy algorithm to calculate the minimum number of shots.

Input: $[4, 6], [11, 13], [1, 5], [3, 7], [10, 16]$ (a sequence of the balloon positions)

Output: 2 (the minimum number of shots)

(1) (10%) Please decide the minimum number of shots for the following input sequence

$[5, 8], [3, 9], [11, 16], [10, 12], [11, 16], [6, 9], [14, 16]$

(2) (40%) Please help Bob design a greedy algorithm that finds the optimal solution in $O(n \log n)$ and prove its time complexity.

(3) (50%) Please prove the correctness of the algorithm, which contains *Greedy Choice Property* and *Optimal Substructure*.