# Homework #2

**Blue Correction Date: 10/20/2022 12:00**

**Red Correction Date: 10/14/2022 22:00**

Due Time: 2022/10/25 14:20

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

## Instructions and Announcements

- There are **four programming problems** and **two hand-written problems**.
- **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. For each sub-problem, please label (on Gradescope) the corresponding pages where your work shows up. 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 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 - Sleeping Stages (Programming) (10 points)

## Problem Description

Wonling is a legendary gifted Mahjong girl and a super talented student, who receives the Academic Excellence Prize every semester. To keep a clear mindset during all quizzes and exams, she schedules her sleeping time very well.

For a single day, she has $N$ minutes to sleep and she likes to partition her sleeping time into $K$ sleeping stages, each of which is at least one-minute long. Today, she watched a horror movie before sleeping, so today's $k$-th minute in her sleeping is cursed by a potential scariness value $a_k$ ($\forall k \in [1..N]$).

Also, Wonling will have exactly one nightmare for each sleeping stage tonight. Formally speaking, for each sleeping stage starting from the $\ell$-th minute and ending at the $r$-th minute, there is a nightmare starting from the $i$-th minute and ending at the $j$-th minute such that:

- $\ell \leq i \leq j \leq r$
- This nightmare contributes a discomfort value $(r - \ell + 1) \cdot \sum_{k=i}^{j} a_k$ to Wonling.

Even if Wonling is a gifted girl, she cannot control the time when a nightmare occurs. Thus, for each sleeping stage, she considers the worst-case discomfort value $\max_{\ell \leq i \leq j \leq r} \left( (r - \ell + 1) \cdot \sum_{k=i}^{j} a_k \right)$. She wants to partition those $N$ minutes of sleeping time into $K$ sleeping stages such that the summation of the worst-case discomfort values for all sleeping stages is **minimized**. Can you help her?

## Input

For each testcase, there are two lines. The first line contains two integers $N$ and $K$, representing how many minutes Wonling can sleep and the number of sleeping stages she would like. The second line contains $N$ integers $a_1, a_2, \cdots, a_N$ representing the potential scariness values.

In other words, the input is presented in the following format:

```
N K
a_1 ... a_N
```

- $1 \leq N \leq 1000$
- $1 \leq K \leq \min(N, 500)$
- $-10^6 \leq a_k \leq 10^6$

## Output

For each testcase, output the minimal summation of the worst-case discomfort values.

**Test Group 0 (0 %)**

- Sample input

**Test Group 1 (20 %)**

- $K = 2$

**Test Group 2 (20 %)**

- $1 \leq N \leq 100$
- $1 \leq K \leq \min(N, 20)$

**Test Group 3 (60 %)**

- No additional constraint

### Sample Input 1

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

### Sample Output 1

```
26
```

### Sample Input 2

```
10 3
1 -5 3 -4 3 -5 9 -6 -3 6
```

### Sample Output 2

```
45
```

### Sample Input 3

```
20 4
3 -1 -4 1 -5 9 -2 6 -5 3
5 -8 9 -7 9 3 -2 3 8 -4
```

### Sample Output 3

```
187
```

## Notes

- The Sample Input 3 only contains two lines; the second and third lines should be in the same line as below.

  ```
  20 4
  3 -1 -4 1 -5 9 -2 6 -5 3 5 -8 9 -7 9 3 -2 3 8 -4
  ```

  It is reformatted into three lines in this document simply because a single line containing 20 numbers is too long to fit in the page margin.
- In the Sample Input 1, $[5, -2, 3]$ and $[-1, 4]$ is a possible partition to achieve the output value 26.

# Problem 2 - Big Chef DanBin (Programming) (10 points)

## Problem Description

"I'm Pasta" is one of the most famous restaurants in Alley 118. Today, several groups of people would like to have spaghetti there for dinner. DanBin, the chef of I'm Pasta, wants to go home earlier to watch his favorite VTuber's live streaming. Hence, he plans to arrange the order of dishes such that people can leave the restaurant as early as possible.

The $j$-th customer from the $i$-th group orders a dish that needs $a_{i,j}$ minutes for DanBin to prepare and $b_{i,j}$ minutes for this customer to finish. In order to keep the high quality of dishes, DanBin can only prepare one dish at a time, but several people can, of course, have their own dishes at the same time.

According to the regulations in I'm Pasta, DanBin has to prepare dishes for each group one by one. That is, DanBin has to prepare dishes for all customers in the same group consecutively. Besides, all people in the same group will leave the restaurant together when all of them have finished their own dishes.

Can you tell DanBin the minimum **sum** of all customers' time leaving the restaurant if he prepares the dishes in the optimal order?

## Input

The first line contains an integer $N$, indicating the number of groups. Then, $N$ lines follow. The $i$-th of these lines contains several integers, the first of which is the size of the $i$-th group, $m_i$, then the rest $2 \times m_i$ numbers are $a_{i,1}, b_{i,1}, a_{i,2}, b_{i,2}, \ldots, a_{i,m_i}, b_{i,m_i}$, representing the needed time for preparing and enjoying the dishes.

- $1 \leq N \leq 10^5$
- $1 \leq m_i \leq 10^5$
- $1 \leq \sum_{i=1}^{n} m_i \leq 10^5$
- $1 \leq a_j, b_j \leq 10^8$

In other words, the input is presented in the following format:

$N$
$m_1\ a_{1,1}\ b_{1,1}\ \ldots\ a_{1,m_1}\ b_{1,m_1}$
$\vdots$
$m_N\ a_{N,1}\ b_{N,1}\ \ldots\ a_{N,m_N}\ b_{1,m_N}$

## Output

Output the the minimum **sum** (in minutes) of all customers' time leaving I'm Pasta.

### Test Group 0 (0 %)

- Sample input

### Test Group 1 (30 %)

- $N = 1$

### Test Group 2 (30 %)

- $m_i = 1$

### Test Group 3 (40 %)

- No additional constraints

### Sample Input 1

```
1
3 4 2 5 8 9 2
```

### Sample Output 1

```
60
```

### Sample Input 2

```
3
1 4 2
1 5 8
1 9 2
```

### Sample Output 2

```
43
```

### Sample Input 3

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

### Sample Output 3

```
373
```

## Notes

- In **Sample 1**, preparing the dishes for customer 2, then customer 1 and finally customer 3 would be the best order. These people will leave I'm Pasta in 20 minutes, so the answer is $20 + 20 + 20 = 60$ minutes.
- In **Sample 2**, preparing the dishes for customer 1, then customer 2 and finally customer 3 would be the best order. The first group of people will leave I'm Pasta in 6 minutes, the second group of people will leave I'm Pasta in 17 minutes, and the third group of people will leave I'm Pasta in 20 minutes. As a result, the answer is $6 + 17 + 20 = 43$ minutes.

# Problem 3 - Three-Mouthed Sheep and the String (Programming) (15 points)

## Problem Description

Three-mouthed sheep is a strange animal in the ADA kingdom. One day, it receives a string $S$ with $N$ characters as a gift. As a strange animal, it wants to make the string as strange as possible, and it thinks that a string $A$ is stranger than a string $B$ if and only if $A$ is lexicographically smaller than $B$, which is also a strange definition. A string $A$ is defined as lexicographically smaller than a string $B$ if one of the following conditions holds:

- $A$ is a proper prefix of $B$; that is, $A$ is a prefix of $B$ and $A \neq B$.
- In the first different characters between $A$ and $B$, the character in $A$ has a smaller ASCII code than the one in $B$.

For example, `"abc"` is stranger than `"bac"` because ASCII(`'a'`) $= 97 <$ ASCII(`'b'`) $= 98$.

Because the three-mouthed sheep is strange, other animals in the ADA kingdom don't want it to move and only allow the three-mouthed sheep to perform at most $K$ operations. In each operation, it can swap any two adjacent characters in $A$. What is the strangest string it can obtain from $A$ if it operates optimally?

## Input

The first line is the string $S$ in the length $N$ the three-mouthed sheep initially receives. The second line contains an integer $K$.

In other words, the input is presented in the following format:

```
S
K
```

- $1 \leq N \leq 5 \times 10^5$
- $0 \leq K \leq \frac{N(N-1)}{2}$
- The string contains only digits and Latin alphabets (including both lowercase and uppercase).

## Output

Output the strangest string that the three-mouthed Sheep can achieve by swapping any two adjacent characters for at most $K$ times.

**Test Group 0 (0 %)**

- Sample input

**Test Group 1 (10 %)**

- $1 \leq N \leq 10$

**Test Group 2 (20 %)**

- $1 \leq N \leq 1000$

**Test Group 3 (30 %)**

- $S$ only contains `'a'`, `'b'`

**Test Group 4 (40 %)**

- No additional constraint

### Sample Input 1

```
dcxxxa
1
```

### Sample Output 1

```
cdxxxa
```

### Sample Input 2

```
dcxxxa
5
```

### Sample Output 2

```
adcxxx
```

### Sample Input 3

```
Aa210
5
```

### Sample Output 3

```
0A2a1
```

### Notes

- The intended solution does not involve any advanced data structure like binary indexed tree, segment tree, treap, etc. If you insist on using these data structures in your code, the TA reserves the right to decline your debugging request.

# Problem 4 - Omelet and the LEGO® Tower Vol. 2 (Programming) (15 points)

## Problem Description

Recall that Omelet loves playing with LEGO® bricks. This time, he comes up with a new way to play. He places $N$ tower bases on the playground and builds towers on them in a mysterious way.

More formally, there are initially $N$ tower bases on the ground without any bricks on top of them. Then, he will perform the following operation at most $K$ times. Each operation consists of two steps:

1. Pick two numbers, $\ell, r\,(1 \leq \ell \leq r \leq N)$.
2. For all $i$ satisfying $\ell \leq i \leq r$, put a brick on the top of the $i$-th tower.

After Omelet finishes building towers, he takes a photo of the playground with all $N$ towers in it. Now he's wondering how many different photos he can collect. Note that all bricks are indistinguishable and two photos are different if and only if there exists at least one $i\,(1 \leq i \leq N)$ such that the $i$-th tower has different heights in two photos.

Since the answer might be very large, we only want to know the answer modulo $M$.

## Input

The input contains one line with three numbers $N, K, M$.

In other words, the input is presented in the following format:

```
N K M
```

- $1 \leq N \leq 300$
- $1 \leq K \leq 300$
- $1 \leq M \leq 10^9 + 7$

## Output

Output a number in a line that represents the answer.

Let $X$ be the number all different photos. Then you should output $F$, where $0 \leq F < M$ and $F \equiv X \pmod{M}$.

**Test Group 0 (0 %)**

- Sample input

**Test Group 1 (5 %)**

- $K = 1$

**Test Group 2 (10 %)**

- $1 \leq K \leq 3$

**Test Group 3 (50 %)**

- $1 \leq K \leq 50$

**Test Group 4 (35 %)**

- No additional constraints

### Sample Input 1

```
2 1 100
```

### Sample Output 1

```
4
```

### Sample Input 2

```
2 2 100
```

### Sample Output 2

```
9
```

### Sample Input 3

```
228 244 1000000007
```

### Sample Output 3

```
502498633
```

## Notes

- In the first sample testcase, all possible photos are $(0,0),(0,1),(1,0),(1,1)$.
- In the second sample testcase, all possible photos are $(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)$.

## Hints

Here are two techniques that may provide insights to this problem.

#### Difference Array & Prefix Sum

Let's consider two arrays, $A$ with the length $N$ and $B$ with the length $N+1$, where all entries are 0 initially. If we define the entries of $A$ and $B$ with

$$\begin{cases} B[1] = A[1] \\ B[i] = A[i] - A[i-1], & \forall 1 < i \leq N \end{cases}$$

Then, you may notice that increasing $A[l..r]$ by one is equivalent to simply increasing $B[l]$ by one and decreasing $B[r+1]$ by one. As for restoring the entries in $A$ from $B$, we can add up a prefix of $B$. Formally, it can be shown that the equation below is equivalent to the definition above.

$$A[i] = \sum_{k=1}^{i} B[k], \quad \forall 1 \leq i \leq N$$

These techniques are called difference array and prefix sum.

#### Modular Operation

- Avoid redundant modular operations since modular operations can be slow. For example, if you want to find $a+b$ with $0 \leq a, b < M$ holds, you should use `(a + b >= M ? a + b - M : a + b)` instead of `(a + b) % M`.

#### Reminder

You might not have space for $300^3$ integers. To save memory usage, you may want to overwrite some data which will no longer be used in the rest of your algorithms.

## Problem 5 - Generals.io (Hand-Written) (25 points)

*Note: In this problem, you are not allowed to write pseudocode. Please explain your algorithm in words.*

As a general of CSIE kingdom, you have to defend your country from invasion. Specifically, you anticipate $N$ waves of attack from the enemies, and the $i$-th wave will arrive on the day $T_i$. As a part of the defense measures, some military contractors proposed $M$ weapon construction plans. The $i$-th construction can be done on the day $t_i$ and costs $p_i$ dollars. We need exactly one weapon to successfully defend against one wave of attack. Also, after each wave of attack, the used weapon will be destroyed and is no longer usable under the heavy gunfire.

Now, your goal is to minimize the cost for constructions while keeping the country safe from all hostility. ($T_i$ and $t_i$ are all distinct. $\langle T_i \rangle$ is in an ascending order. Also, you can assume that there exists at least one way to successfully defend against all attacks, which leads to $N \in O(M)$.)

### (a) (2 points)

Given $N = 4,\ M = 7,\ T = [9, 21, 28, 32],\ t = [10, 3, 16, 39, 25, 31, 5],\ p = [4, 5, 6, 1, 2, 7, 3]$, find the optimal **choice**.

### (b) (7 points)

Design an $O(M \log M)$ greedy algorithm to find the minimum cost and prove its correctness (including the greedy-choice property) and time complexity.

Sadly, you find out that your country is too small to accommodate so many constructions. Thus, the number of weapons that are ready for service must not exceed $K$ at any time.

### (c) (2 points)

Given $N = 5,\ M = 9,\ K = 2,\ T = [9, 21, 31, 39, 45],\ t = [25, 10, 16, 43, 50, 5, 28, 32, 3],\ p = [9, 1, 2, 8, 7, 3, 6, 5, 4]$, find the optimal **choice**.

### (d) (7 points)

Design an $O(MK + M \log M)$ dynamic-programming algorithm to find the minimum cost and prove its correctness and time complexity.

### (e) (bonus 5 points)

Design an $O(M \log M)$ greedy algorithm to find the minimum cost and prove its correctness (including the greedy-choice property) and time complexity.

Clever as you are, you come up with the idea of requisitioning private residence to resolve the land-size restriction. Now, you can construct as many weapons as you want, but you have to pay the residents 1 dollar per day per weapon as long as the weapon is ready for service.

## (f) (7 points)

Design an algorithm to find the minimum cost that runs as fast as possible and prove its correctness and time complexity.

## Problem 6 - $(N+1)$-Legged Race (Hand-Written) (25 points)

The ADA Kingdom is about to hold an $(n+1)$-legged race ! Everyone in the ADA Kingdom has to participate in this race, and each person has his/her own velocity $v_i$.

To compete in the race, all participants form several teams. There is **no limit** to the number of people in a team. That is, you can form a team with only one person or even all people in the kingdom. The **speed difference of a group**: $\delta$, is defined as the speed of the fastest person in that group minus the speed of the slowest person in that group (i.e., for a group $g$, $\delta(g) = \max_{i \in g} v_i - \min_{i \in g} v_i$). Note that a team consisting of only one person has a speed difference of 0. The **total speed difference** is the sum of the speed differences in all groups (i.e., $\sum_g \delta(g)$). You can assume that $v_i \neq v_j$ if $i \neq j$. (i.e. There are no two people having the same velocity.)

Please solve each subproblem based on the above premises. Also, you can assume that basic integer operations ($+$, $-$, $\times$, $\div$) can be done in $O(1)$ time and you don't need to worry about integer overflow.

In all subproblems, We recommend you to explain the algorithm in words. If you want to answer it in pseudo code, make sure it is correct, readable (comment or explain if needed), and **less than 35 lines**.

### (a) (4 points)

The king of the ADA kingdom wants to separate the participants into teams for the $(n+1)$-legged race, but he does not know how many groups to divide the people into.

Please design an algorithm which runs in $O(N \log N)$ time to calculate the **minimum total speed difference** for every $i$ from 1 to $N$, if he separates all people into $i$ groups. Briefly explain the correctness and time complexity of your algorithm.

### (b) (5 points)

The king wants to divide all participants into **two unordered groups**: $A$ and $B$, and conducts an $(n+1)$-legged race such that the total speed difference of these two groups is **not greater than** a given $K$ (i.e., $\delta(A) + \delta(B) \leq K$).

Please design an algorithm that runs in $O(N \log N)$ to calculate how many different ways he can divide the participants. Note that two ways are different if and only if there exists two people $a, b$ such that they are in the same group in one way but in different ones in another. Briefly explain the correctness and time complexity of your algorithm.

### (c) (1 point)

Please list **all different ways** for grouping **four people** into any number of groups without further restriction.

You can use $i$ to denote the $i$-th person, e.g., $\{\{1,2,3,4\}\}, \{\{1\},\{2\},\{3\},\{4\}\}, \ldots$

### (d) (7 points)

Design an algorithm using **dynamic programming** with time complexity $O(N^2)$ to solve the following problem:

Without any other restriction, **how many ways** are there to group $N$ people into any number of groups? We recommend writing your solution in the following form:

1. Define the **subproblem** of dynamic programming and define **each variable** in it. (2 points)
2. Write down the **recurrence relationship** between subproblems. (2 points)
3. Briefly explain the **correctness** and both **time** and **space** complexity of your algorithm. (3 points)

## (e) (1 point)

There are four people with their velocity $1, 2, 3, 5$, respectively. Assuming the total speed difference cannot be greater than **3**, list **all different ways** of grouping them into any number of groups.

You can use a person's velocity to represent this person, e.g., $\{\{1, 2\}, \{3, 5\}\}, \{\{1\}, \{2\}, \{3\}, \{5\}\}, \ldots$

## (f) (7 points)

Design an algorithm using **dynamic programming** with time complexity $O(N^2K)$ to solve the following problem:

The king limits the total speed difference to be not greater than $K$. Given the velocity $v_i$ of every person, **how many ways** are there to group $N$ people into any number of groups? We recommend write your solution in the form below:

1. Define the **subproblem** of dynamic programming and define **each variable** in it. (2 points)
2. Write down the **recurrence relationship** between subproblems. (2 points)
3. Briefly explain the **correctness** and both **time and space** complexity of your algorithm. (3 points)