Homework #2

Due Time: 2018/11/06 (Tues.) 18:00
Contact TAs: `ada-ta@csie.ntu.edu.tw`

Instructions and Announcements

- There are **four programming problems** and **three hand-written problems**, and the homework set including bonus are worthy of 110 points. If you get more than 100 points, your score will still be counted as 100 points.
- **Programming.** The judge system is located at `https://ada18-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.
- **Hand-written.** For other problems (also known as the "hand-written problems"), please turn in **a printed/written version** of your answers to the instructor at the beginning of the class on 2018/11/01, or put them in the box in front of R307 before the deadline. **Remember to print your name/student ID on the first page of your submitted answers.** In case that your homework is lost during the grading, you can also upload your homework to the NTU COOL system. 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.
- With TAs' discretion, too complicated solutions will be counted as wrong.

Problem 1 - Pusheen the Cat (Programming) (10 points)

Problem Description

Pie, a snack lover, noticed that there are cupcakes randomly appear in his house recently. After many efforts (including eating cupcakes), he finds out that these cupcakes are produced by his pet cat, Pusheen.

After some survey about Pusheen, Pie finds a book named "Assistant for Dark-magic Access", which explains how Pusheen produces cupcakes using dark magic (*not* from its digestive system). The book says that a level-k cat with dark magic will produce 2^k cupcakes every day and Pusheen is currently a level-0 cat. If you feed Pusheen with ramen *after it produces cupcake(s)*, it will level up! That is, a level-k cat will become level-$(k+1)$ after consuming a bowl of ramen, but Pusheen can only eat one bowl of ramen per day. Note that Pusheen must produce cupcake(s) *every day.*

After knowing this, Pie decides to get cupcakes from Pusheen! However, after considering the high calories of cupcakes, Pie decides to have *exactly* N cupcakes from Pusheen. Pie wants to find out the minimum day(s) needed to get N cupcakes starting from a level-0 Pusheen. Can you help him find the answer? As a reward, Pie will give you the score of this problem.

Input

The first line of the input file contains an integer T, indicating the number of testcase(s). Each testcase contains an integer N, indicating the desired number of cupcake(s).

- $1 \leq T \leq 200,000$
- $1 \leq N \leq 1,000,000,000$

Subtask 1 (40 %)

- $N \leq 1,000$

Subtask 2 (60%)

- no other constraints.

Output

For each testcase please output a line with an integer indicating the minimum day(s) need to reach N cupcakes.

Sample Input	Sample Output
4 1 13 127 4610	1 5 7 15

Problem 2 - ADA Farm (Programming) (15 points + Bonus 3 points)

Problem Description

Have you ever heard of the ADA farm? It is a great place to visit with your friends (if any)!

The ADA farm is well-known for the enormous number of horses, and you will agree with this after reading the "Input" section. There are N horses in total on the ADA farm, where each horse lives in a different position. However, you noticed that some horses look lonely. After observing them for a while, you know that the horses often visit their friends during their free time. If other horses are really far away, then visiting them becomes time-consuming and tiring so that the horse will be upset.

The way horses move is really unusual. If you consider the ADA farm as a 2-dimensional space, then a horse at coordinate (x, y) has 2 options to move:

1. The well-known way: the horse can spend 2 seconds and jump to one of $\{(x+2, y+1), (x+2, y-1), (x+1, y-2), (x-1, y-2), (x-2, y-1), (x-2, y+1), (x-1, y+2), (x+1, y+2)\}$, like the normal horses.
2. The unusual way: the horse can spend 1 second and walk to one of $\{(x+1, y), (x, y+1), (x-1, y), (x, y-1)\}$, just like a soldier! It's really unbelievable, isn't it?

We assign numbers for these horses from 1 to N with the i-th horse h_i living at a position (x_i, y_i). Then we can define a *loneliness value* L for each horse h:

$$L(h) = \sum_{i=1}^{n} d(h, h_i),$$

where $d(a, b)$ is the minimum time cost between the position of the horse a and the horse b. That is, the loneliness of a horse is the sum of the minimum time needed for it to move from its home to another horse's home.

In order to know more about the horses, can you find the loneliness value of each of N horses in the ADA farm?

Input

The first line of the input file contains an integer N, indicating the number of horses in the ADA farm.

For the next N lines, the i-th line contains 2 integers x_i, y_i, indicating the position of i-th horse in the ADA farm.

- $2 \leq N \leq 100,000$
- $0 \leq x_i, y_i \leq 1,000,000$

Subtask 1 (70 %)

- $N = 2$

Subtask 2 (30 %)

- $N \leq 1000$

Subtask 3 (Bonus, 20%) (Very Difficult!)

- no other constraints.

Output

Please output N lines, where the i-th line contains an integer indicating the loneliness of i-th horse.

Sample Input 1

```
2
0 0
1 2
```

Sample Output 1

```
2
2
```

Sample Input 2

```
3
0 0
3 3
4 3
```

Sample Output 2

```
9
5
6
```

Problem 3-1 - Illuminati Matrix (Programming) (10 points)

Problem Description

Joe is convinced of Illuminati. He strongly believes that some numbers represent the holy trinity and calls them Illuminati numbers, where the definition of is that a number x satisfies all below constraints:

- $x \equiv 0 \ (mod\ 33)$
- The number of digits 3 in x $\equiv 0 \ (mod\ 3)$
- The number of digits 6 in x $\equiv 0 \ (mod\ 3)$
- The number of digits 9 in x $\equiv 0 \ (mod\ 3)$

After you lost Joe in the stone game a few weeks ago, you got trapped in a matrix maze consisted of (nm) cells. Each cell has a magic power. If you pass through a cell whose magic power satisfies the Illuminati constraints, you will be killed by Satan and be in hell forever.

Initially, you are in the top-leftmost cell, and the exit is in the bottom-rightmost cell. You want to know the number of possible paths to escape the maze, but you can only go right or down in order to leave the maze as fast as possible.

Input

The first line contains two integers n, m indicating the size of the matrix, where $1 \leq n, m \leq 1,000$.

For the next n lines, the i^{th} line contains m integers $A_{i,j}$ indicating the magic power in cell (i, j), where $1 \leq A_{i,j} \leq 10^{18}$.

Subtask 1 (20 %)

- $n, m \leq 4$

Subtask 2 (80 %)

- No other constraints.

Output

Output the number of safe simple paths from top-left corner to bottom-right corner module 10^9+7.

Sample Input 1

```
3 3
142 3 528
112 875 2475
615 781 1
```

Sample Input 2

```
2 5
1 666666 3 4 55
2 4 5 333333 1
```

Sample Output 1

```
3
```

Sample Output 2

```
0
```

For the second sample, even though "1 2 4 5 3 4 55 1" do not contain any Illuminati number, it is not a valid path since you can only go right or down.

Problem 3-2 - Illuminati and Joe (Programming) (15 points)

Problem Description

After you escaped from the maze, Joe asked you to join the Illuminati. Your believes are the Pyramid, the Eye, the Light, and the Eternal Circle. You also know that the selfish pursuit of money is a hollow goal, but the pursuit of the goodness that money can create is one of humanity's greatest responsibilities. Therefore, you decided to devote yourself to Illuminati.

Now, you are doing your first job "strengthening the matrix maze". You want to know how many Illuminati numbers there are in a specific range, so that you can design a robust maze.

Input

There is only one line containing two integers l, r indicating the range, where $1 \leq l \leq r \leq 10^{1000}$.

Output

Output the number of Illuminati numbers in the range $[l, r]$ module $10^9 + 7$.

Subtask 1 (20 %)

- $l, r \leq 10^7$

Subtask 2 (80 %)

- No other constraints.

Sample Input

```
1 1243567
```

Sample Output

```
5218
```

Problem 4 - Digit Dynamic Programming (Hand-Written) (15 points)

Digit Dynamic Programming is a useful trick for solving problems with some constraints about digits. Let's take the following problem as an example.

Censorship

You have heard of *ADA Kingdom* in homework 1, but I think few of you know the miserable history of this kingdom. Let me tell you a story.

ADA Kingdom is actually built by *Handsome*, a TA of ADA course, by using his outstanding appearance and charisma. However, "power tends to corrupt and absolute power corrupts absolutely." Only in a few months, *ADA Kingdom* had become an autocratic country under *Handsome*'s terror reign.

In order to consolidate his power, he has put literal inquisition into effect. Some 2-digit numbers, for some reasons, are considered *ugly*. In addition, every integer that contains any ugly number as its substring, when written in decimal, is considered *illegal*. For example, if both "87" and "38" are ugly, then "1**387**" and "1**87**63" are all considered illegal, while "378" and "30083" are legal.
You are a mathematician in *ADA Kingdom*, and you would like to know how many available integers are there in a specified interval. Now, given those ugly 2-digit numbers, and an integer N, how many legal numbers are there between 1 and N? Please come up with an algorithm to compute this, with time complexity $O(\lg N)$.

Note that when input and output, the length of N, in decimal, is only $O(\lg N)$. Thus, neither inputting N nor outputting the answer would cause `TLE` directly.

The time limit is too tight to enumerate all the legal/illegal integers. Fortunately, we can compute this through simple dynamic programming.

For simplicity, a function S and an array `dp` are defined as follow.

$$S(n,k) = \{x | x \text{ is legal, of length } n, \text{ and begins with } k. \text{ If } k = 0 \text{ then } x \text{ can have leading zeros.}\}$$
$$\texttt{dp[n][k]} := |S(n,k)|.$$

For example, "378" belongs to $S(3,3)$, and "30083" belongs to $S(5,3)$.

(1) (5 pts) Obviously, every legal number $a_1a_2...a_n$ can be viewed as a_1 followed by a legal number $a_2a_3...a_n$. In the other hand, for every legal number $a_2a_3...a_n$, if a_1a_2 is not prohibited, then $a_1a_2...a_n$ is also a legal number.

Based on the observation above, and assuming that every `dp[i][j]` where `i` $<$ `n`, has been computed. Please describe how to compute `dp[n][k]`.

(2) (2 pts) A legal integer should not begin with zero. Then, do we still need `dp[n][0]`? Why or why not?

(3) (8 pts) Now assume that you've computed the whole `dp` array. Please design an algorithm to compute the number of legal numbers in $[0, n)$ with time complexity $O(\lg n)$.

Problem 5 - The Robbers (Hand-Written) (30 points)

In 2500 A.D., the world suffers from drugs, violence, and crimes.

It is no longer a peaceful, but horrible and suspicious society. The gap between the rich and the poor is becoming larger and larger. Moreover, because the population density is much higher than before, the living space is extremely compressed and packed. As a result, the roads become considerably narrow, and in turn makes robbery much easier to rob people than before. Under such circumstances, robbers can easily find a victim and rob all of his/her valuable belongings, because the victims have no way to escape. Robbers, accordingly, have become one of the most thriving occupations in this evil society. What's worse, the robbers are often brilliant people, so they always work as a group and use their intelligence to rob as much as possible. Consequently, there is almost no way for an innocent people to escape from their threat. They have no choice but to give out their belongings. To the robbers, taking over one's valuable belongings is only a matter of time.

(1) There are N innocent people standing in a line on one side of a narrow one-end alley, and M ($\leq N$) robbers in a group standing in a line on the other side of the alley. Note that the alley is extremely narrow so that the innocent people cannot exchange the relative positions with others. In addition, a security, who is already bribed by the robbers, has closed the only entrance of the alley, so the people can never escape.

Some people are so strong that they can resist for a longer period of time, while others are not powerful enough to fight with the robbers. Namely, the i-th person can resist for t_i ($0 < t_i \leq N$) seconds, which implies that a robber needs t_i seconds to acquire all the belongings of him/her. Moreover, one robber can only rob one person at a time, so if a robber would like to rob the first and second people, then it will take him $t_1 + t_2$ seconds.

The robbers are going to rob all the people in the alley. Since they cannot change their relative positions, one single robber can only rob a consecutive interval of people, and all the intervals cannot overlap with one another. Also, because the robbers work as a group, the overall time the group spends on robbing all the people will be the maximum time spent by individual members. For example, if there are 3 robbers spending $5, 7, 9$ seconds robbing people respectively, then the overall time spent is 9 seconds.

(a) (4 pts) The police are going to arrive at the alley in T seconds. Therefore, the robbers should finish their crime before the police arrive. Given the value of T, please design a greedy algorithm to decide whether the robbers can rob all the innocent people before the police arrive. Your algorithm should run in $O(N)$ time, and you should justify why your algorithm satisfies the time requirement.

(b) (6 pts) Please show the correctness of your algorithm in (a) by proving that it satisfies *Greedy Choice Property* and *Optimal Substructure*.

(c) (2 pts) Let $f(T)$ be a function where T is a non-negative integer. The value of $f(T)$ is 1 if the robbers can finish their jobs within T seconds, and 0 if the robbers cannot finish. Show that $f(T)$ is a monotonically increasing (i.e. non-decreasing) function.

(d) (2 pts) Given that $N = 10$, $M = 3$, and t_i as follows, please propose a plan (i.e., how should the M robbers divide their work) and calculate the shortest possible time for the robbers to rob all of the innocent people.

$$5, 8, 2, 1, 6, 4, 10, 9, 7, 3$$

(e) (4 pts) Please design an algorithm to calculate the minimum time the robbers need to rob all the people. Your algorithm should run in $o(N^2)$ time. Note that the complexity is small-o, not big-o. Please also justify that your algorithm meets the time complexity requirement.

(2) After robbing all the people inside the lane, the leader of those robbers shouted "*Hey, the shopkeeper there, our bag can still hold some stuffs of total weight W!*"

You, a shopkeeper at the end of the lane, are still curious about why they gave you such information, but you soon realize that you'd better lock some of your goods into your vault, because the robbers will rob you in minutes!

You have N goods, numbered as 1, 2, ..., N. The i-th of them is of weight w_i and of value v_i. Due of the size limit of the vault, you can only put K ($< N$) of them into the vault. After the robber come, they will choose some goods from the rest ($N - K$) goods, which are not in the vault. Since they are experienced robbers, they will rob in the optimal way. That is, they will choose some of the remaining goods to maximize the total value V, with total weight not greater than W, just like the well-known knapsack problem.

Your goal is to minimize V, the total robbed value. Now, given N, K, W, w_1, w_2, ..., w_N, v_1, v_2, ..., and v_N.

(1) (4 pts) First, let's focus on the special case when $K = 1$. That is, you can only put one good into the vault.

By solving a knapsack problem with backtracking on the N goods, you've computed a set of goods, S, such that taking S away is one of the optimal plans for the robbers when you don't put anything into the vault.

Observe that when $K = 1$, without loss of optimality, the good you need to lock must be in S.

Please design an algorithm with time complexity $O(|S|NW)$ to calculate the minimized V for the special case $K = 1$.

(2) (8 pts) Please design an algorithm to calculate the minimized V in $O((N + W)^3)$ time for the general case.

Problem 6 - The ADA Chocolate Factory (Hand-Written) (12 points)

Zolution, a TA of ADA, is also a manager of a chocolate company named `ADA Chocolate`. His secret recipe, *love*, soon made his company become the world's biggest chocolate company. `ADA Chocolate` is well known for its perfect square chocolate, which is composed of $N \times N$ cells.

Due to greed, a vicious TA *Handsome* established another chocolate company `AD4 Ch0co1ate` to compete with him. With all dirty tricks, such as smear, hacking, and faking, *Handsome*'s company caught up with *Zolution*'s, and became one of the top three chocolate companies.

Being aware of the competitive threat of *Handsome*'s business, *Zolution* decided to build a brand-new, cutting-edge chocolate factory with high technology and pioneering infrastructure. However, building such a new factory takes time and money, and in the meanwhile *Zolution* wants to expand his profits as soon as possible. Therefore, he would like to maximize the profits earned when the new factory is still under construction.

There are N necessary components $x_1, x_2, ...x_N$ to be constructed and installed in the new factory. Each component can increase the output of the factory. The more components are installed, the higher output the factory can achieve. Yet, the components cannot be built simultaneously. The i-th component will take you t_i full days to construct, and will make the factory produce v_i more chocolates every day after the installation. Once a component is constructed, it will be immediately installed at the end of the day. Initially, the factory cannot produce any chocolate.

(1) (2 pts) Consider that there are only two components x_1, x_2, whose time cost and extra chocolate productivity are t_1, t_2 and v_1, v_2 respectively. Which component will you prefer to build first? Briefly explain your reason.

(2) (5 pts) Please design an algorithm to calculate the maximum total number of chocolate produced from day 1 to day Σt_i. Your algorithm should run in $O(n \log n)$. You need to explain why your algorithm meets the time complexity requirement.

(3) (5 pts) Please prove the correctness of your proposed algorithm in (2). Note that if you use Greedy or Dynamic Programming, you should also prove their properties.