Damaged Tags

locked

Problem Submissions Discussions

Time Limit: $C/C++$ $(3s)$, $Java$ $(6s)$
Memory Limit: $512MB$

The *FRBD(Fun Research & Development)* is going to arrange a big event coming up. For this event, the management purchased several of Samsung's latest Smart devices. Each of these devices has a unique long-digit identifier in the device tags. There is a client app where those smart devices can be registered and used. When a device is booted up and placed into pairing mode, it appears in the nearby devices list within the app, displaying the last 4 digits of its unique identifier for identification purposes.

For the setup, all of the newly purchased devices have been turned on and placed into pairing mode simultaneously. The client app now shows the list of 4-digit identifiers of all nearby devices(ready to be paired) including all newly purchased devices and old devices which are in pairing mode. So, the nearby list can be much larger than purchased devices list. As the identifiers are showing, so anyone can easily select their newly purchased devices from the nearby list by matching identifiers.

Problem begins when management find most of the devices tags are damaged somehow. Some identifiers are visible partially.

Each purchased device's identifier (as shown in the app) is represented by a 4-character string, which contain:

- **Digits (0-9)** representing specific positions in the identifier.
- **Wildcards (?)** representing damaged digits that can match any digit (0-9) in the corresponding position.

A purchased device with wildcards can match multiple nearby devices. For example, the identifier $?78?$ can match both 0780 and 9788. It is also possible for a fully visible identifier from purchased devices list to be matched with multiple identifiers from nearby devices list, as app is showing only last 4-digit of the identifiers. Now the management wants to determine how many distinct matching sets can be formed between purchased and nearby devices, subject to the following constraints:

- Each purchased device must match **exactly** one nearby device.
- A nearby device can match **atmost** one purchased device in a set.
- All purchased devices must be included in each set.

(see sample explanation for better understandings)

Management needs your help to find the total number of valid sets.

Input Format

The first line contains an integer T denoting the number of test cases.

Every test case starts with a integer P denoting the number of purchased devices.

Each of the next P lines contain a 4-length string denoting each of purchased device's identifier. Each string will consists of either '?' or '0'-'9'(digits) or combination of both.

Next line contains another integer N denoting the number of nearby devices.

Each of next N lines contain a 4-length string denoting each nearby devices's identifier. Each string will consist of '0'-'9'(digits).

Constraints

$1 \leq T \leq 10$

$1 \leq P \leq 15$

$1 \leq N \leq 100$

Use Faster I/O Methods.

Output Format

For each test case, you need to print the number of distinct purchased device sets matched from nearby devices list. As the number can be very large, print it with **modulo 1000000007**(10^9 + 7).

Sample Input 0

```
1
2
?7??
????2
5
5607
2742
8754
4724
8672
```

Sample Output 0

```
5
```

Explanation 0

Here, 2742, 8754 and 4724 from nearby devices list can be chosen in case of ?7?? purchased device and 2742, 8672 from nearby devices list can be chosen in case of ????2 purchased device. Distinct matching sets will be {2742, 8672}, {8754, 2742}, {8754, 8672}, {4724, 2742} and {4724, 8672}. So, The number of distinct sets will be 5.