Problem 1. Arrange Students

(Time Limit: 2 seconds)

Problem Description

Grace is a teacher in an elementary school. She would like to line up students in her class by height in ascending order. Unfortunately, these young students are difficult to control, and lining up them is not an easy job. In order to reduce the chaos, she decides to sequentially call up students by their IDs one by one, and each called student could (1) be added to the beginning of the line, (2) be added to the end of the line, or (3) not be put into the line. With the lining-up rules, find the length of the longest possible line.

Input Format

The first line contains an integer $0 \leq n \leq 3000$, the number of students. The following n lines contain n positive integers representing the heights of the n students, correspondingly, sorted by IDs. None of any two students have the same height.

Output Format

The output is the length of the longest line.

Example

Sample Input:	**Sample Output:**
3 101 124 103	2

Problem 2. One-Dimensional Tiling

(Time Limit: 5 seconds)

Problem Description

The *one-dimensional tiling problem* is to tile an $1 \times n$ floor with different types of 1×1 tiles. In this problem, we assume that there are five types of tiles as follows: the letter in the center of each type of tile denotes the name of the type and the numbers on its left and right sides are used to ensure the compatibility of consecutive tiles. We assume the upward compatibility from the left to the right between consecutive tiles. That is, if *x* and *y* are tile together, then the number on the right side of *x* must be less than or equal to that on the left side of *y*. Hence, for example, the tiling *bbdca* is allowed but *bbcda* is forbidden. How many different ways are possible to tile an $1 \times n$ floor with the five types of tiles?

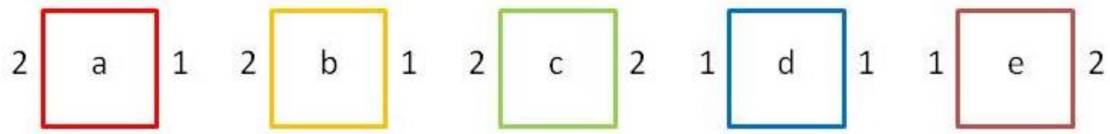

Figure 1: Different types of tiles

Technical Specification

- The length of the floor is a positive integer *n* such that $0 \leq n \leq 30$.

Input Format

The first line of the input consists of a single number *k* denoting the number of test cases and the second line consists of a sequence of *k* positive integers separated by space to denote the input of each test case.

Output Format

For each test case with input *n*, compute the number of different ways to tile the $1 \times n$ floor with the five types of tiles. Output the answers one by one in *k* lines.

Example

Sample Input:	Sample Output:
2 2 3	21 89

Problem 3. Rosetta Oracles

(Time Limit: 3 second)

Problem Description

The Rosetta Stone is a rock stele inscribed with a decree issued at Memphis, Egypt, in 196 BC. The stone, carved in black granodiorite, is believed to have originally been displayed within a temple. It was the first Ancient Egyptian bilingual text recovered in modern times, and it aroused widespread public interest with its potential to decipher this previously untranslated hieroglyphic language. The stone was initially thought to be a fragment of a larger stele, owing to its damaged state and none of the texts is readable. In fact, the text was later found to be a circular encrypted string composed of both meaningful vocabulary words and false ones, in order to hide the oracle of the Rosetta Stone. The only clue to decipher the Rosetta oracle is that the frequency of meaningful word is larger than that of false one in the original circular text. Therefore, it is necessary to estimate the frequency of each vocabulary word from the encrypted circular string.

Although nobody knows how to decipher the encrypted circular string, the encryption method has been discovered. Both of the original and encrypted circular string are from standard alphabet {a-z}. The encryption of the original circular string was done by first generating all clockwise rotations of this circular string. For example, given a circular string {banana}, all possible clockwise rotated strings at the same length are {banana}, {abanan}, {nabana}, {anaban}, {nanaba}, and {ananab}. These rotated strings are sorted lexicographically, leading to {abanan}, {anaban}, {ananab}, {banana}, {nabana}, and {nanaba}. The last letters in these sorted strings, i.e., {nnbaaa} forms the encrypted circular string shown on the Rosetta stone. In order to decipher the oracle message hidden in the Rosetta stone, a simple method is brute-force query of the frequency of each vocabulary word and collecting the words with higher frequency, which are more likely to be the meaningful words used in the oracle. However, since the circular string is encrypted, we need a smart person to help us achieve this goal.

Given the encrypted circular string (e.g., nnbaaa) and a number of vocabulary words (e.g., aba), write a program that answers the (clockwise) frequency of each word in the original circular string (e.g., 1). Note that the frequency may be zero if the word does not appear in the original circular string.

Technical Specification

- The letters in the original/encrypted strings and query vocabulary are from standard alphabet {a-z}.
- The lengths of original and encrypted strings L range from 1 to 30000.
- The length of each query vocabulary W ranges from 1 to 30000 and $W \leq L$.
- The number of queries N for each encrypted string range from 1 to 10.

Input Format

The first line contains the number of test cases. For each test case, the first line is the encrypted circular string. The following line contains the number of queries N for this test case. Each of the following N lines stores the query string.

Output Format

Each line contains the clockwise frequency of each query vocabulary corresponding to each test case.

Example

Sample Input:	**Sample Output:**
2 nnbaaa 1 aba pssmipissii 2 s mmi	1 4 0

Problem 4. Diffusion time minimization problem

(Time Limit: 3 seconds)

Problem Description

A boss wants to promote his product through social networks via viral marketing. After doing the market research, the boss has found a set of target audiences who are interested in his product. Here, a target audience is a person in the social network with some specified attributes. The attributes maybe include locations (e.g., living in Tainan) or demographics like age, gender and interests. For instance, a typical target audience of Mercedes Benz is a high-income male who is 40-60 years old.

Now the boss wants to find a set of *k* people in the social network so that they can diffuse the advertisement of his product to the target audiences through the network within the shortest time period. In the problem we denote the person who delivery the advertisement to others as a seed.

The social network in this problem is defined as a graph *G*(*V*, *E*). A person in the social network is considered as a node *v* in *V*. If a person v_1 who follows another person v_2, then there is a directed edge from v_2 to v_1. On the other hand, we say that v_1 is a follower of v_2. The weight on the edge is the length of the time duration from the time v_2 posts the advertisement on the social network to the time v_1 sees the advertisement.

Given you a social network *G*, a set of target audiences *T*, *k* seeds *S*, and assume that all the seeds post the advertisement at the same time. Write a program to determine the shortest diffusion time. That is, the shortest time period from the time all seeds post the advertisement to the time that all target audiences see the advertisement.

Technical Specification

- The number of seeds is k, $1 \leq k \leq 200$
- The number of nodes in the social network is at most 10000.
- v_i is the node id of the i-th node in the social network and $0 \leq i \leq 9999$.
- $1 \leq$ the weight on an edge ≤ 800.
- $1 \leq$ the number of target audiences ≤ 5.
- $0 \leq$ the number of followers of a node ≤ 16.

Input Format

There are several test cases in the input. Each test case consists of three parts. Each part is terminated by a line containing a space.

The first line of the first part is an integer indicating the number of nodes in the social network. The following M lines contain the information of the social network. Each line contains 3 integers delimited by a space. The first two integers are the node ids. The third integer is the weight between two nodes. For example, "1 2 100" represents that node 1 takes 100 time units to deliver an advertisement to node 2.

The second part is a line containing several integers. The integers are delimited by a space. Each integer denotes a node id of a target audience.

The third part is the information of the requirements. Each test case has several requirements. The first line of a requirement is an integer (denoted as v) which indicates the number of requirements ($1 \leq v \leq 5$). Each requirement consists of two lines. The first line is the value of k ($1 \leq k \leq 6$). The second line contains k integers delimited by a space. Each integer in the second line is a node id of a seed.

Output Format

For each requirement, please output the shortest diffusion time on a line.

Example

Sample Input:	**Sample Output:**
5 0 1 96 0 2 14 1 2 37 2 3 19 3 4 75 1 1 2 0 3 5 0 1 57 0 2 48 1 2 90 2 4 19 3 4 52 2 1 2 0 3	96 96

Problem 5. #3SAT

(Time Limit: 5 seconds)

Problem Description

In propositional logic, we are given a set of variables x_1, x_2, …, x_n. Each variable can take value 1 or 0, which stands for *true* and *false*, respectively. These variables can be used to construct more complicated structures which are defined as follows. A *literal* is either a variable x_i or its negation $-x_i$. A *clause* is the *disjunction* (i.e. logical-or) of literals. If we assign 0 or 1 to all variables, we can evaluate the *truth value* of any clause by applying the following rules. A *positive literal x* takes the same value as variable x; a *negative literal* $-x$ takes value $1 - x$; and finally, a clause has value 1 if and only if *at least one* of its ingredient literals has value 1. For example, suppose we have three variables x_1, x_2 and x_3. According to the above definition, both x_3 and $-x_2$ are literals. Both $\{-x_1, x_2, x_3\}$ and $\{x_3, -x_2\}$ are clauses. If we assign $x_1 = 1$, $x_2 = 0$ and $x_3 = 0$, we would get the values of clauses $\{-x_1, x_2, x_3\}$ and $\{x_3, -x_2\}$ to be 0 and 1, respectively. Your job is to write a program to count the total number of truth assignments that can make *all* clauses specified in a test case to have value 1.

Technical Specification

- There are at most 10 test cases.
- For each test case, there are *n* variables, where $1 \le n \le 33$, and there are *m* clauses, where $1 \le m \le 200$.
- The number of literals in a clause is at least 1 and at most 3.

Input Format

The first line of the input contains an integer indicating the number of test cases. The first line of each test case contains two integers *n* and *m*, separated by a space. In the following *m* lines, each line specifies a clause. Each clause is a sequence of distinct nonzero numbers between –*n* and *n* ending with 0 on the same line. Positive numbers denote the corresponding positive literals; negative numbers denote the negation of the corresponding variables. There are no opposite literals *x* and –*x* occurred simultaneously in the same clause.

Output Format

Please output the total number of distinct truth assignments that can make all clauses to be true for each test case.

Example

Sample Input:	Sample Output:
3 2 4 1 2 0 1 -2 0 -1 2 0 -1 -2 0 3 2 3 1 2 0 -1 -2 -3 0 33 1 4 0	0 6 4294967296