제 3회
전국 대학생 프로그래밍 대회 동아리 연합 여름 대회

후원

KAKAO

출제 및 채점

ALGOSPOT

그리고

박성진, 윤형석, 이준성

2013년 8월 17일

문제 A 부터 K 까지, 총 19 페이지

Problem A. Alchemy

Alchemists are mysterious people who try to synthesize valuable materials (such as gold) out of mundane materials (such as copper or lead). According to legends, their work has culminated in a cryptic piece of book, called the Emerald Tablet. The Emerald Tablet is said to contain the ultimate recipe — for synthesizing gold, and the Elixir of Life!

By a lucky coincidence, you found an ancient book that you suspect to be the Emerald Tablet. This book deals with N kinds of materials, each of them numbered with unique integers from 1 to N. Each material is either:

- Synthesizable: synthesizable materials can be made from mixing other materials (possibly synthesizable or non-synthesizable) listed in this book. We would be rich if gold was synthesizable, and we would be immortal if Elixir of Life was!
- Non-synthesizable: non-synthesizable materials cannot be made from other materials, so you'll have to harvest or buy them.

The book lists all of the N materials that are known to the world, both synthesizable and non-synthesizable. The book also provides a recipe for each of the synthesizable material. A recipe specifies a list of the ingredients need to synthesize the target material, where each ingredient is one of the N materials.

To synthesize a (synthesizable) material, you need to put the ingredients of the target material into the "Mysterious Flask", **one at a time**. The Mysterious Flask will take care of mixing the ingredients. As soon as you put the last ingredient into the flask, the target material will be synthesized, and pop out of the flask magically.

This all sounds very simple, and we seem to be on our way to infinite wealth and eternal life, but there is a problem in practice; you may not be able to synthesize what you want!

For example, consider the following case: there are two synthesizable materials, X and Y. X can be made by mixing A and C in the flask, and Y can be made by mixing A, B, C and D. In order to make Y, you can try to put the 4 ingredients into the flask in the following order: A, B, C and D. However, the moment you put C in the flask, X will be synthesized in the flask, and pop up!

You would think you can avoid this problem by putting the ingredients in a different order, such as A, B, D, and then C. Well, after you put C into the flask, it contains the ingredients of Y as well as those of X — which of the two will be synthesized in the flask, then? After numerous experiments, you have finally figured out the following rule:

- When there are two or more materials that can be synthesized (from the ingredients in the flask), the Mysterious Flask mixes and synthesizes the one that needs the largest number of ingredients.
- In case of a tie, the material with the largest number (which has the highest reactivity) will be synthesized.

With the above rule in effect, Y will be synthesized as it requires 4 ingredients while X requires 2 ingredients, in our example.

Unfortunately, the recipes in the Emerald Tablet do not tell us in what order we should put the ingredients into the flask. But we can't stop here! To gain infinite wealth and eternal life, write a program that finds an ordering of ingredients that can synthesize the desired material for each recipe.

Input

The first line of the input will contain the number of test case, T.

Each test case starts with a line containing N ($1 \leq N \leq 16$), the number of known materials. N following lines will give information of each material, from 1 to N. The $i+1$-th line contains an integer c_i which is the number of ingredients needed for synthesizing material i. Then c_i distinct integers follow, which are the number of ingredients needed (none of them equals i). If c_i is 0, material i is non-synthesizable.

Test cases are separated by a blank line.

Output

For each test case, print N lines. Each of this lines should be a reordered recipe for each of the materials, from 1 to N. Each line should begin with c_i, and c_i numbers should follow which is the order of the ingredients. If the material was listed as synthezable, but cannot be synthezed, print a line containing "`IMPOSSIBLE`" (without quotes) instead. If there are multiple orderings that can synthesize a material, print any of them.

Separate adjacent test cases by a blank line.

Sample input and output

Standard Input	Standard Output
3 3 0 1 1 1 1 6 0 0 0 0 2 1 2 4 1 2 3 4 5 0 0 1 5 2 1 3 2 2 3	0 IMPOSSIBLE 1 1 0 0 0 0 2 2 1 4 4 3 2 1 0 0 1 5 2 3 1 2 3 2

Problem B. Bunker Rush

'BoxeR' Lim, Yo-Hwan(임요환) and 'Yellow' Hong, Jin-Ho(홍진호) will face each other at the StarCraft 6 Championship in 2022!

StarCraft is a real-time strategy video game series. The series consists of many gameplay modes, but usually two players compete to destroy each other. Players choose one of many strategies to play each match, for example, "Bunker Rush" or "Hidden Barracks". Sometimes, the result of a match is trivially determined based on players' strategies like a rock-scissors-paper game. So in professional tournaments, it is usual to play more than a single match to declare who is victorious.

BoxeR has a peculiar style of choosing his game plans: he chooses a sequence of strategies, and repeats them over multiple games.

Match No.	Strategy
Game 1	Bunker Rush
Game 2	Bunker Rush
Game 3	Bunker Rush

For example, the above table shows how BoxeR chose to use one strategy (the Bunker Rush) three times in a row (in a best-of-five series). In this case, the sequence was of length 1 and was repeated three times.

Match No.	Strategy
Game 1	Marine Rush
Game 2	Bunker Rush
Game 3	Double Command Center
Game 4	Bunker Rush
Game 5	Double Command Center
Game 6	Dropship Rush
Game 7	Dropship Rush

The above table shows another example. In this case, BoxeR played "Marine Rush" once, and he repeated a sequence of length 2 twice - "Bunker Rush" and "Double Command Center". Finally, he repeated a sequence of length 1 twice, which was "Dropship Rush".

An average player would notice if his opponent repeats the same sequence of strategies more than two times, and will not get fooled the third time. However, Yellow has had bad games against BoxeR before, and now Yellow is determined to be not fooled again.

As such, Yellow is interested in finding all sequences of strategies that BoxeR used three times consecutively in the past. You are to write a program that tells Yellow how many such sequences exist as well as the length of the longest such sequence, given a complete log of strategies used by BoxeR in his career.

Please help Yellow prepare for the championship series!

Input

The first line contains the number of test cases, T.

For each test case, a single line contains a string that describes the complete log of the strategies used by BoxeR. Each character is an English letter (either lower-case or upper-case) that represents a single-game strategy. The total number of games of BoxeR is no more than 10,000 for each test case.

Output

For each test case, output two integers in a single line, separated by a single white space: the number of all sequences of strategies that BoxeR repeatedly used three times, and the length of the longest such sequence.

Note that if the same sequence of strategies was repeatedly used at different times, then it must be counted multiple times, once per its starting index (see the sample test case for clarification).

If no sequence of strategies was repeated three times or more, output two '-2's, separated by a single space (quotes for clarification only).

Sample input and output

Standard Input	Standard Output
3 aabcbcbcbcbcbcbcbc AaABABAAAB abcdefabcdef	16 4 1 1 -2 -2

Notes

Clarification for the sample input and output:

1. There are four different sequences of strategies that BoxeR repeated three times. One is '`bc`' that starts and ends at the indices [3, 8], [5, 10], [7, 12], [9, 14], [11, 16], and [13, 18]. Another is '`cb`' that starts and ends at the indices [4, 9], [6, 11], [8, 13], [10, 15], and [12, 17], and '`bcbc`' that starts and ends at the indices [3, 14], [5, 16], and [7, 18]. The other is '`cbcb`' that starts and ends at the indices [4, 15] and [6, 17]. Hence, the total number of sequences of the interest is 16 and the length of the longest one is 4 due to '`bcbc`'.
2. The only sequence that was repeated three times is '`A`' at index 6. Hence the correct answer is '1 1'.
3. In this case no sequence was repeated three times.

Problem C. 용감한 오리

어느 더운 여름날, '모두의 오리'라는 오리 농장에서 복날을 맞이해 농장 주인이 1번 오리부터 차례대로 트럭에 집어넣고 있었다. 그러나 용감한 오리 Pekaz가 농장으로부터 탈출하고 말았다 (후일 이 오리는 0번째 오리라고 알려졌다).

Pekaz가 농장으로부터 완전히 벗어나기 위해서는 커다란 강을 건너야 했다. 불행하게도 최근 장마로 인해 물살이 너무 거세져 제대로 헤엄을 칠 수 없기에, 그 대신 강 위에 설치된 돌다리를 이용해 건너기로 했다.

여기서 강은 2차원 평면으로, 돌다리를 구성하는 각 돌은 그 위의 점으로 보기로 하자. 우리의 Pekaz는 돌다리가 아닌 강 위의 특정 지점에서 출발해 목표 지점까지 이동하려고 한다.

Pekaz는 언제건 자신이 위치한 곳에서 직선거리가 J 이하인 곳으로 뛰어서 이동할 수 있다. 이 J를 최대 점프력이라 하자. 뛰어오르는 횟수에는 아무런 제약이 없지만, 만약 돌다리가 아닌 곳으로 뛸 경우 거센 물살에 휩쓸려 내려가 버리고 말 것이다.

과연 우리의 Pekaz는 탈출에 성공할 수 있을까? 아니면 안정적인 맛의 오리고기가 되고 말 것인가? 돌다리의 구성과 점프력이 주어질 때, Pekaz의 탈출 계획이 성공할 수 있는지 알아내는 프로그램을 작성하라.

Input

입력은 T 개의 테스트 케이스로 이뤄지며, 입력의 첫 줄에는 T가 주어진다.

각 테스트 케이스의 첫 번째 줄에는 Pekaz의 최대 점프력인 정수 J ($1 \leq J \leq 1000$)가 주어진다. 두 번째 줄과 세 번째 줄에는 각각 Pekaz의 시작 지점의 좌표와 도착 지점의 좌표가 주어지고, 네 번째 줄에는 돌다리를 구성하는 돌의 개수 N ($0 < N \leq 100$)이 주어진다. 그다음 줄부터 N개의 줄에 걸쳐 각 돌의 좌표가 주어진다. 모든 좌표는 공백으로 구분된 두 정수 x, y ($-1000 \leq x, y \leq 1000$) 꼴로 주어지며 시작 지점, 도착 지점 및 각 돌의 좌표들이 동일한 경우는 주어지지 않는다.

Output

각 테스트 케이스마다 한 줄에 하나씩 탈출이 가능하면 "`YES`", 불가능하면 "`NO`"를 출력한다.

Sample input and output

Standard Input	Standard Output
2 2 1 1 4 1 3 4 2 1 2 3 2 3 1 1 10 10 5 6 7 4 1 9 7 6 4 4 4	YES NO

Problem D. 어서와, 영문이름은 처음이지?

전국민의 메신저 카카오톡으로 유명한 카카오는 다른 회사와는 다른 여러 문화가 있다. 그 중 하나로, 상하 관계에서 오는 문제를 줄이기 위해 입사할 때 영문 이름을 정하고 직책과 관련 없이 서로를 영문 이름으로 부르는 문화가 대표적이다. 이러한 영문 이름 덕분에 카카오에선 서로 간의 커뮤니케이션이 경직성 없이 친근하게 이루어지고 업무 상의 의견 또한 자유롭게 주고받을 수 있는 분위기가 정착되었다.

하지만 회사 규모가 점점 커짐에 따라 신규입사자들이 자신이 원하는 영문 이름을 사용하지 못하는 경우가 많이 발생하게 되었고, 인사 담당자가 일일이 전화를 걸어 "해당 이름이 사용 중이니 다른 이름으로 해주세요."라고 얘기하는 것도 슬슬 한계에 다다르게 되었다.

이 현상을 지켜보던 개발자 I모씨는 한가지 제안을 했다. 신규입사자들의 영문 이름을 3지망까지 받은 뒤, 각 입사자들의 만족도의 총합을 최대한 높이는 방법으로 이름을 정해주자는 것이었다. 기존 사원과 이름이 중복되거나 다른 신규입사자와 이름이 중복될 경우, 비록 만족도가 적겠지만 영문 이름 뒤에 온점(.)을 찍고 영문 성을 붙인 이름을 사용할 수 있다. 카카오에서는 이런 방법을 통해 최대한 이름이 중복되는 것을 막아서 불필요한 인사팀의 노고와 신규입사자들의 이름 고민에 대한 고통을 줄여보기로 하였다.

그리하여 신규입사자들에게 자신이 원하는 3개의 영문 이름을 써서 내도록 하고, 각각 이름에는 자신이 느낄 만족도를 0 이상 10 이하의 정수 수치로 표현하여 제출하도록 했다. 만약 영문 이름 뒤에 영문 성이 붙을 경우 그 사람의 만족도는 원래 써 낸 만족도의 절반(1/2)이 된다고 한다.

기존 사원들의 이름과 신규입사자의 지원 서류가 주어질 때, 신규입사자들의 만족도의 합이 최대가 되는 이름 영문 이름들을 할당해보자.

Input

입력의 첫 줄에는 기존 사원 수 K가 주어진다. ($K \leq 400$) 두 번째 줄에는 기존 사원 K명의 영문 이름이 공백으로 구분되어 주어진다. 기존 사원에 대한 정보는 모든 테스트 케이스에 공통으로 적용된다.

세 번째 줄에는 테스트 케이스의 수 T가 주어진다. 각 테스트 케이스의 첫 줄에는 신규입사자의 수 N ($N \leq 50$)이, 그 다음 N 줄에는 각각의 신규 입사자가 제출한 성 L_i, 첫 번째로 희망하는 영문 이름 $N_{i,1}$과 그 만족도인 $S_{i,1}$, 두 번째로 희망하는 영문 이름 $N_{i,2}$와 그 만족도인 $S_{i,2}$, 그리고 세 번째로 희망하는 영문 이름 $N_{i,3}$과 그 만족도인 $S_{i,3}$이 공백으로 구분되어 주어진다. 희망하는 영문 이름인 $N_{i,1}$, $N_{i,2}$, $N_{i,3}$은 모두 다르며 그 만족도는 $10 \geq S_{i,1} \geq S_{i,2} \geq S_{i,3} \geq 0$ 임이 보장된다.

입력에서 주어지는 모든 영문 이름은 20자 이하의 영문 대소문자로만 구성된 문자열이다. 각 테스트 케이스는 다른 테스트 케이스에 영향을 주지 않는다.

Output

각 테스트 케이스 마다 한 줄에 하나씩 신규입사자들의 만족도 총합의 최대값을 출력한다. 모든 신규입사자들에게 고유한 영문 이름을 할당할 수 없는 경우는 없다고 가정해도 좋다.

Sample input and output

Standard Input	Standard Output
8 Inoran Glen Chloe Steven Robby Dante Sheldon Ian 2 3 Kim Ian 9 Roy 1 Alain 1 Heo Ian 5 Gilbert 3 Adrian 0 Park Steven 10 Sheldon 9 Andrew 8 3 Han Vex 1 Linda 1 Lesile 1 Kwon Vex 4 Linda 3 Lesile 2 Lee Vex 5 Linda 4 Lesile 3	15.5 9

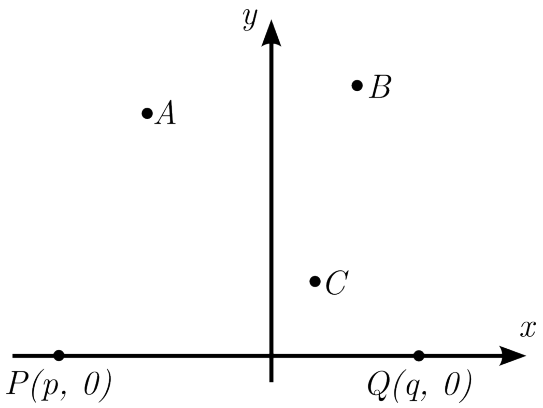
Notes

첫번째 테스트 케이스의 경우 Ian, Steven, Sheldon은 기존 사원의 이름과 중복이 되므로 .(온점)과 함께 영문 성을 붙여야 하고 만족도는 원래 만족도의 절반(1/2)이 되어 다음과 같게 된다.

- 입사자1 : Ian.Kim 4.5 Roy 1 Alain 1
- 입사자2 : Ian.Heo 2.5 Gilbert 3 Adrian 0
- 입사자3 : Steven.Park 5 Sheldon.Park 4.5 Andrew 8

따라서 신규입사자들의 만족도 합의 최대값은 4.5 + 3 + 8 = 15.5가 된다.

Problem E. Annie and Tibber



애니와 티버는 x축 위에서 여행을 하고 있었다. 애니의 위치는 $P(p, 0)$이고, 티버의 위치는 $Q(q, 0)$이다. 어느 날 문득 하늘을 바라본 애니와 티버는 서로 이야기를 하던 도중 별들 사이의 좌우관계에 흥미를 가지게 되었다.

애니: A별이 B별보다 왼쪽에 있네!
티버: 응!
애니: 그리고 B별은 C별보다 왼쪽에 있네!
티버: B가 C보다 오른쪽에 있는것 같은데?
애니: 그래? 내가 봤을때는 왼쪽에 있는데?!

여기서 점 X에서 보았을 때 A별이 B별보다 왼쪽에 있다는 것은, $X \rightarrow B \rightarrow A$가 왼쪽으로 꺾였다는 것을 의미한다. 애니는 하늘에 있는 모든 별들의 쌍 중에서 티버가 보았을 때와 좌우관계가 반대인 쌍이 몇 개나 되는지 알고 싶어졌다. (A, B) 쌍은 (B, A) 쌍과 같은 것으로 보아 한 번씩만 센다. 모든 별들은 애니가 보았을때도 직선상에 둘 이상 있지 않고, 티버가 보았을때도 직선상에 둘 이상 있지 않다.

Input

첫 줄에 테스트 케이스의 수 T가 주어진다. 각 테스트 케이스마다 첫 번째 줄에 애니와 티버가 관찰한 별의 수 $N(1 \leq N \leq 100,000)$과 애니의 위치와 티버의 위치를 표현하는 두 정수 p, q 가 주어진다.

그 다음 줄부터 N줄에 걸쳐 애니와 티버가 관찰한 별의 좌표를 표현하는 두 정수 x_i, y_i 가 주어진다. 애니, 티버의 위치와 모든 별의 x 좌표는 -10^6 이상 10^6 이하의 정수이다. 모든 별의 y 좌표는 1 이상 10^6 이하의 정수이다.

Output

각 테스트 케이스에 대해서 애니와 티버가 보았을 때 서로 좌우관계가 반대인 쌍의 수를 출력한다.

Sample input and output

Standard Input	Standard Output
1 3 -5 5 -2 4 2 5 1 1	2

Problem F. 강시대소동

강시는 중국의 흡혈귀이자 일종의 좀비로서 어떤 힘에 의해서 되살아나 피해자의 기를 빼앗거나 피해자를 물어 같은 강시로 만들어 해치는 존재이다. 본래는 객지에서 죽은 자들의 원혼이 깃든 시체를 강시라 하는데, 고향으로 이관하기 위하여 영환술사가 부적을 붙여 움직일 수 없도록 한 후 고향으로 돌려 보내었다고 한다. 시체이므로 몸이 굳어 관절을 구부리지 못하여서 영환술사의 종소리에 맞추어 뛰면서 이동한다고 전해진다.

오래도록 강시에 대해 연구한 한 학자에 의하면, 강시는 알려진 바와 달리 다음의 세 가지 이동 법칙에 의해서만 뛰어다닌다고 한다:

1. 어떤 구간을 계속 순환하기만 한다.
2. 동서남북의 네 방위에 있는 사신들(청룡, 백호, 주작, 현무)의 기운을 이용해서 움직이기 때문에, 맨하탄 거리가 항상 L인 곳으로만 뛰어다닐 수 있다. 두 점 $P(x_P, y_P)$와 $Q(x_Q, y_Q)$ 사이의 맨하탄 거리는 $|x_P - x_Q| + |y_P - y_Q|$로 정의된다.
3. 또한 강시들은 뛸 때 자신이 출발할 당시 자신이 바라보던 방향과 90도 이내의 각도를 이루는 곳으로 뛰어다닐 수 있다. 그뿐만 아니라, 몸의 회전 방향은 이동 방향과 상관없이 조절할 수 있어서 뛰는 도중에 몸을 90도 이내의 범위에서 회전할 수 있다. 예를 들어, 강시가 정동쪽 방향을 보고 있을 때, 정북쪽 방향으로 뛰는 동시에 도착 지점에서 정남쪽 방향을 보고 있을 수 있다.

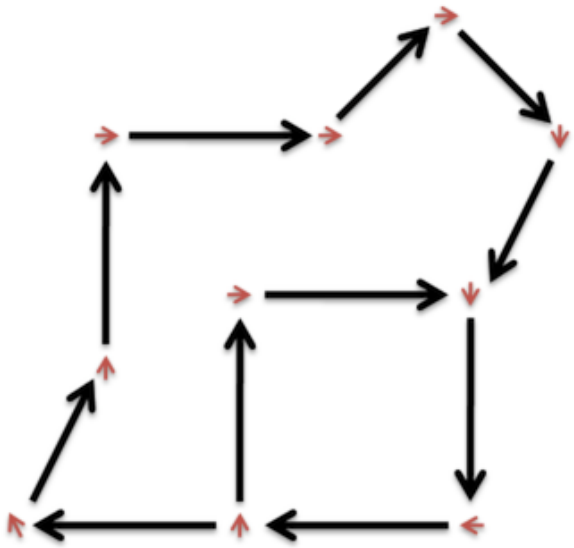


그림. 강시의 이동법칙에 따른 이동경로 예시(작은 화살표는 발자국의 방향)

윤하가 사는 어느 마을에 갑작스레 강시가 나타났다! 마을의 사람들은 윤하를 지키기 위해 강시 전문가인 여러분을 찾아왔다. 여러분은 찍힌 발자국들을 통해 강시의 경로를 추측하기로 하였는데, 사람들의 발자국과 강시들의 발자국이 함께 뒤엉켜있어 이를 구분하기 어렵다.

발자국을 토대로 추측했을 때 강시의 수가 지나치게 많으면 조사가 너무 어려워지므로, 이를 해결하기 위해 발자국으로부터 존재할 수 있는 최소한의 강시의 수를 찾아내는 프로그램을 작성하라.

Input

입력은 T 개의 테스트 케이스로 구성된다. 입력의 첫 줄에는 T가 주어진다.

각 테스트 케이스의 첫 줄에는 두 정수로 강시의 이동거리 $L(0 < L \leq 10)$과 발자국의 개수 $N(1 \leq N \leq 50,000)$이 공백으로 구분되어 주어진다. 이후 N개의 줄에 걸쳐 각 줄마다 각 발자국의 정보를 의미하는 세 정수 x, y, t가 공백으로 구분되어 주어진다. (x, y)는 발자국의 위치이고, t는 발자국의 방향을 x축으로부터의 동경으로 표현한 60분법 각도이다. $(-500,000 \leq x, y \leq 500,000, 0 \leq t < 360)$ 즉 t가 0이면 동쪽, 90이면 북쪽, 180이면 서쪽, 270이면 남쪽으로 향한 발자국을 의미한다. 같은 위치에 여러 발자국이 존재하는 경우는 없다.

Output

각 테스트 케이스마다 한 줄에 하나씩 존재할 수 있는 최소한의 강시의 수를 출력한다.

Sample input and output

Standard Input	Standard Output
2 10 5 0 0 0 0 10 90 10 10 180 10 0 270 20 0 270 10 7 0 0 45 10 0 135 10 10 225 0 10 315 40 40 315 45 45 315 5 5 0	1 2

Problem G. Colorful Necklaces

You have a bunch of beads, and each bead is coated with one of C colors. You would like to connect them into necklaces. However, you want to make sure each of such necklaces will have no two beads of the same color (in other words, all beads in a single necklace should have distinct colors), because you think that is less boring. On top of it, each necklace should use exactly N beads.

Your plan is to make as many necklaces as possible. Write a program that calculates the maximum number of necklaces you can make, given the number of beads you have for each of the C colors.

Input

The first line of input contains T, the number of test cases. Each test case starts with a line that contains N ($1 \leq N \leq 10,000$) and C ($1 \leq C \leq 10,000$). N is the number of beads needed for each necklace and C is the number of colors of the beads. The next line contains C numbers, denoted by $b_1, b_2, ..., b_C$ where b_i is the number of beads you have of the color i ($1 \leq b_i \leq 10^9$).

For each test case, the total number of beads you have is always a multiple of N and is no greater than 10^9.

Output

For each test case, your program must output the maximum number of necklaces you can make.

Sample input and output

Standard Input	Standard Output
2 3 3 3 3 3 3 3 2 3 4	3 2

Problem H. Inventory

Diablo 3 (which is so out of fashion now that only oldies play it) is all about collecting good items. In this game, each player has a bag organized as a grid with 6 rows and 10 columns.

이래봐야-득은-없습니다.jpg

As you can see in the picture above, all items are of size 1×1 or 2×1. Items cannot overlap or be rotated. The game has a variety of items with different sizes, but in this problem we only consider two kinds of items: rings (of size 1×1) and daggers (of size 2×1). Furthermore, the game allows multiple items of the same kind to be "stacked", but we will ignore that feature in this problem.

When the player picks up a new item, the item is placed in the bag according to the following simple algorithm dependent on the size of the item:

- If the new item's size is 1×1: Starting from the left top cell of the bag, look for an empty cell from top to bottom first, and then left to right. The item is placed into the first empty cell found.
- If the new item's size is 2×1: Starting from left top cell of the bag, look for two vertically adjacent empty cells from left to right first, then top to bottom. The item is placed into the first 2-vertically-adjacent-empty cells found.

Because of the algorithm above, you might not be able to pick up an item even if the bag is not full, if you pick up items in the wrong order. Consider the following case with a 2x2 small bag, with an item of size 1×1 placed in the upper right cell. What happens when you pick up a ring and a dagger?

```
. X
. .
```

If you pick up the dagger first, both items would fit in the bag. However, if you pick up the ring first, there will be no room for the dagger to be placed in the bag.

Now, suppose you have a bag of size $R \times C$, and there are N rings and M daggers that you want to pick up. You would like to calculate the number of orderings that allow you to pick up all items. Assume that items of the same type are indistinguishable when counting the number of orderings, since those are unidentified yet. See the sample test case for clarification.

Input

The input consists of T test cases. T will be given in the first line of the input file, and T test cases will follow.

Each test case starts with a line with 4 integers, R, C ($1 \leq R, C \leq 15$), N and M ($0 \leq N, M \leq 225$). The next R lines will tell you the current state of the bag. Each line denotes a row in the bag, and a '.' represents an empty cell, and a 'X' represents a cell occupied with an item.

Output

For each of the test case, print the number of orderings which will let you pick up all items, MOD 20130817.

Sample input and output

Standard Input	Standard Output
3 2 2 4 0 .. .. 2 2 1 1 .X .. 3 3 3 1 XX. XX. ...	1 1 4

Problem I. Mismatched Parenthesis

You are working on a sequence of parentheses. There are four types of parentheses and the symbols used are `()`, `{}`, `[]`, `<>`. We call '`(`', '`{`', '`[`', and '`<`' the left parentheses and '`)`', '`}`', '`]`', and '`>`' the right parentheses.

The original sequence was a perfectly matched sequence. That is, every left parenthesis was matched to a right parenthesis of the same type (e.g. '`(`' and '`)`'). But by a mistake, some of the left parentheses are incorrectly transcribed to left parentheses of different type(s), and so are some of the right parentheses.

You want to modify the sequence such that the resulting sequence is a perfectly matched sequence, by correcting a mismatched pair of a left parenthesis and a right parenthesis into a matched pair of parentheses of the same type. However, the problem complicates as there is a rule you must follow: to correct a mismatched pair, you should change exactly one of the two parentheses. Also, you can only replace a lower priority parenthesis with a higher priority parenthesis – not the other way. Suppose that the type '`{}`' has higher priority than the type '`()`'. Then you must correct the mismatched pair '`(}`' to '`{}`', but not to '`()`'.

Given the mistranscribed seqeunce of parentheses and an ordering of priority of different types, write a program that fixes the given sequence. You are to find all of the mismatched pairs, and for each pair, change the parenthesis of lower-priority type to a higher-priority type.

Input

The input consists of T test cases. T will be given in the first line of the input file, and T test cases will follow.

Each test case will be given by two strings in a single line. The first string is the sequence of transcribed parentheses, and the second string denotes the priority order of parentheses types. Each character of the first string(sequence of parentheses) will be one of the eight symbols: '`(`', '`)`', '`{`', '`}`', '`[`', '`]`', '`<`', '`>`'. And for the priority order, left parenthesis symbols will be given in the order of highest to lowest.

Strings will not contain any whitespaces, and two strings will be separated by a single space. The length of the first string will not exceed 100, and the length of the second string will be always 4.

Note that you started with a perfectly matched sequence and that no left parentheses are wrongly transcribed to right parentheses, and vice versa. For example, '`(}`', '`()([]`', and '`([}{])`' are valid mistranscribed sequences, but '`(}}`', '`()(([]`', '`([]{])`' are not. You can assume that input will consist only of valid sequences.

Output

For each of the test case, print the resulting sequence in a single line.

Sample input and output

Standard Input	Standard Output
2 (} {(<[()([])> <({[	{} ()<()>

Notes

For the second test case, there are two mismatched pairs: '(>' and '[)'. The priority order of the parentheses, from highest to lowest, is '<>', '()', '{}', and '[]'. The first mismatched pair '(>' should be modified to '<>', as '<>' has higher priority than '()'. And the second mismatched pair '[)' should be modified to '()', as '()' has higher priority than '[]'. Thus the result should be `()<()>`.

Problem J. 에어컨을 끈다고 전력난이 해결될까?

점점 더워지는 여름! 가뜩이나 집에서 바깥바람과 선풍기만으로 더위를 이겨내려고 하는 대학원생 LIBe에게 근무 시간 내내 에어컨을 틀 수 있었던 연구실은 지상낙원이었다.

어느 날 갑자기 연구실에 근로학생이 찾아와 청천벽력 같은 소식을 전했다. 교내 전체 전력 사용량 감축을 위해 학교 모든 연구실에서 특정시간 동안 에어컨 가동을 중단하기로 했다는 이야기였다.

이야기를 듣자마자 시설팀에 전화를 걸어 항의를 해보았지만, 시설팀에서는 전력 사용량을 정부가 제시한 목표량 이상으로 감축하지 못할 경우 많은 벌금이 부과되기 때문에 어쩔 수 없다는 말만 되풀이할 뿐이었다. 더위를 못 이겨 몰래 에어컨을 켜면 금세 근로학생이 나타나 에어컨을 끄는 것을 부탁하느라 이마저도 포기했다.

이러한 날이 계속되던 와중에, 마침 시설팀에서 감축 정책 개시 이후 첫날의 건물별 시간당 전력 사용량을 알리는 공지 메일을 보냈고, 이를 본 LIBe는 과연 학교에서 정한 감축 정책이 정말 실효를 거두고 있는지를 확인해보려 한다. 메일에는 건물별 목표 전력 사용량과 9시부터 18시까지의 시간대별 전력 사용량이 적혀있다. 이를 토대로 건물별로 목표하는 전력 사용량을 지켰는지를 확인하는 프로그램을 작성하라.

Input

입력의 첫째 줄에는 교내 건물의 수를 뜻하는 숫자 T가 입력된다.

그다음 줄부터 총 T개의 건물의 전력 사용량에 대한 정보가 입력되는데, 이는 두 줄로 이뤄진다. 첫째 줄은 해당 건물의 목표 전력 사용량인 W가 입력되고, 둘째 줄에는 총 9개의 숫자 A_9, A_{10}, $\cdots$ A_{17}가 입력되는데, A_i는 i시부터 $i+1$시까지의 전력 사용량을 뜻한다. 전력 사용량에서 입력되는 숫자는 모두 0 이상 1,000 이하의 정수이며, 숫자는 공백으로 구분된다.

Output

각 건물에 대해 입력된 순서대로 전체 전력 사용량의 합이 목표 사용량 이하일 경우 "YES"를, 그렇지 못할 경우에는 "NO"를 한 줄에 출력한다.

Sample input and output

Standard Input	Standard Output
3 90 10 10 10 10 10 10 10 10 10 1000 77 77 70 11 34 35 41 83 54 50 10 20 30 40 50 60 50 40 30	YES YES NO

Problem K. 영국 아일랜드 여행

알고스팟 대회의 단골 출제위원 태윤이는 지금 영국과 아일랜드를 여행하고 있다. 태윤이는 여행을 떠나기에 앞서, 가고 싶은 여행지 N개를 골랐다. 그리고 각 여행지에 1부터 N까지 번호를 붙였고, 인접한 여행지 사이에 거리를 기록해 여행지도를 만들었다. (역시 프덕후!)

태윤이의 이번 여행은 런던(1번 여행지)에서 시작해 더블린(N번 여행지)에서 끝난다. 그리고 그사이에 가게 될 여행지는 아래와 같은 방법으로 결정한다.

태윤이는 현재 위치를 떠날 때마다, 인접한 모든 여행지를 확인한다. 현재 위치부터 더블린까지의 최단거리보다 인접한 여행지부터 더블린까지의 최단거리가 더 짧다면, 그곳은 이동 가능한 여행지다.

이동 가능한 여행지를 뽑은 후에는, 그중에 현재 위치에서 가장 가까운 여행지부터 순위를 매긴다. 거리가 같다면 번호가 작은 여행지가 더 높은 순위를 가진다. 이동 가능한 여행지 P개가 있다면, 1위는 50%, 2위는 25%, 3위는 12.5% ... P위는 $100/2^P\%$ 확률로 다음 여행지를 결정하고 이동한다. 이동한 후에는 다시 또 그다음 여행지를 고민하기 시작한다……. 아! 그리고 나머지 $100/2^P\%$ 확률로는 아이스크림을 하나 사 먹고, 더 이상 고민 없이 현재 위치에서 최단경로로 더블린까지 쭉 간다. 거리가 같은 최단경로가 여러 개 있다면, 매 순간 그중 가장 번호가 작은 여행지로 이동한다.

입력으로 여행 지도와 Q개의 여행지 번호가 주어진다. Q개의 여행지에 대해, 태윤이가 여행 도중 지나갈 확률을 구하라.

Input

입력은 T 개의 테스트 케이스로 구성된다. 입력의 첫 줄에는 T가 주어진다.

각 테스트 케이스의 첫 줄에는 여행지의 개수 N과 인접한 여행지들을 잇는 길의 개수 M이 주어진다. ($2 \leq N \leq 10000, 1 \leq M \leq 30000$) 그 후 M줄에 각각 길의 정보로써 인접한 두 여행지의 번호 V_i, U_i와 길의 거리를 나타내는 정수 D_i가 주어진다. ($1 \leq V_i < U_i \leq N, 1 \leq D_i \leq 100$) 인접한 두 여행지는 양쪽에서 서로 이동이 가능하다. 즉, V_i에서 U_i로 이동할 수도 있고, U_i에서 V_i로도 가능하다. 또한, 두 여행지 사이에는 길이 두 개 이상 존재하지 않는다. 그다음 줄에는 확률을 계산해야 할 여행지의 개수 Q가 주어지고, 그다음 줄에는 Q개의 여행지 번호 B_i가 주어진다. ($1 \leq Q \leq 100, 1 \leq B_i \leq N$)

런던에서 더블린으로 가는 길은 항상 하나 이상 존재한다.

Output

각 테스트 케이스마다 한 줄에 Q개씩, 각 도시에 대해 태윤이가 여행 도중 지나갈 확률을 공백으로 구분하여 출력한다. 10^{-8} 이내의 절대/상대 오차는 정답으로 인정된다. 그러므로 소수점 이하 8자리 이상 출력하기를 권장한다.

Sample input and output

Standard Input	Standard Output
2 3 3 1 3 7 1 2 3 2 3 2 3 1 2 3 5 9 1 2 2 1 3 3 1 4 1 1 5 8 3 4 5 2 3 1 2 5 4 3 5 3 4 5 9 5 1 2 3 4 5	1 0.75 1 1 0.625 0.75 0 1