The 33rd Annual

ACM International Collegiate Programming Contest ASIA Regional - Seoul

Problem A
Heptathlon

Ms. Lee is a heptathlon athlete participating in the Olympic Games for Korea. A heptathlon is a track and field athletics competition made up of 7 events (100m hurdles, high jump, shot put, 200m, long jump, javelin throw, and 800m). Given Lee's performances in the 7 events, we want to compute Lee's score.

The total score in a heptathlon is computed by summing the scores in all 7 events. The formula for the score of each event follows the following format:

- Score = $\lfloor A \times (B - P)^c \rfloor$ for Running Events
- Score = $\lfloor A \times (P - B)^c \rfloor$ for Field Events

where A, B, and C are constants given below and P is a player's performance in the units described below. Note that the score for each event is an integer.

Event	A	B	C	P	Type
100 m Hurdles	9.23076	26.7	1.835	sec	Running
High Jump	1.84523	75	1.348	cm	Field
Shot Put	56.0211	1.5	1.05	m	Field
200 m	4.99087	42.5	1.81	sec	Running
Long Jump	0.188807	210	1.41	cm	Field
Javelin Throw	15.9803	3.8	1.04	m	Field
800 m	0.11193	254	1.88	sec	Running

We assume that $B \leq P$ for field events and $P \leq B$ for running events.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T ($1 \leq T \leq 1000$) is given in the first line of the input. Each test case consists of one line with 7 integers. These integers represent Lee's performances in the units described in column P of the table above. The order of events is 100m hurdles, high jump, shot put, 200m, long jump, javelin throw, and 800m.

Output

Your program is to write to standard output. Print Lee's total score.

The following shows sample input and output for three test cases.

Sample Input	Output for the Sample Input
19 90 11 29 264 20 131 12 95 21 37 224 35 221 17 168 15 23 275 22 241	2901 3419 3772

The 33rd Annual

ACM International Collegiate Programming Contest ASIA Regional - Seoul

Problem B
MCS

In modern molecular biology the genome of an organism is its hereditary information encoded in DNA. The genome includes both the genes and the non-coding sequences of the DNA. In the view of computer science, the genome can be regarded as a very long string consisting of only four letters {A, G, T, C}.

In order to study specific genetic diseases, it is very important to examine if there are DNA (substring) patterns which appear more frequently than others in a whole genome. Especially we are intended in finding "*compositionally equivalent*" substrings. A string P is said to be compositionally equivalent to a string Q if the number of 4 letters {A, G, T, C} appearing in P and Q is exactly the same. For example, P="ATTATGC" is compositionally equivalent to Q="GTATCTA" since the number of 'A' , 'G' , 'C' and 'T' in P is exactly same to that in Q, respectively. In the other hand, "TTGCA" is not compositionally equivalent to "TGCCA".

In this problem we want to find the *k*-Major Composition Substring (*k*-MCS for short). For a genome string given, *k*-MCS is defined as the most frequently appearing substring of length *k* up to compositionally equivalence. Since the *k*-MCS for a given input genome is not necessarily unique, two or more different *k*-MCS could be possible. In the following, *k*-substring means a substring of length *k*.

Let us show one example. We have a genome string W="GCAGGAGCGCCAGG" with length 14. There are many different compositionally equivalent 3-substrings such as GCA, CAG, GGA, GAG,… CAG and AGG. In W it is easy to find that "AGG"is a *3*-MCS which appears four times as AGG, GGA, GAG and AGG, since there is no other *3*-substring (up to compositionally equivalence) which appears more than 4 times in W.

Input

Your program is to read the input from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case (input genome) starts with the value *k* for *k*-MCS and the genome string W, where $\mathbf{1} \le k \le \mathbf{600}$. The length of the input genome string, |W|, is bounded by $\mathbf{10} \le |\mathbf{W}| \le \mathbf{60000}$.

Output

Your program is to write the number of occurrence of a *k*-MCS appearing in each genome string.

Sample Input	Output for the Sample Input
3 3 GCAGGAGCGCCAGG 4 AGTCCTTAGAG 5 GGGAGGGGGGGTGGGGGGGGT	4 2 7

The 33rd Annual

ACM International Collegiate Programming Contest ASIA Regional - Seoul

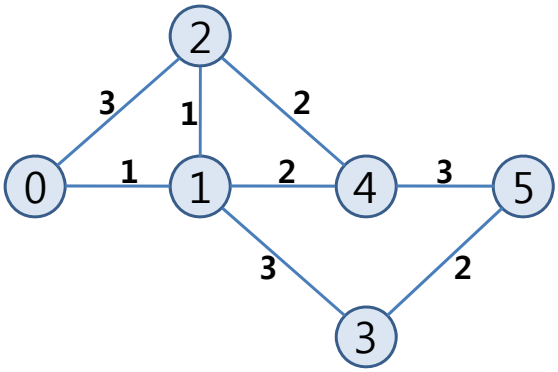
Problem C
Homing

Mr. Kim visits his hometown annually. He always drives his car following the shortest path to his hometown. Since he is a very economical person, he only fills his fuel tank with the amount of gasoline needed to follow the shortest path for fuel efficiency. Last year, he fueled the exact amount to travel the shortest path, but he was in big trouble because an unexpected accident occurred on the shortest path. That is, he was short of gasoline because he had to find a detour to get to his hometown.

This year, he wants to fuel enough gasoline considering unexpected detours caused by an accident. According to statistical reports, at most one accident occurs in a day. (His hometown can be reached if he drives all day long.) Thus he wants to fill his fuel tank with the smallest amount of gasoline supposing that there will be only one accident. For this, he will find the shortest path to get to his hometown wherever the accident occurs.

The road system can be represented as a weighted graph $G = (V, E)$, where V is the set of vertices that represent cities, E is the set of edges that represent roads connecting two cities, and $w(e)$ is the weight of each edge e that represents the amount of gasoline needed to drive the corresponding road. An accident always occurs in the middle of a road and the occurrence of the accident can be found when he reaches the cities which are incident with the road. This means that no accident can be reported in advance.

For example, consider the case when the city of departure is node 0 and the city of arrival is node 5 in the given graph below. Then the shortest path is $P = \langle 0,1,4,5 \rangle$ and the units of gasoline, i.e., the sum of weights on the shortest path, is $W(P) = 6$. Suppose that an accident occurs in the edge (0, 1) between the two cities 0 and 1. Then, the shortest detour at node 0 which does not make use of edge (0, 1) becomes $\langle 0,2,4,5 \rangle$ and it requires 2 more units of gasoline than path P. When an accident occurs at edge (1, 4) (resp. edge (4, 5)) the shortest detour will be $\langle 1,3,5 \rangle$ (resp. $\langle 4,1,3,5 \rangle$) and it requires 0 (resp. 4) more units of gasoline than the path P. So, Mr. Kim should fill his fuel tank with at least 10 units of gasoline since he needs 4 more units of gasoline than that of $W(P)$, i.e., 4 more units of gasoline is the largest amount considering the detour by an unexpected accident for every node on the shortest path P.



When we are given the road system represented as a weighted graph and the shortest path from the city of departure to the city of arrival followed by Mr. Kim, find the smallest amount of gasoline that Mr. Kim has to fuel considering a detour by an accident.

Input

Your program is to read from standard input. The input consists of T test cases and the number of test cases T ($1 \le T \le 20$) is given in the first line of the input. Each test case starts with a line containing two integers n and m which represent the number of cities and the number of roads with ranges of $3 \le n$, $m \le 10{,}000$, respectively. The cities are numbered from 0 to n-1. In the next m lines are given with three integers c, d and w (separated by a space), where w represents the amount of gasoline needed to travel between the cities c and d ($c \ne d$). In the next line, $k+1$ integers are given representing the shortest path that Mr. Kim has chosen. The first integer is k as the total number of cities on the shortest path and the next k integers represent the cities along the path. That is, the second integer represents the city of departure and the last integer represents the city of arrival.

Output

Your program is to write to standard output. Print the amount of gasoline with which Mr. Kim should feed his fuel tank. If there exists no way to go to his hometown if an accident occurs, print -1.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 6 8 0 1 1 0 2 3 2 1 1 4 2 2 1 4 2 5 4 3 3 1 3 5 3 2 4 0 1 4 5 4 3 0 1 2 2 1 4 1 3 3 3 0 1 3	10 -1

The 33rd Annual

ACM International Collegiate Programming Contest ASIA Regional - Seoul

Problem D
Cycle

A small computer of Von Neumann architecture named ICPC is used for the BDN Programming Contest. ICPC is a 16-bit integer machine. There is a sufficient number of instruction memory cells in ICPC but it has only one data memory cell. ICPC has 6 registers: R1, R2, R3, R4, R5, and PC. The R*n* registers are general purpose but PC is the program counter storing the address of the next instruction to be executed. The PC value can be changed only by a group of control-flow instructions. This machine follows the usual fetch-decode-execute cycles. The PC value is normally increased automatically except for the cases of the control flow instructions. ICPC has only two addressing modes, immediate value and register, but PC cannot be used as an operand. The whole set of instructions of ICPC is shown in Table 1. In Table 1, *r* denotes a register, *v* denotes a register or an integer value, and *M* denotes the data memory cell.

Table 1: The instruction set of ICPC

ICPC Instruction	The meaning of the instruction in C
`load` *r*	*r* = *M*;
`store` *v*	*M* = *v*;
`move` *r v*	*r* = *v*;
`add` *r v*	*r* += *v*;
`sub` *r v*	*r* -= *v*;

`loop` *r*
instructions
`pool`
	`while (`*r*`> 0) {`
	execute the *instructions*
	`}`

`cond` *r*
instructions
`dnoc`
	`if (`*r*`> 0) {`
	execute the *instructions*
	`}`

Every step of the fetch-decode-execute cycle is called “cycle.” Therefore every instruction consumes three cycles at least and every ICPC instruction, except for `dnoc`, consumes exactly three cycles. The instruction `dnoc` is just used for denoting the end of `cond` and not executed at all. This kind of instruction is called a pseudo instruction; `dnoc` is the only pseudo instruction in ICPC. The instruction `pool` denotes the end of `loop`, just like `dnoc`, but it is executed indeed for the control flow should return the beginning of the `loop` at the end of a `loop`.

To minimize the execution time of programs, pipelining is usually adopted in modern computer architectures and ICPC also adopts it. Assuming that three instructions, namely A, B, and C, are to be executed in sequence, the decode cycle of A can overlap the fetch cycle of B and the execute cycle of A can overlap the decode cycle of B and also can overlap the fetch cycle of C. Therefore, it takes only 4 cycles for one `move` and one `add` instruction in sequence rather than 6 cycles, as shown in the first two instructions of Fig. 1(a). In Fig. 1, we used F for fetch, D for decode, and E for execute cycle.

Pipelining is stalled if the PC encounters a control flow instruction because we do not know the next instruction to be executed. The next instruction can be determined only after the control flow instruction is executed. The `cond` instruction in Fig. 1(a) shows this fact. Note that `dnoc` is not executed at all since it is a pseudo instruction and it takes 9 cycles to execute all the instructions in Fig. 1(a).

However the instruction `pool`, similar to `dnoc`, is really a control flow instruction. For example, in Figure 1(b), not only the instruction `loop` but also the instruction `pool` stalls the pipelining.

Programs		Cycles										
		1	2	3	4	5	6	7	8	9	10	11
(a)	`move R1 0`	F	D	E								
	`add R1 1`		F	D	E							
	`cond R1`			F	D	E						
	`add R1 2`						F	D	E			
	`dnoc`											
	`add R1 5`							F	D	E		
(b)	`move R1 1`	F	D	E								
	`loop R1`		F	D	E					F	D	E
	`sub R1 1`					F	D	E				
	`pool`						F	D	E			

Figure 1: Example programs and the corresponding cycle counts

Input

Your program is to read the input from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains the number of lines L of ICPC instructions ($L > 0$) and the remaining L lines contain the sequence of ICPC instructions of the test case. Every instruction line contains exactly one ICPC instruction. The input may be indented according to the nesting of control structures. The `loop` and `cond` should contain at least one instruction, i.e. there is no empty loop or empty branch. The maximum number of characters in an input line is 100. Immediate values are 16-bit two's complement integers, i.e. an immediate value N is in $-32768 \le N \le +32767$. The op code and the operands are separated with at least one blank character. There is no *infinite loop* in the test cases.

Output

Your program is to write to standard output. Your program should count the number of cycles when executing the ICPC program given in standard input. The initial contents of the data memory cell and the registers are assumed to be 0. When overflow or underflow occurs in any of the registers or the data memory cell, your program should write "error" rather than a cycle count.

The following shows sample input and output for three test cases.

Sample Input

```
3
7
move R1 0
move R2 10
loop R2
  add R1 R2
  sub R2 1
pool
store R1
4
move R1 1
loop R1
  add R1 1
pool
5
move R1 1
cond R1
  add R1 2
dnoc
add R1 5
```

Output for the Sample Input

```
88
error
8
```

The 33rd Annual

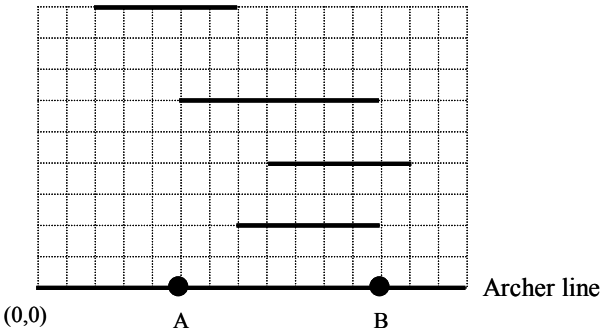
ACM International Collegiate Programming Contest ASIA Regional - Seoul

Problem E
Archery

Korea's reputation in archery is well known because the Korean archery teams have been sweeping almost all gold, silver, and bronze medals in the Olympic Games.

An archery game ICPC supported by NEXON (one of Korea's leading publishers of online contents) will be held in Korea. As a ceremonial event of the game, a famous master of archery will shoot an arrow to hit through all target boards made of paper. Because an arrow flies along a straight line, it depends on his position of the archer line whether or not he hits all targets.

The figure below shows an example of the complete view of a game field from the sky. Every target is represented by a line segment parallel to the archer line. Imagine the coordinate system of which the origin is the leftmost point of the archer line and the archer line is located on the positive x-axis.



In the above figure, the master can hit all targets in position B. However, he never hits all targets in position A because any ray from A intersects at most 3 targets.

Given the width of the archer line and the target locations, write a program for determining if there exists a position at which the master can hit all targets. You may assume that the y-coordinates of all targets are different. Note that if an arrow passes through an end point of a target, it is considered to hit that target.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T ($1 \le T \le 30$) is given in the first line of the input. Each test case starts with a line containing an integer W ($2 \le W \le 10{,}000{,}000$), the width of an archer line. The next line contains an integer N ($2 \le N \le 5{,}000$), the number of target boards. The i-th line of the following N lines contains three integers D_i, L_i, R_i ($1 \le D_i \le W$, $0 \le L_i < R_i \le W$), where $1 \le i \le N$, D_i represents the y-coordinate of the i-th target, and L_i and R_i represent the x-coordinates of the leftmost point and the rightmost point of the target, respectively. Note that $D_i \neq D_j$ if $i \neq j$.

Output

Your program is to write to standard output**.** Print exactly one line for each test case. Print "YES" if there exists a position on the archer line at which a master of archery can hit all targets, otherwise, "NO".

The following shows sample input and output for three test cases.

Sample Input	Output for the Sample Input
3 15 4 10 2 7 7 5 12 2 7 12 4 9 13 6 3 2 1 3 4 0 2 5 4 6 10 4 8 2 5 4 2 5 6 5 8 2 5 8	YES NO YES

The 33rd Annual

ACM International Collegiate Programming Contest ASIA Regional - Seoul

Problem F
Processor

An "early adopter" Mr. Kim bought one of the latest notebooks which has a speed-controlled processor. The processor is able to operate at variable speed. But the higher the speed, the higher the power consumption is. So, to execute a set of programs, adjusting the speed of the processor dynamically results in energy-efficient schedules. We are concerned in a schedule to minimize the maximum speed of the processor.

The processor shall execute a set of programs and each program P_i is given having a starting time r_i, a deadline d_i, and work w_i. When the processor executes the programs, for each program P_i, the work w_i should be done on the processor within the interval $[r_i, d_i]$ to complete P_i. Also, the processor does not have to execute a program in a contiguous interval, that is, it can interrupt the currently running program and later resume it at the interrupted point. It is assumed that r_i, d_i, and w_i are given positive integers. Recall that the processor can execute the programs at variable speed. If the processor runs the program P_i with work w_i at a constant speed s, then it takes $\frac{w_i}{s}$ time to complete P_i. We also assume that the available speeds are positive integers, that is, the processor operates only at integer points of speed. The speed is unbounded and the processor may operate at sufficiently large speeds to complete all the programs. The processor should complete all the given programs and the goal is to find a schedule minimizing the maximum of the speeds at which the processor operates.

For example, there are five programs P_i with the interval $[r_i, d_i]$ and work w_i, $i = 1,\ldots,5$, where $[r_1, d_1]$=[1, 4], $[r_2, d_2]$=[3, 6], $[r_3, d_3]$=[4, 5], $[r_4, d_4]$=[4, 7], $[r_5, d_5]$=[5, 8] and w_1=2, w_2=3, w_3=2, w_4=2, w_5=1. Then the Figure 1 represents a schedule which minimizes the maximum speed at which the processor operates. The maximum speed is 2 in this example.

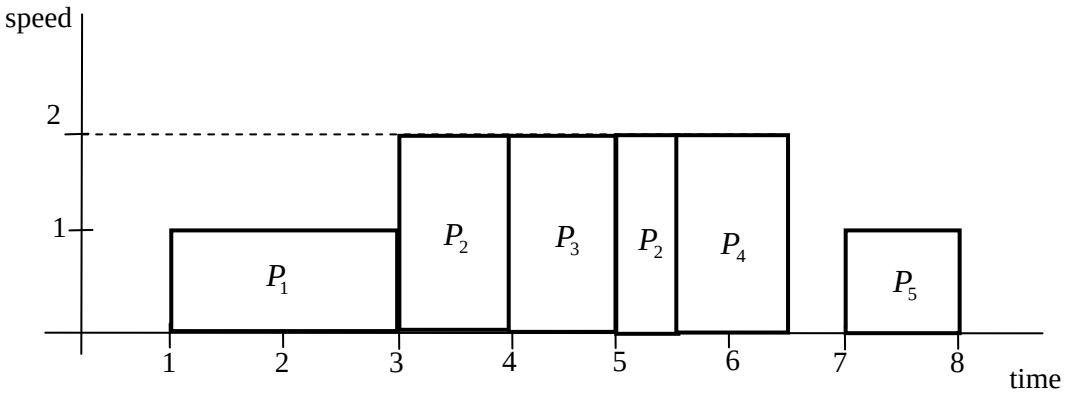


Figure 1

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases $T(1 \le T \le 20)$ is given on the first line of the input. The first line of each test case contains an integer $n(1 \le n \le 10{,}000)$, the number of given programs which the processor shall execute. In the next n lines of

each test case, the i-th line contain three integer numbers r_i, d_i, and w_i, representing the starting time, the deadline, and the work of the program P_i, respectively, where $1 \le r_i < d_i \le 20{,}000$, $1 \le w_i \le 1{,}000$.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line contains the maximum speed of a schedule minimizing the maximum speed at which the processor operates to complete all the given programs.

The following shows sample input and ouput for three test cases.

Sample Input	Output for the Sample Input
3 5 1 4 2 3 6 3 4 5 2 4 7 2 5 8 1 6 1 7 25 4 8 10 7 10 5 8 11 5 10 13 10 11 13 5 8 15 18 10 20 24 16 8 15 33 11 14 14 1 6 16 16 19 12 3 5 12 22 25 10	2 5 7

The 33rd Annual

ACM International Collegiate Programming Contest ASIA Regional - Seoul

Problem G

Guess

Given a sequence of integers, $a_1, a_2, \ldots, a_n$, we define its *sign matrix* S such that, for $1 \le i \le j \le n$, S_{ij}="+" if $a_i + \ldots + a_j > 0$; S_{ij}="−" if $a_i + \ldots + a_j < 0$; and S_{ij}="0" otherwise.
For example, if $(a_1,a_2,a_3,a_4)=(-1, 5, -4, 2)$, then its sign matrix S is a 4×4 matrix:

	1	2	3	4
1	−	+	0	+
2		+	+	+
3			−	−
4				+

We say that the sequence (−1, 5, −4, 2) *generates* the sign matrix. A sign matrix is *valid* if it can be generated by a sequence of integers.

Given a sequence of integers, it is easy to compute its sign matrix. This problem is about the opposite direction: Given a valid sign matrix, find a sequence of integers that generates the sign matrix. *Note that two or more different sequences of integers can generate the same sign matrix.* For example, the sequence (−2, 5, −3, 1) generates the same sign matrix as the sequence (−1,5, −4,2).

Write a program that, given a *valid* sign matrix, can find a sequence of integers that generates the sign matrix. You may assume that every integer in a sequence is between −10 and 10, both inclusive.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case consists of two lines. The first line contains an integer n ($1 \le n \le 10$), where n is the length of a sequence of integers. The second line contains a string of $n(n+1)/2$ characters such that the first n characters correspond to the first row of the sign matrix, the next $n-1$ characters to the second row, ..., and the last character to the n-th row.

Output

Your program is to write to standard output. For each test case, output exactly one line containing a sequence of n integers which generates the sign matrix. If more than one sequence generates the sign matrix, you may output any one of them. Every integer in the sequence must be between −10 and 10, both inclusive.
The following shows a sample input with three test cases and its output.

Sample Input	Output for the Sample Input
3 4 -+0++++--+ 2 +++ 5 ++0+-+-+--+-+--	-2 5 -3 1 3 4 1 2 -3 4 -5

The 33rd Annual

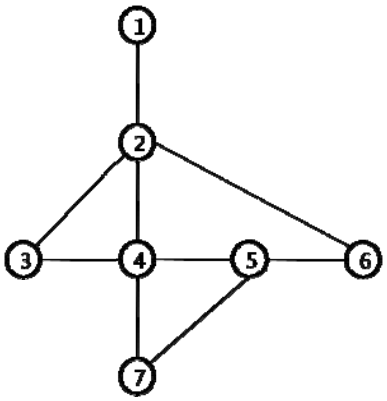
ACM International Collegiate Programming Contest ASIA Regional - Seoul

Problem H

Salesmen

Traveling salesmen of nhn (the prestigious Korean internet company) report their current location to the company on a regular basis. They also have to report their new location to the company if they are moving to another location. The company keeps each salesman's working path on a map of his working area and uses this path information for the planning of the next work of the salesman. The map of a salesman's working area is represented as a connected and undirected graph, where vertices represent the possible locations of the salesman and edges correspond to the possible movements between locations. Therefore the salesman's working path can be denoted by a sequence of vertices in the graph. Since each salesman reports his position regularly and he can stay at some place for a very long time, the same vertices of the graph can appear consecutively in his working path. Let a salesman's working path be *correct* if two consecutive vertices correspond to either the same vertex or to two adjacent vertices in the graph.

For example, on the following graph representing the working area of a salesman,



a reported working path [1 2 2 6 5 5 5 7 4] is a correct path. But a reported working path [1 2 2 7 5 5 5 7 4] is not a correct path since there is no edge in the graph between vertices 2 and 7. If we assume that the salesman reports his location every time when he has to report his location (but possibly incorrectly), then the *correct* path could be [1 2 2 ***4*** 5 5 5 7 4], [1 2 ***4*** 7 5 5 5 7 4], or [1 2 2 ***6*** 5 5 5 7 4].

The length of a working path is the number of vertices in the path. We define the *distance* between two paths $A=a_1a_2\ldots a_n$ and $B=b_1b_2\ldots b_n$ of the same length n as

$$dist(A,B) = \sum_{i=1}^{n} d(a_i, b_i)$$

where

$$d(a,b) = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{otherwise} \end{cases}.$$

Given a graph representing the working area of a salesman and a working path (possible not a correct path), *A*, of a salesman, write a program to compute a correct working path, *B*, of the same length where the distance *dist*(*A*, *B*) is minimized.

Input

The program is to read the input from standard input. The input consists of *T* test cases. The number of test cases (*T*) is given in the first line of the input. The first line of each test case contains two integers n_1, n_2 ($3 \le n_1 \le 100, 2 \le n_2 \le 4{,}950$) where n_1 is the number of vertices of the graph representing the working map of a salesman and n_2 is the number of edges in the graph. The input graph is a connected graph. Each vertex of the graph is numbered from 1 to n_1. In the following n_2 lines, each line contains a pair of vertices which represent an edge of the graph. The last line of each test case contains information on a working path of the salesman. The first integer n ($2 \le n \le 200$) in the line is the length of the path and the following n integers represent the sequence of vertices in the working path.

Output

Your program is to write to standard output. Print one line for each test case. The line should contain the minimum distance of the input path to a correct path of the same length.

The following sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 7 9 1 2 2 3 2 4 2 6 3 4 4 5 5 6 7 4 7 5 9 1 2 2 7 5 5 5 7 4 7 9 1 2 2 3 2 4 2 6 3 4 4 5 5 6 7 4 7 5 9 1 2 2 6 5 5 5 7 4	1 0

The 33rd Annual

ACM International Collegiate Programming Contest ASIA Regional - Seoul

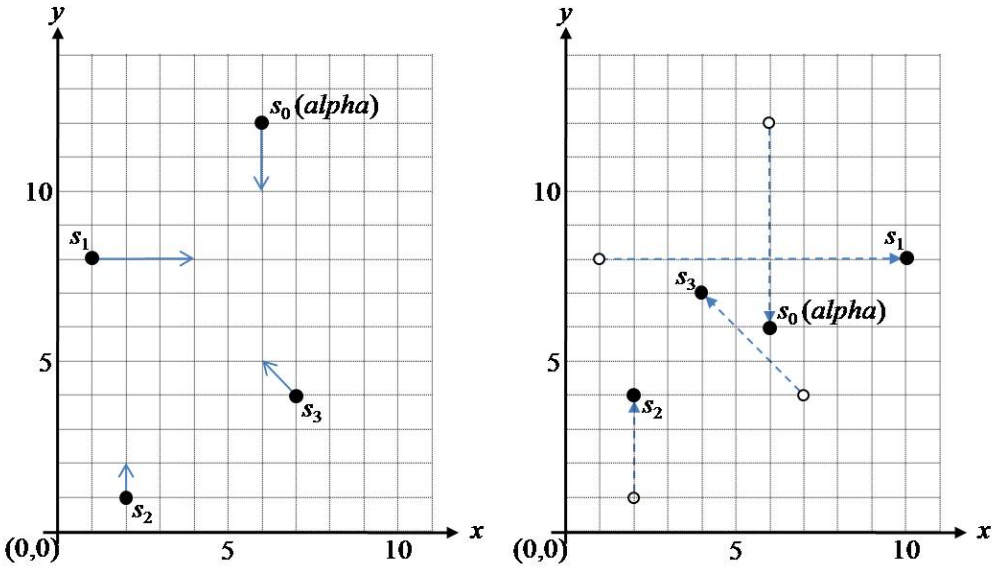
Problem I

Hubble Space Telescope

An astrophysicist makes observations of a spiral galaxy known as the Andromeda Galaxy using the Hubble space telescope. In particular, he is interested in the motion of n stars in the galaxy. Each star moves with a constant velocity along a straight line in the picture of the Hubble space telescope. Among the n stars there is a special star named *alpha*. He wants to know when the maximum distance from *alpha* to the remaining $n-1$ stars is minimized.

The picture screen of the Hubble space telescope can be represented by a 2-D Cartesian coordinate system and let $S = \{s_0, s_1, \ldots, s_{n-1}\}$ be a set of n stars in the plane. Assume that the s_0 is the star *alpha*. A star s_i of S moves along the trajectory $p_i + tv_i$ over time t, where $p_i = (x_i, y_i)$ is the initial position of s_i in the 2-D coordinate system and $v_i = (a_i, b_i)$ is the velocity vector of s_i. We assume that there is no actual collision of two stars, i.e. the two stars pass through each other when they meet a point in 2-D space. Given a set of stars and their velocity, compute the time when the maximum distance from *alpha* to the remaining stars is minimized within 10^5 time units. If there is more than one such time, then output the earliest time.

Following figure 1 shows an example with 4 stars. The arrow of a star indicates its velocity vector. In this example, when the time is $t = 3$, the maximum distance from *alpha* to the remaining 3 stars is minimized.



(a) at time $t = 0$ (b) at time $t = 3$

Figure 1. An Example.

Input

Your program is to read the input from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing an integer n ($2 \le n \le 50{,}000$), the number of stars of S. Each of the next n lines contains four integers x_i, y_i and a_i b_i; (x_i, y_i) is the position of a star s_i at time zero and (a, b_i) is the velocity vector of the star s_i ($-200{,}000 \le x_i, y_i \le 200{,}000, -500 \le a_i, b_i \le 500$). Two or more stars of S may have the same coordinates at time zero.

Output

Your program is to write to standard output. Print the time when the maximum distance is minimized from s_0 (*alpha*) to the remaining $n-1$ stars $\{s_1, s_2, \cdots, s_{n-1}\}$ within 10^5 time units, rounded to 4 fractional digits.

The following shows sample input and output for five test cases.

Sample Input	Output for the Sample Input
5 4 6 12 0 -2 1 8 3 0 2 1 0 1 7 4 -1 1 2 1000 1000 -1 0 -1000 -1000 1 0 5 0 0 0 0 -20000 0 10 0 19990 0 -10 0 0 19900 0 -10 0 -19999 0 10 8 -621 -213 3 1 -875 782 1 -4 584 700 -5 -2 -12 -628 3 2 -771 -460 1 3 676 57 -1 -2 420 -864 -2 4 190 -950 -4 5 2 -200000 0 2 0 200000 0 -2 0	3.0000 1000.0000 1995.0000 196.4510 100000.0000

The 33rd Annual

ACM International Collegiate Programming Contest ASIA Regional - Seoul

Problem J

Metal

You, a promising metal-artist, try to make a metalwork, a piece of steel. You first mark n points on a big steel board, and cut a polygon connecting those n points from the board. For this cutting, you melt the board along the boundary of the polygon with two lasers hanging down from a long bar over the board as shown in Figure 1. The bar is vertical, i.e., parallel to the *y*-axis, and moves continuously (without stopping) only in the positive *x*-direction, i.e., from the left side to the right side of the board. The lasers can move only along the bar and can never meet on the bar. Furthermore, the bar moves monotonously from left to right, it can never move to a position left of its current position. These conditions imply that the polygons you can obtain must be *simple* and *monotone*. A polygon P is *simple* if any two edges of P do not intersect except for the endpoints of adjacent edges, and there are no holes inside P. A polygon P is *monotone* if any intersection of P with a vertical line is either a point or a line segment. To choose a proper shape for your metalwork, you want to know how many different simple and monotone polygons of the n points exist.

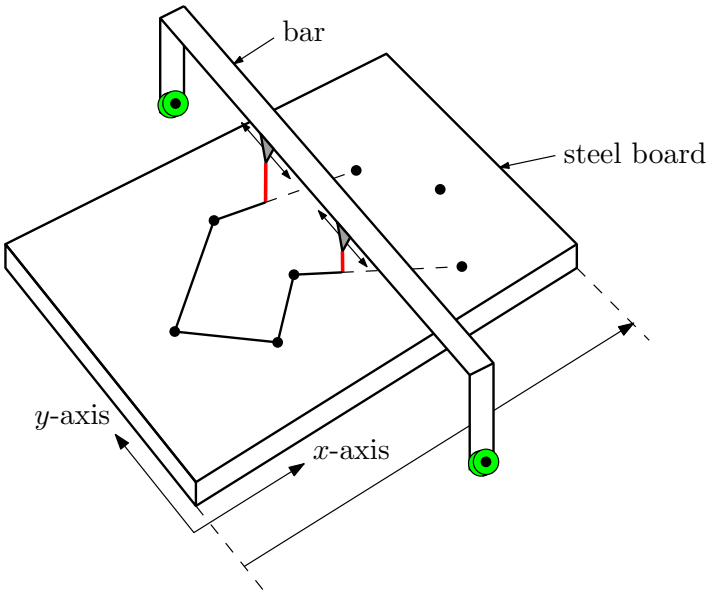


Figure 1. A cutting tool with two lasers hanging from a vertical bar.

For example, see Figure 2, where seven points are given as an input. There are four different simple and monotone polygons for this input. Your task is to compute the number of different simple and monotone polygons of n given points.

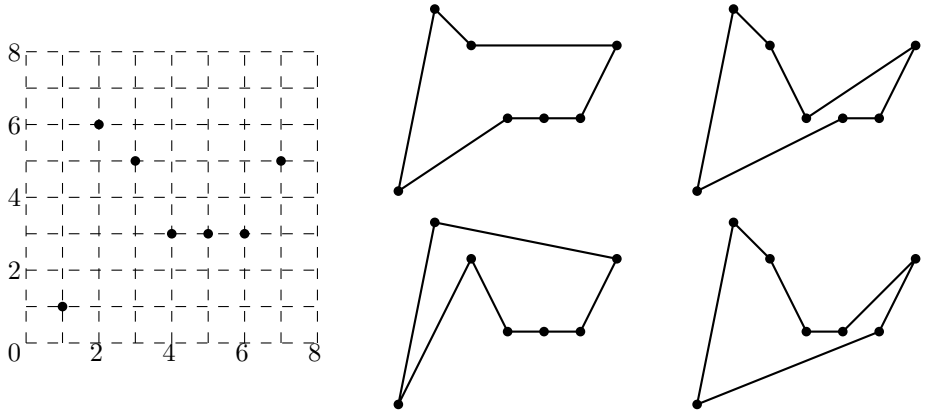


Figure 2. There are four different simple and monotone polygons for seven input points.

Input

Your program is to read the input from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing an integer n ($3 \le n \le 50$), the number of points of $S = \{s_0, s_1, \ldots, s_{n-1}\}$. Each of the next n lines contains two non-negative integers x_i and y_i ($0 \le x_i, y_i \le 200{,}000$), where point s_i has coordinates (x_i, y_i). No two points of S have the same x-coordinate.

Output

Your program is to write to standard output. For each test case, print the number of different simple and monotone polygons which connect n points of S.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 4 0 0 6 3 7 1 4 1 7 7 5 2 6 1 1 3 5 4 3 6 3 5 3	2 4