Algospot.com 9주년 기념 대회

주최 및 주관

ALGOSPOT

2016년 10월 29일

우리의 친구이자 가장 훌륭한 운영진이었던 류원하에게 넘치는 감사와 존경의 뜻을 함께 전합니다.
비록 이르게 떠나갔지만 그가 세상에 남긴 빛나는 흔적은 잊혀지지 않을 것입니다.

알고스팟 운영진 일동

참가자를 위한 도움말

주의 사항

- 대회 시간은 10:00부터 15:00까지입니다.
- 모든 입력은 *표준 입력*으로 주어지며, 모든 출력은 *표준 출력*으로 합니다.
- 테스트 케이스가 존재하는 문제의 경우, 테스트 케이스에 대한 출력을 모아서 하실 필요 없이, 각 테스트 케이스를 처리할 때마다 출력해도 괜찮습니다.
- 대회 도중에 ACM-ICPC 에서 정하는 규정에 따라 인터넷 검색, bison이나 yacc와 같은 자동 코드 생성 도구 사용 등의 행위를 삼가해 주시기 바랍니다.
- *리턴 코드*에 주의하십시오. 프로세스가 0 이 아닌 리턴 코드를 되돌리는 경우 "*Runtime Error*" 를 받게 됩니다.
- 문제에 대한 질의 사항은 대회 페이지의 질문 기능을 사용해 주시기 바랍니다. 이 때 대답해 드리기 어려운 질문에 대해서는 "*No response, read problem statement carefully*" 로 대답될 수 있으므로 유의하십시오. 대답해 드리기 어려운 질문의 예는 아래와 같습니다 :
  - 테스트 케이스의 수는 몇 개나 되나요
  - 시간제한은 얼마나 되나요
  - 제 솔루션이 왜 WA인가요
  - 문제에서 이러이러한 사항을 임의로 가정해도 되나요

채점 결과에 대하여

Correct 제출하신 답안이 모든 테스트 데이터를 정해진 시간 안에 통과하여 정답으로 인정되었음을 의미합니다.

No - Wrong Answer 제출하신 답안이 특정 테스트 케이스에 대해 제한시간 안에 종료되었으나, 생성한 출력이 출제자의 정답과 일치하지 않음을 의미합니다.

No - Too-Late 대회가 끝나고 제출을 하였음을 의미합니다.

No - No Output 제출하신 답안 프로그램이 아무런 출력을 하지 않았음을 의미합니다.

No - Time-limit Exceeded 제출하신 답안 프로그램이 정해진 시간 안에 종료되지 않았음을 의미합니다.

No - Run-Error 제출하신 답안 프로그램을 실행하는 도중 프로세스가 비정상적으로 종료되었음을 의미합니다.

No - Compiler-Error 제출하신 답안 프로그램을 컴파일하는 도중 오류가 발생하였음을 의미합니다

문제의 제한 사항

A. 슈퍼판타 지워 : 1 second, 512 MB

B. 단칼빙 : 2 seconds, 512 MB

C. Coloring Madness : 1 second, 512 MB

D. Spotboard : 1 second, 512 MB

E. Master of Baduk : 1 second, 512 MB

F. 요새 2 : 3 seconds, 512 MB

G. 총기 밀수 : 1 second, 512 MB

H. 놀이공원 : 1 second, 512 MB

I. Circle Routes : 1 second, 512 MB

J. Time to Chicken : 3 seconds, 512 MB

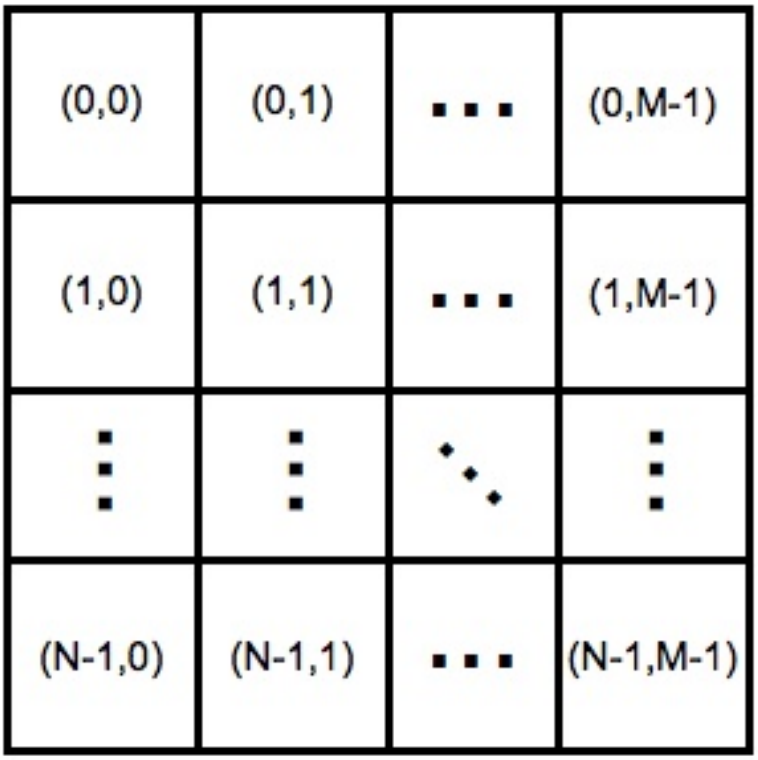
K. k-edit Subset : 2 second, 512 MB

L. PPAP : 1 second, 512 MB

Problem A. 슈퍼판타 지워

원하의 팀이 관리하는 모바일 게임 중 하나인 '슈퍼판타 지워'는 플레이어가 슈퍼 주인 아주머니가 되어 슈퍼를 노리는 진상 고객으로부터 슈퍼를 지키는 턴제 RPG 경영 게임이다. 모바일 RPG 게임 시장에서 자동 전투 모드가 없는 게임은 경쟁력이 떨어지므로, '슈퍼판타 지워'도 이번 대규모 업데이트에 자동 전투를 추가하려고 한다.

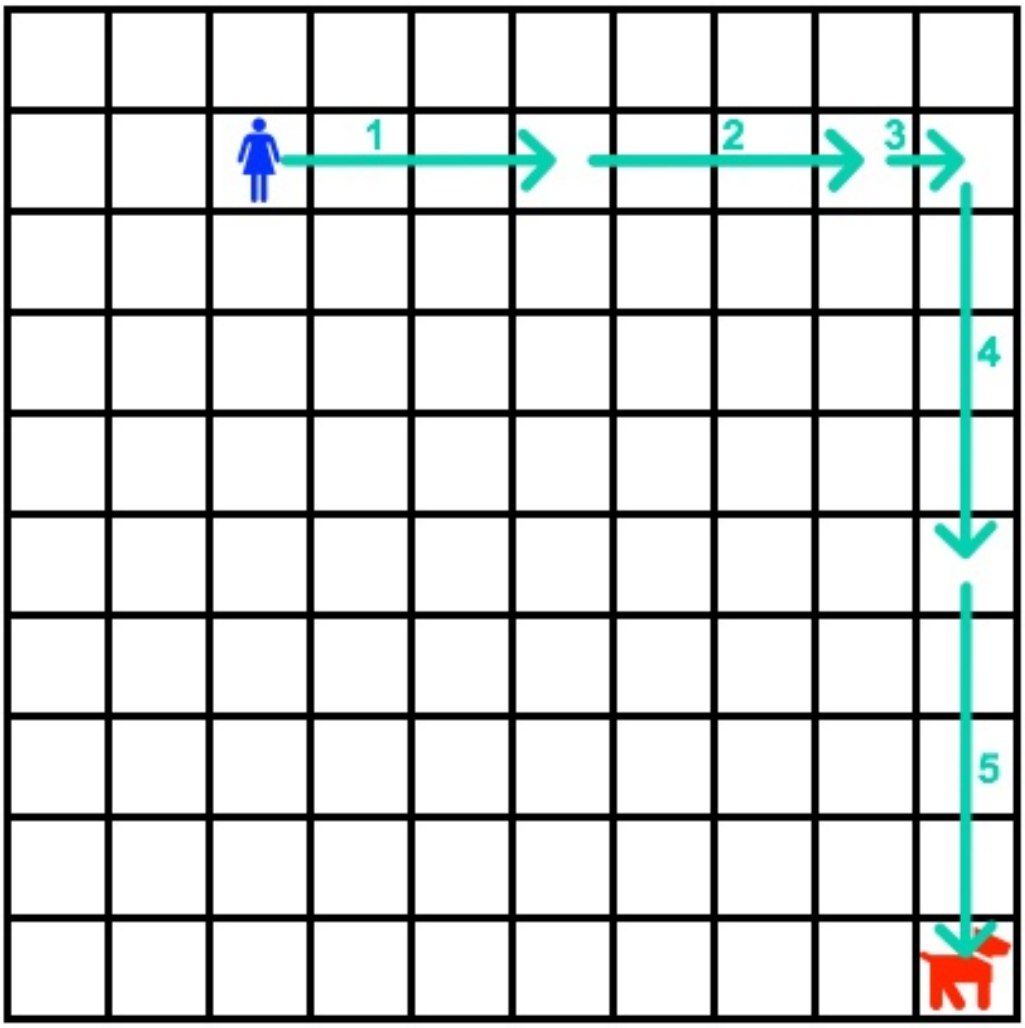
'슈퍼판타 지워'의 전투 화면은 내부적으로 아래와 같이 $(0,0)$부터 $(N-1, M-1)$의 정수 칸으로 되어 있다.



처음 슈퍼 아주머니는 $(r0, c0)$에서 시작하여, 한 턴 당 세로로 최대 A칸을 이동하거나, 가로로 최대 B칸을 이동할 수 있다. 진상 손님의 위치가 (r, c)라면, 슈퍼 아주머니가 (r, c)에 도달하면 진상 손님을 쫓아낼 수 있다. 이 때, 진상 손님은 턴이 지나도 움직이지 않는다.

자동 전투 AI를 만들기 위해 슈퍼 주인 아주머니 캐릭터가 최소 몇 턴이 지나야 진상 손님을 쫓아낼 수 있는지 계산해야 한다.

예를 들어, $A = 4$, $B = 3$이고, 슈퍼 아주머니의 처음 위치가 $(1, 2)$, 진상 손님이 $(9, 9)$에 있다고 하면 아래와 같이 이동하여 5턴만에 진상 손님을 쫓아낼 수 있다.



Input

입력은 여러개의 테스트케이스로 이뤄지며, 첫 줄에 테스트 케이스의 수 T ($1 \leq T \leq 1\,000$)가 정수로 주어진다.

테스트 케이스마다 정수 N, M ($1 \leq N, M \leq 2\,000\,000\,000$), $r0$, $c0$, r, c ($0 \leq r0, r < N$, $0 \leq c0, c < M$), A, B ($1 \leq A < 2\,000\,000\,000$, $1 \leq B < 2\,000\,000\,000$)가 각각 공백 하나로 구분되어 한 줄에 주어진다.

Output

각 테스트 케이스마다 첫 번째 줄에 정답을 출력한다.

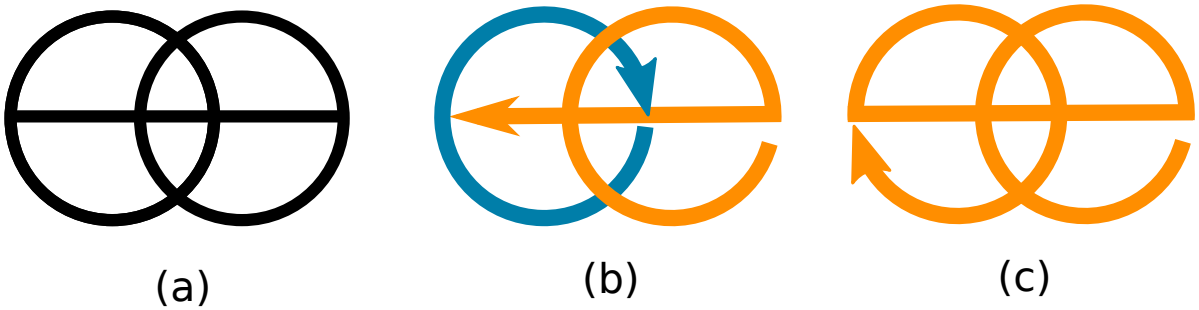
Sample input and output

standard input	standard output
3 9 9 0 1 8 8 4 3 9 9 8 8 0 1 4 3 9 9 1 1 1 1 3 4	5 5 0

Problem B. 단칼빙

단칼빙은 무엇이든 싹둑싹둑 잘라 버리는 훌륭한 포켓몬이다. 단칼빙은 그 이름답게 대부분의 것들을 한 번에 잘라 버릴 수 있지만, 가끔 한 번에 자를 수 없는 것을 만나면 혼란에 빠지게 된다. 오늘 단칼빙에게는 선분과 원들이 그려진 종이가 주어졌다. 이 선분과 원들의 합집합으로 구성된 도형을 따라 종이를 잘라야 하는데, 한 선분이나 곡선은 정확히 한 번만 자르고 지나갈 수 있다. 따라서 중간에 칼을 종이에서 떼고 다른 점에서 자르기 시작해야 할 수도 있다. 물론 단칼빙은 칼질 횟수를 최소한으로 하고 싶어한다.

주어진 도형을 정확히 한 번씩 자르기 위해 칼을 종이에 최소 몇 번이나 대야 하는지를 계산하는 프로그램을 작성하라. 예를 들어 아래 그림 (a)에 주어진 종이는 (b)처럼 자르면 칼질을 두 번 해야 하지만, (c)처럼 자르면 칼질 한 번만에 자를 수 있다.



Input

입력의 첫 번째 줄에 테스트 케이스의 개수 $C(1 \leq C \leq 100)$가 주어진다.

각 테스트 케이스의 첫 번째 줄에는 종이 위에 그려진 도형의 개수 $N(1 \leq N \leq 100)$이 주어진다. 각 테스트 케이스의 두 번째 줄부터 N개의 줄에 걸쳐 각 줄에 하나씩 도형의 정보가 주어진다. 도형의 정보는 해당 도형이 (x_1, y_1)에서 (x_2, y_2)로 가는 선분일 경우 해당 줄은 "segment x_1 y_1 x_2 y_2"로 표현되며, (x, y)를 중심으로 하고 반지름 r인 원일 경우 "circle x y r"로 표현된다. 이 때, 각 선분의 두 끝 점은 같지 않다.

주어지는 모든 좌표는 $[-1000, 1000]$ 범위 내의 정수이고, 반지름은 $(0, 1000]$ 범위의 정수이다.

Output

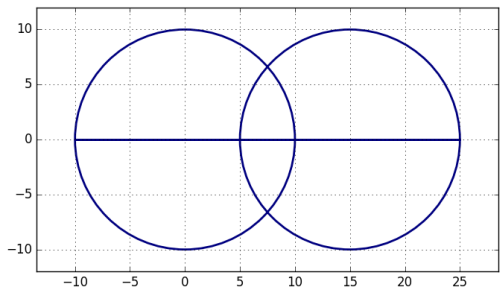
각 테스트 케이스마다 첫 번째 줄에 해당 선분들을 모두 자르기 위해 필요한 칼질의 최소 횟수를 출력한다.

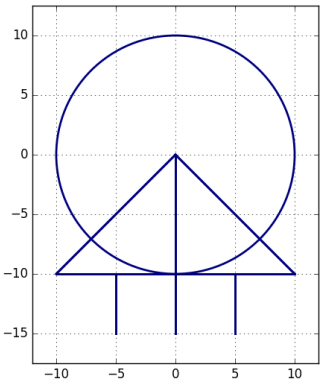
Sample input and output

standard input	standard output
2 4 circle 0 0 10 circle 15 0 10 segment -10 0 5 0 segment 0 0 25 0 7 circle 0 0 10 segment 0 0 -10 -10 segment -10 -10 10 -10 segment 10 -10 0 0 segment 0 0 0 -15 segment -5 -10 -5 -15 segment 5 -10 5 -15 circle 0 5 2	1 3

Notes

예제 입력의 테스트 케이스에서 주어지는 도형들을 그리면 아래 그림을 각각 얻을 수 있다.





Problem C. Coloring Madness

N 개의 정점을 가진 방향성 그래프가 있다. 각 그래프의 정점에 1 에서 N 까지의 번호가 매겨져 있다. 이 그래프에는 $N \times (N-1)$ 개의 간선이 존재하는데, 이는 임의의 서로 다른 두 정점을 연결하는 방향성 간선이 하나씩 있는 것이다. 이 간선을 이용해 x 번 정점에서 y 번 정점으로 이동하는 데에는 $t_{x,y}$ 의 시간이 걸린다. 이 때 $t_{x,y} \neq t_{y,x}$ 일 수도 있음에 유의하라.

또한, 각 정점에는 색이 칠해져 있다. 이 색은 모두 구분이 가능하며, N 가지의 색이 있다. 색에도 1 에서 N 까지의 번호를 붙이도록 한다. 초기에 i 번 정점에 색칠된 색의 번호를 C_i 라고 하자. 경근이는 x 번 정점에서 시작하여 그래프 위를 돌아다니며 정점에 칠해진 색을 바꾸려고 하는데, 그 과정은 다음과 같다 :

1. 모든 정점이 같은 색으로 칠해져 있다면 과정을 종료한다.
2. 만약 경근이가 현재 i 번 정점에 있다면, 다음에 이동할 정점으로 i 번 정점과는 다른 j 를 각각 $\frac{1}{N-1}$ 의 확률로 선택한다.
3. i 번 정점의 색 C_i 를 p 의 확률로 색 C_j 로 칠한다.
4. 경근이가 j 번 정점으로 이동한다. 이 때 $t_{i,j}$ 의 시간이 걸린다. 그 후, j 번 정점의 색 C_j 를 q 의 확률로 2 번 단계에서의 C_i 로 칠한다. 1 번 단계로 돌아간다.

경근이가 색칠하는 과정을 끝내기까지 걸리는 시간의 기댓값을 구하여라.

Input

첫 번째 줄에 정점의 개수를 나타내는 정수 N $(1 \leq N \leq 25)$ 이 주어진다.

두 번째 줄에 두 개의 정수 P, Q $(1 \leq P, Q < 10^6)$ 가 공백으로 구분되어 주어진다. 이는 $p = \frac{P}{10^6}$, $q = \frac{Q}{10^6}$ 임을 나타낸다.

세 번째 줄부터 N 개의 줄에 걸쳐 각 줄에 N 개의 정수가 공백으로 구분되어 주어진다. 이 때, i 번째 줄의 j 번째 정수는 $t_{i,j}$ $(0 \leq t_{i,j} \leq 10^6$ 이고 $t_{i,i} = 0)$ 를 나타낸다.

$N+3$ 번째 줄에 초기 상태의 개수를 나타내는 정수 Q $(1 \leq Q \leq 100)$ 가 주어진다.

$N+4$ 번째 줄부터 Q 개의 줄에 걸쳐 각 줄에 $N+1$ 개의 정수 $C_1, C_2, \cdots, C_N, x$ $(1 \leq C_i, x \leq N)$ 가 공백으로 구분되어 주어진다. 이는 처음 각 정점에 $C_1, C_2, \cdots, C_N$ 의 색이 칠해져 있고, x 번 정점에서 시작할 때 색칠하는 과정을 끝내기는 데 걸리는 시간의 기댓값을 구해야 한다는 의미이다.

Output

Q 개의 줄에 걸쳐, 각 초기 상태에 대한 기댓값을 주어진 순서대로 출력한다. 이 때 기댓값을 기약분수로 나타냈을 때 $\frac{a}{b}$ 라고 하면, $a = bx \pmod{1\,000\,000\,007}$ 을 만족하는 0 이상 $1\,000\,000\,006$ 이하의 x 를 출력해야 한다. 이러한 x 가 존재하는 것이 보장된 입력만 주어지고, x 가 유일하다는 것은 증명되어 있다.

Sample input and output

standard input	standard output
2 500000 500000 0 3 6 0 2 1 2 1 1 2 2	8 10
4 1 999999 0 1 2 3 4 0 5 6 7 8 0 9 10 11 12 0 8 1 2 3 4 1 2 2 4 4 2 2 4 2 4 3 1 3 3 3 4 2 1 1 4 1 2 1 3 2 2 1 1 1 1 3 4 3 2 1 4	927109659 403093474 333557301 397999267 133598154 72237072 0 167611085
10 12345 67890 0 1 2 3 4 5 6 7 8 9 2 0 8 6 4 2 4 6 8 2 3 3 0 3 3 3 3 3 3 3 9 9 7 0 5 5 3 3 1 1 4 5 4 5 0 9 9 9 9 9 7 8 7 8 7 0 8 7 8 7 1 2 1 2 1 2 0 2 1 2 8 8 8 6 6 6 6 0 5 5 2 3 4 2 3 4 2 3 0 4 9 8 7 6 5 4 3 2 1 0 10 10 1 2 3 4 5 6 7 8 9 1 2 10 8 6 4 2 4 6 8 2 2 3 3 10 3 3 3 3 3 3 3 3 9 9 7 10 5 5 3 3 1 1 4 4 5 4 5 10 9 9 9 9 9 5 7 8 7 8 7 10 8 7 8 7 6 1 2 1 2 1 2 10 2 1 2 7 8 8 8 6 6 6 6 10 5 5 8 2 3 4 2 3 4 2 3 10 4 9 9 8 7 6 5 4 3 2 1 10 10	237022287 143958236 496080996 656087792 138754989 710634903 257351692 992622856 107435663 237022287

Problem D. Spotboard

알고스팟의 뛰어난 운영진 분들은 알고리즘 대회에서 스코어보드로 쓸 수 있는 Spotboard를 개발하였다. 예쁜 UI와 더불어 시상식을 할 때 각 팀들의 제출 결과를 보여주며 등수가 올라가는 장면은 아주 인상적이다. 다음은 Spotboard 일부의 스크린 샷이다.

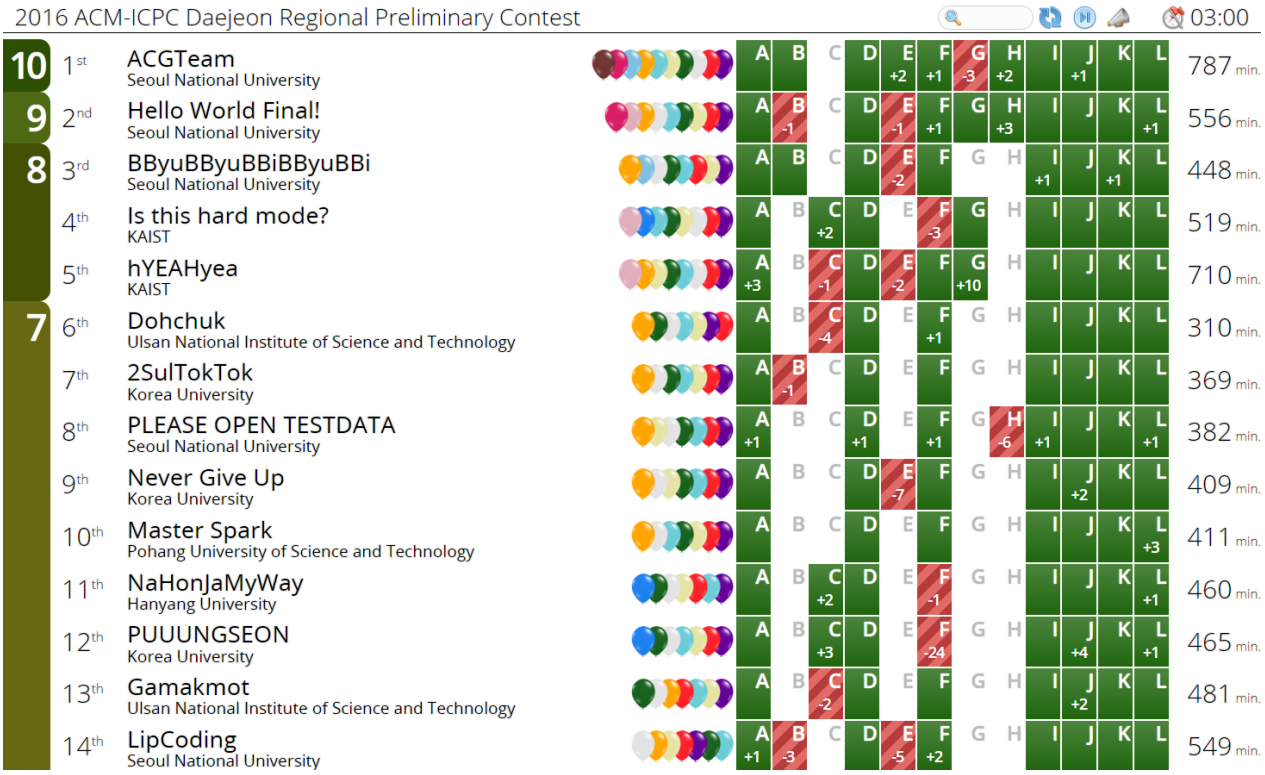


그림 1. Spotboard, 2016 ICPC 한국 예선 대회 결과, Spotboard Project from algospot.com, Being and wookayin

알고리즘 대회는 각 대회마다 규칙이 미세하게 다를 수 있지만, 이 문제에서는 다음의 규칙을 가정한다.

1. 많은 문제를 푼 팀이 등수가 더 높다.
2. 푼 문제의 수가 같다면, total time이 작은 팀이 더 등수가 높다. Total time은 해당 팀이 푼 모든 문제에 대해, 대회 시작부터 각 문제를 풀기까지 걸린 시간의 합이다. 만약 문제를 성공적으로 풀기 전 잘못된 답안을 제출했었다면, 한 번당 20의 페널티가 total time에 추가된다. 결국 풀지 못한 문제에 주어진 페널티는 total time에 반영되지 않는다. 예를 들어, 어느 팀이 1번 문제를 대회 시작 후 10분이 지난 후 맞췄는데 2번째 제출이었을 경우 20 + 10 = 30 의 시간이 total time에 더해진다. 2번 문제를 여러 번의 제출 후에도 대회 종료까지 풀지 못했다면 이 문제에 대해선 0이 더해진다.
3. Total time이 같은 경우 마지막으로 푼 문제를 먼저 푼 팀이 등수가 높다.
4. 이상의 조건에서 등수가 판별이 되지 않을 경우 공동 등수가 된다. 공동 등수는 이후 팀의 등수에 영향을 미치지 않는다. 예를 들어, 2팀이 공동 1등일 경우 다음 팀은 2등이 아니라 3등이 된다.

대회 중 특정 시간 이후에는 스코어보드가 Freeze되어 그 이후에 제출된 모든 답안의 결과는 스코어보드에 반영되지 않으며, 따라서 끝까지 누가 1등을 했는지 알 수 없는 것이 대회의 큰 묘미이다. 시상식에서 Freeze 이후 결과는 다음과 같은 방식으로 보여주게 된다.

1. Freeze된 시점에서의 스코어보드 상태에서 시작한다. 스코어보드는 위 규칙을 따라 계산된 등수의 오름차순으로 팀들을 정렬하며, 등수가 같을 경우 팀 번호의 오름차순으로 정렬한다.
2. 스코어보드에서 가장 아래에 있는 팀을 선택한다. 1번 문제부터 차례대로 순회하며, 이 팀의 미공개 답안 결과를 하나씩 발표한다. 이 때 정답이 나올 경우, 이 정답을 반영해 등수가 재계산되고, 위 규칙에 따라 스코어보드가 재정렬된다.
3. 한 팀이 정답을 제출한 이후에도 계속 해당 문제에 대해 답안을 제출했을 경우, 정답 이 후의 답안들은 무시된다.
4. 가장 아래 있는 팀의 모든 제출 결과가 공개되고 나면, 그 위 팀으로 올라가 위 과정을 반복한다.

대회 운영을 하고 있던 태현이는 시상식에 일어나는 이벤트를 미리 시뮬레이션 해보고 싶다. 태현이를 도와 시상식에서 정답 제출이 나올 때마다 등수가 어떻게 변화하는지 계산해보자.

Input

첫 번째 줄에 팀 수를 나타내는 수 N과 전체 문제 수 M, 전체 제출의 수 K, Freeze가 된 시간 F가 주어진다. (제출된 시간 t가 F 이상일 경우 해당 답안의 결과는 시상식 전까지 공개되지 않습니다.) ($1 \leq N \leq 500$, $1 \leq M \leq 20$, $1 \leq K \leq 20\,000$, $0 \leq F \leq 1\,000$)

두 번째 줄부터 K개의 줄에 걸쳐 각 줄에 하나씩 답안의 정보들이 제출 순서대로 주어진다. 각 답안의 정보는 "T t p r" 형태의 공백으로 구분된 네 정수로 이루어져 있으며, T는 제출 시간, t는 팀 번호, p는 문제 번호를 나타낸다. r은 0 혹은 1이며, 0은 해당 답안이 오답, 1은 정답인 경우를 각각 나타낸다. ($0 \leq T \leq 1\,000$, $1 \leq t \leq N$, $1 \leq p \leq M$)

Output

첫 번째 줄에 시상식 중에 공개되는 정답 답안의 개수 X를 출력한다.

두 번째 줄부터 X개의 줄에 걸쳐 한 줄에 하나씩, 공개되는 순서대로 정답의 정보를 세 개의 정수를 공백으로 구분하여 "A B C" 형태로로 출력한다. 이 때, A는 팀 번호, B는 해당 정답이 공개되기 전에 팀의 등수, C는 해당 정답이 공개된 후 해당 팀의 등수를 나타낸다.

Sample input and output

standard input	standard output
3 2 6 3 1 1 1 1 2 1 1 1 3 1 1 0 4 2 2 1 10 2 1 1 11 1 2 0	2 2 2 2 2 2 1
3 2 6 2 1 1 1 1 1 2 1 1 1 3 1 1 2 1 2 0 2 2 2 1 3 1 2 1	2 2 1 1 1 2 2

Problem E. Master of Baduk

Baduk is an ancient board game which requires deep strategy and tactics, where two players (Black and White) exchange their turns to place a stone in one of empty positions of the board. Black play with black stones, and White with white stones. You can also 'pass' to give up your opportunity to place a stone.

Baduk is known to be notoriously hard to solve with computers, but don't worry! We will be dealing with a simplified and highly abstracted version of the game. See below for the rules of the game.

In this version of Baduk, each player has one stone group. Each empty position which is adjacent to a stone is called a liberty. We can classify each liberty as one of the following three types:

- Shared liberties (S) – empty positions adjacent to both colors of stones.
- Inner liberties (I) – empty positions completely enclosed within a stone group.
- Outer liberties (O) – empty positions that is adjacent to a stone, which are not either Shared or Inner liberties.

Using these definitions, we can summarize the entire state of the game board into 5 non-negative integers: BO (# of Outer liberties of Black), BI (# of Inner liberties of Black), WO (# of Outer liberties of White), WI (# of Inner liberties of White), S (# of liberties shared by White and Black).

At each turn, a player can choose one of the numbers and decrease it by 1. There are a few rules:

1. No move can result in a negative number of liberties. (i.e. BO, BI, WO, WI, S should all be 0 or greater after a move)
2. To decrease BI from 1 to 0, both BO and S must be 0 already.
3. To decrease WI from 1 to 0, both WO and S must be 0 already.
4. A player can pass his/her turn.

The winner of the game is determined as follows.

- A player loses when it does not have any liberties. For example, when $BO = BI = S = 0$, White wins.
- When all five numbers become 0 by decreasing S the last time, the last person who made the move wins.

Write a program that, given the 5 numbers, determine who will win assuming Black plays first, and both players play optimally. Note that there are cases where no player can win the game.

Input

The first line of the input will contain the number of test cases N. ($1 \leq N \leq 10\,000$) N lines follow, with 5 non-negative integers, each representing BO, BI, S, WO and WI, in this order. ($0 \leq BO, WO, S \leq 10^9$, $0 \leq BI, WI \leq 2$, $BO + BI + S \geq 1$, $WO + WI + S \geq 1$)

Output

For each test case, print a single line containing the winner of the game ("White" or "Black"). In case no one can win the game, print "Neither",

Sample input and output

standard input	standard output
3 3 0 0 3 0 3 0 0 4 0 0 0 2 0 0	Black White Neither

Problem F. 요새 2

중세의 성과 요새들은 보안을 튼튼히 하면서도 더 넓은 영역을 보호하기 위해 여러 개의 성벽을 갖고 있었다고 한다. 전세계에서 가장 편집증이 심한 영주가 지은 스트로고(Strawgoh) 요새는 이의 극치를 보여준다.

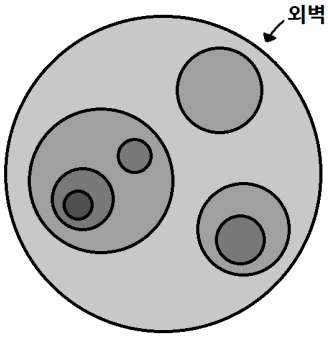


그림 1. 요새

이 요새는 그림 1과 같이 커다란 원형 외벽 내에 여러 개의 원형 성벽이 겹겹이 지어진 형태로 구성되어 있는데, 어떤 성벽에도 문이 없어서 성벽을 지나가려면 사다리를 타고 성벽을 오르내려야 한다. 욕심이 많은 이 성의 영주는 각 주민이 성벽을 한 번 오르내리는 데에 통행세를 거두고 있어 원성이 자자하다. 그렇기에 이에 반발하는 몇몇 주민들은 영주를 성토하기 위한 비밀 집회를 계획하고 있고, 요새의 한 위치에 모이기로 했다. 이 때, 모든 집회에 참여하는 모든 주민이 지불할 통행세의 총합을 최소화 하고 싶어 한다.

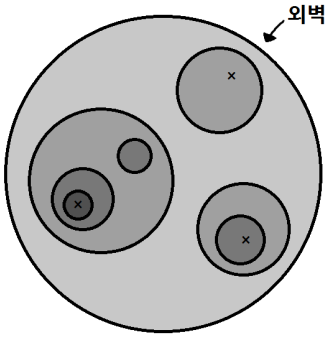


그림 2. 상황 1

예를 들어 그림 2 처럼 세 명의 주민이 비밀 집회를 하려고 한다고 생각해 보자. × 표시는 각 주민의 현재 위치를 나타낸다. 실제로는 각 성벽의 통행세가 다를 수 있지만, 이 상황에서는 모든 성벽의 통행세가 같다고 생각하자.

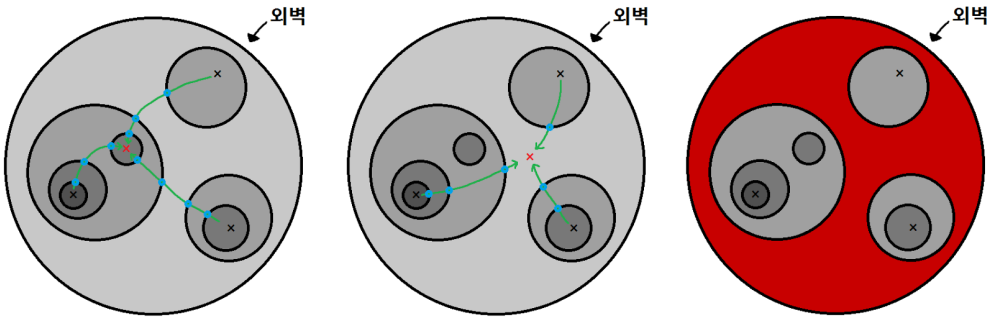


그림 3. 상황 1 설명

그림 3은 상황 1에서 비밀 집회를 가지는 위치에 따른 결과를 나타낸다. 파란색 점이 성벽을 오르내리는 지점이다. 가장 왼쪽의 그림은 총 10회를 오르내려야 하며 최적의 경우가 아니다. 중간에 있는 그림은 총 6

회를 오르내리야 하며 이 때 가장 최적이 된다. 가장 오른쪽의 그림은 최소화된 통행세로 집회를 할 수 있는 영역을 붉은 색으로 칠한 그림이다.

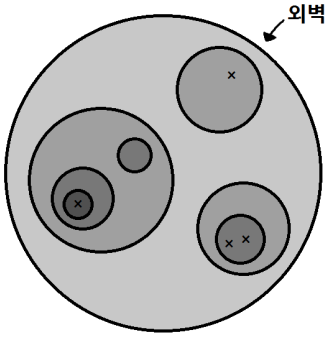


그림 4. 상황 2

다음으로 그림 4는 그림 2의 경우에서 한 명의 주민이 더 끼어든 상황을 나타낸다. 어떤 영역 내에(심지어 같은 위치에도) 여러 주민이 있을 수 있기 때문에 이런 경우도 충분히 가능하다.

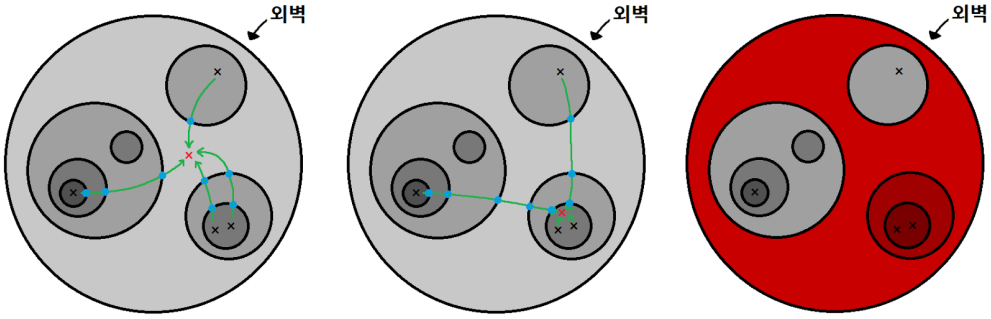


그림 5. 상황 2 설명

그림 5는 상황 2에서 비밀 집회를 가지는 위치에 따른 결과를 나타낸다. 가장 왼쪽의 그림은 총 8회를 오르내려 최적인 경우이다. 그러나 이 상황에서는 다른 성벽 영역에서 집회를 해도 최적이 될 수 있는데, 중간에 있는 그림이 이를 나타내며 역시 똑같이 8회를 오르내리면 된다. 가장 오른쪽의 그림은 최소화된 통행세로 집회를 할 수 있는 영역을 붉은 색으로 칠한 그림이다.

성벽들의 정보와 주어질 때 비밀 집회에 참가하는 사람들의 위치를 나타내는 상황이 여럿 주어질 때, 어떤 위치로 모든 사람이 모이기 위해 성벽을 오르내리는데 드는 통행세의 총합을 최소화 하고, 그것이 가능한 집회 장소 영역의 넓이를 구하는 프로그램을 작성하라.

Input

첫 번째 줄에 성벽의 수를 나타내는 정수 N ($1 \le N \le 10^5$)이 주어진다.

두 번째 줄부터 N개의 줄에 걸쳐 i번째 줄에 i번 성벽의 정보인 네 정수 x_i, y_i, r_i, c_i ($-10^8 \le x_i, y_i \le 10^8$, $1 \le r_i \le 10^8$, $1 \le c_i \le 10^6$)가 공백으로 구분되어 주어진다. 이 때, i번 성벽은 (x_i, y_i)를 중심으로 하는 반지름 r_i인 원형으로 설치되어 있으며 오르내리는데 드는 통행세가 c_i임을 나타낸다. 편의상 모든 성벽의 두께는 0이라고 가정하며, 어느 두 다른 성벽에서 잡은 임의의 두 점은 주민의 통행을 위해 최소 1이상의 거리를 가지는 것이 보장된다. 입력에서 가장 반지름이 큰 원은 다른 모든 원을 포함하는 외벽이다. 외벽은 $(0, 0)$을 중심으로 하는 반지름 10^8인 원 안에 포함된다.

$N+2$번째 줄에는 상황의 수를 나타내는 정수 Q ($1 \le Q \le 2 \times 10^5$)가 주어진다.

$N+3$번째 줄부터는 Q개의 상황에 대한 정보가 주어지는데, 각 상황은 다음과 같은 형식으로 주어진다.

j번째 상황의 첫 번째 줄에는 비밀 집회에 참가하는 주민의 수 M_j ($1 \le M_j \le 2 \times 10^5$)가 주어진다. M_j의

총합은 2×10^5을 넘지 않는다.

j번째 상황의 두 번째 줄부터 M_j개의 줄에 걸쳐 k번째 줄에 두 정수 $X_{j,k}$, $Y_{j,k}$ $(-10^8 \leq X_{j,k}, Y_{j,k} \leq 10^8)$가 공백으로 구분되어 주어진다. 이는 이 상황에서 k번째 주민이 있는 곳의 좌표가 $(X_{j,k}, Y_{j,k})$임을 나타낸다. 같은 좌표에 위치한 주민이 여럿 있을 수 있다. 모든 주민은 외벽 안에 위치하며, 최소한 1이상의 거리는 이동해야 성벽에 도착할 수 있는 위치에 있다.

Output

각 상황에 대해 하나의 줄에 두 정수를 공백 하나로 구분하여 출력하여야 한다. 이 때, 첫 번째 정수는 j번째 상황에 주어진 M_j명의 주민이 요새의 한 위치에 모이기 위한 통행세 총합의 최솟값, 두 번째 정수는 통행세 총합을 최소화하는 집회 장소 영역의 넓이가 $R\pi$일 때, R을 나타내는 값이다.

Sample input and output

standard input	standard output
8 13 12 5 1 15 15 10 1 12 12 3 1 21 15 20 3 30 24 5 1 32 10 6 1 19 19 2 1 32 9 4 1 2 3 12 12 32 25 33 9 4 12 12 32 25 33 9 31 8	6 239 8 275

Problem G. 총기 밀수

돈이 매우 많은 부자 정민이는 자신이 가진 많은 돈을 지키기 위해 총을 많이 구매해 두기로 했다. 총기 소지가 불법인 한국에서는 정상적으로 총기를 구매할 수 없기 때문에, 정민이는 유능한 총기 밀수업자 경근이에게 총기 밀수를 의뢰하기로 했다.

경근이는 총 N일 동안 총기를 제공해 줄 수 있음을 공언했다. 각 날마다 총기를 밀수하는 데에는 외부적인 요인에 따라 난이도가 다를 수 있기에, 밀수해줄 수 있는 총기의 수와 총기 당 비용이 달라질 수 있다. 이에 경근이는 i번째 날에 제공 가능한 총기의 수가 총 k_i개이며, 난이도에 따라 밀수하는 비용을 결정하는 두 개의 상수 a_i, b_i가 있음을 정민이에게 알려주었다. 하루에 밀수해야 하는 양이 늘어나면 늘어날수록 밀수의 난이도는 높아지기 때문에, 더 많은 총기를 제공해야 하는 경우 a_i, b_i에 따라 총기당 가격이 올라간다. 정확히 말해서, i번째 날에 x ($\leq k_i$)개의 총기를 밀수해야 한다면 총기 하나당 밀수하는 데 드는 가격이 $a_i x + b_i$원이 되어, 총 $x(a_i x + b_i)$의 비용을 들여야 x개의 총기를 밀수해줄 수 있다.

정민이는 이 일에 최대 C원을 투자할 것이며, 최대한 많은 총기를 사고 싶어 한다. 경근이가 알려준 정보에 따라 최적의 구매 전략을 정할 때, 구매할 수 있는 총기의 개수는 최대 몇 개일까?

Input

첫 번째 줄에 두 정수 N, C ($1 \leq N \leq 10^5$, $1 \leq C \leq 2 \times 10^{18}$)가 공백으로 구분되어 주어진다. 다음 N개의 줄의 i번째 줄에는 세 양의 정수 k_i, a_i, b_i ($1 \leq k_i(a_i k_i + b_i) \leq 2 \times 10^{18}$)가 공백으로 구분되어 주어진다.

Output

첫 번째 줄에 최대 C원을 투자하여 구매할 수 있는 총기 개수의 최댓값을 출력한다.

Sample input and output

standard input	standard output
2 11 3 2 1 2 1 2	3
2 11 3 2 1 1 1 2	2

Notes

첫 번째 예제는 첫 번째 날에 한 개(3원), 두 번째 날에 두 개(8원)을 밀수하도록 지시하는 것이 최대한 많이 총기를 구매하는 방법이다.

두 번째 예제는 첫 번째 예제에 비해 두 번째 날에 밀수할 수 있는 총기의 개수가 하나 줄었다. 그러므로 첫 번째 날에 두 개를 밀수하거나(10원), 두 날 모두 한 개씩을 밀수하는 수(6원) 밖에 없다.

Problem H. 놀이 공원

21xx년, 오렌지는 우주 최고의 놀이 공원인 네버랜드에 놀러 왔다! 네버랜드에서 가장 유명한 놀이 기구는 약 11km의 길이를 약 1분만에 주파하는 롤러 코스터 A-Express! 이 놀이기구를 타지 않으면 죽을 것 같다는 기분이 든 오렌지는 네버랜드에 도착하자 마자 바로 A-Express를 향해 달렸다! 그러나 A-Express는 매우 유명한 놀이 기구라 오렌지가 입구에 도착 했을 때 이미 줄이 가득 차 있어 줄에 들어가기 위한 줄을 서서 기다려야 했다. 이 기다리는 시간을 이용해 네버랜드가 줄을 어떻게 관리하는지 알아보자.

네버랜드는 줄을 선 사람들의 관리를 위해 줄을 특정 크기의 정사각형 격자로 나누어, 한 칸에 한 사람만 들어가 있도록 줄을 세운다. A-Express의 줄을 이루는 격자의 변은 남북 방향, 동서 방향에 평행하다. 이 격자에서 남북방향으로 있는 칸의 개수는 N개, 동서 방향으로 있는 칸의 개수는 M개로 $N \times M$크기의 격자를 이루며, 이 중에서 몇 개의 칸이 줄로 사용되고 있다. 앞으로 북쪽에서 i번째, 서쪽에서 j번째에 있는 격자를 (i, j)로 표현하도록 하자. 이 때, 줄의 입구는 $(1, 1)$의 왼쪽 변이고 A-Express를 타는 곳은 (N, M)의 오른쪽 변이다.

다음은 $N = 3$, $M = 4$인 예이다. $N \times M$크기의 격자에서 줄로 사용되는 칸에는 양의 정수가 적혀 있고, 수가 적힌 순서대로 이동하면 줄이 된다. 양의 정수가 적히지 않은 칸은 표지판이나 쓰레기통 등으로 사용되는 칸으로 사람이 서는 줄로는 사용되지 않는 칸이다. 결국 사람들이 기다리는 줄은 인접한 칸이면서, 각 칸에 적힌 정수의 차이가 정확히 1이 나는 것들의 정중앙을 연결하는 선분들로 표현할 수 있다. 그림 1은 이런 식으로 그려진 줄을 나타낸다.

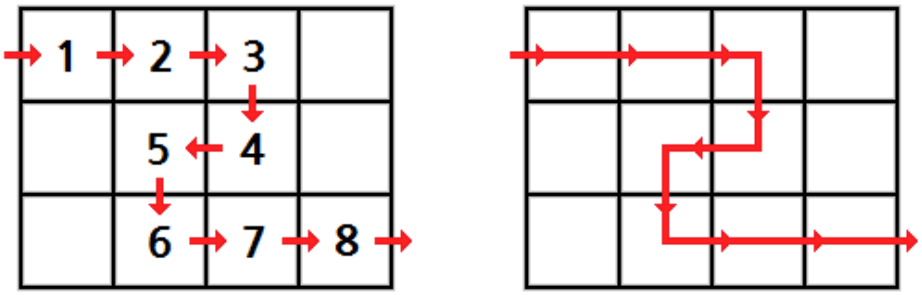


그림 1

드디어 오렌지가 A-Express의 줄에 들어갈 차례가 되어 A-Express의 줄에서 1이 써진 칸$((1, 1))$에서 줄서기를 시작했다. 그 순간 오렌지에게 놀라운 메시지가 왔는데, 평소에 잘 알고 지내던 멜론도 이 줄에 서있어서 오렌지가 들어오는 것을 보았다는 것이다. 또한, 현재 S가 적힌 칸에 멜론이 있다는 사실도 알려주었기 때문에 오렌지도 멜론을 찾아보기로 했다.

줄에 서 있는 사람들의 이동은 다음과 같다. 먼저, 오렌지가 1이 써진 칸에 있을 때, 줄에 속한 다른 모든 칸에는 사람이 한 명씩 있다. A-Express는 최고의 경험을 위해 한 번의 운행에 한 사람만을 태우는 롤러코스터이기 때문에, 한번에 줄의 가장 앞에 있는 한 사람만 빠진다. 위의 예에서는 8이 써진 칸$((3, 4) = (N, M))$에 있는 사람이 A-Express에 탑승하기 위해 줄에서 빠지는 것이다. 이 사람이 빠지는 동시에, 순간 이동 기술을 이용해 사람들은 직접 몸을 움직일 필요 없이 다음 칸으로 자동적으로 이동한다. 그리고 인기가 많은 A-Express의 특성상 동시에 1이 써진 칸에 다른 사람이 순간 이동해 들어온다.

그림 2와 그림 3은 각각 $S = 6$, $S = 7$일 때에 대한 사람들의 이동을 그린 그림이다. 격자는 사람에 비해 매우 크기 때문에, 앞으로는 모든 사람을 점과 같은 크기를 가지는 것으로 생각한다. 주황색 점은 오렌지, 초록색 점은 멜론, 검은색 점은 다른 사람들이다. 파란색 선은 오렌지가 멜론을 바라보는 시야를 나타낸다. 오렌지가 바라보는 방향에 다른 사람이 있으면, 그림 3처럼 오렌지는 멜론을 못 볼 수 있다. 오렌지의 시야를 가리는

사람은 붉은색 점으로 표시하고, 이후의 보이지 않는 시야를 갈색 선으로 표시하였다.

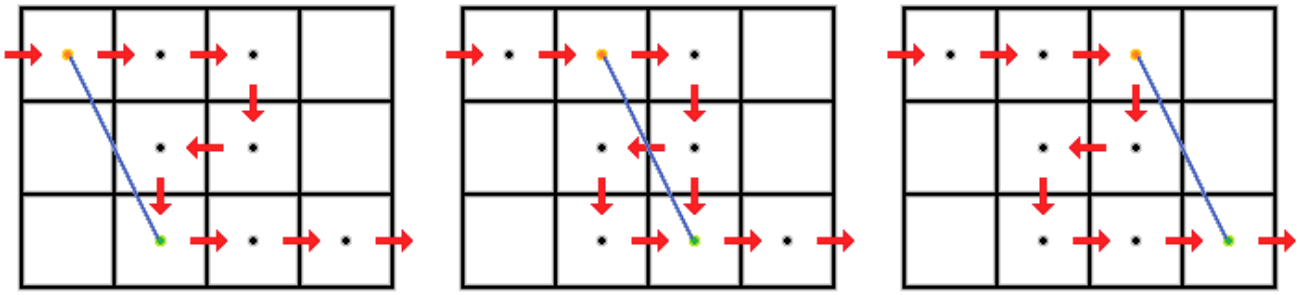


그림 2. $S = 6$, 오렌지는 총 3번 멜론을 볼 수 있다.

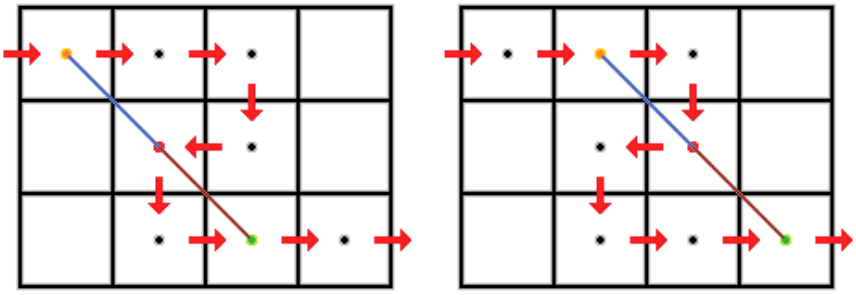


그림 3. $S = 7$, 오렌지는 한 번도 멜론을 볼 수 없다.

줄을 서는 모든 사람이 칸의 정중앙에서 벗어나지 않으며, 빈 칸은 시야를 막지 않는다고 가정한다. 이 때 오렌지가 줄에 들어온 시점부터 멜론이 A-Express에 탑승하기 직전까지 오렌지에게서 멜론이 보이는 횟수를 구하고, 그 때 오렌지가 있었던 위치에 어떤 정수가 적혀 있는지 출력하는 프로그램을 작성하라. 그림 2의 경우에는 1, 2, 3이 적힌 칸으로 세 개, 그림 3의 경우에는 어떤 칸도 없는 것이 답이다.

Input

첫 번째 줄에 세 정수 N, M, S가 공백 하나로 구분되어 주어진다. ($1 \le N, M \le 20$) N, M은 A-Express의 줄을 포함하는 격자의 크기를 나타내며, S는 오렌지가 1이 써진 칸에 들어 왔을 때, 멜론이 있는 칸에 써진 수를 나타낸다.

두 번째 줄부터 N개의 줄에 걸쳐 각 줄에 격자의 정보인 M개의 음이 아닌 정수가 공백 하나로 구분되어 주어진다. 이 때, i번째 줄의 j번째 수는 격자의 칸 (i, j)에 써진 양의 정수인 $P_{i,j}$를 나타낸다. 만약 $P_{i,j} = 0$이면, (i, j)는 줄에 포함되지 않는 칸으로 생각한다. $P_{1,1} = 1$이고, $2 \le S \le P_{N,M}$를 만족한다. P는 문제에서 주어진 A-Express의 줄을 올바르게 나타낼 수 있도록 주어진다.

Output

첫 번째 줄에 오렌지가 멜론을 볼 수 있는 횟수를 출력한다. 이 횟수를 C라고 하면, 다음 C개의 줄에는 오렌지가 어떤 칸에 있을 때 멜론이 보이는지 그 칸에 적힌 정수를 각각 오름차순으로 출력한다.

Sample input and output

standard input	standard output
3 4 6 1 2 3 0 0 5 4 0 0 6 7 8	3 1 2 3
3 4 7 1 2 3 0 0 5 4 0 0 6 7 8	0
3 4 5 1 8 9 10 2 7 6 11 3 4 5 12	2 3 6
5 3 5 1 2 3 0 0 4 7 6 5 8 0 0 9 10 11	6 1 2 3 5 6 7

Problem I. Circle Routes

The city of Algospot has lots of circle bus routes. A circle bus route means the bus' s route ends at where it started. Donghyun runs a bus company, which will be servicing a new circle bus route. He wants to find out the minimum number of buses he needs to meet the bus schedule. The schedule works like this:

1. Every day, the first bus departs from the terminal at time S.
2. After that, a new bus will depart from the terminal periodically, with a fixed interval D. For example, a new bus will depart at time S, $S + D$, $S + 2D$, and so on.
3. The last bus of the day departs the terminal at E. (You can assume the difference between E and S is a multiple of D.)
4. Each bus takes a fixed amount of time, C, to traverse the route and come back to the terminal.

For example, let' s assume the following scenario: the first bus departs at 05:50, and a new bus departs every 30 minutes until the last bus departs at 20:20. How many buses do we need, if it takes 65 minutes for a bus to traverse the route?

The first bus departs at 05:50 and returns to the terminal at 06:55. The second bus departs at 06:20 and returns at 07:25. The third bus departs at 06:50, and returns at 07:55. The fourth bus must depart at 07:20 – but at this time, the first bus is already back in the terminal, so Donghyun only needs 3 buses to serve this schedule.

Write a program to find the minimum number of buses.

Input

The first line of the input gives the number of test cases T. ($1 \leq T \leq 10\,000$) T lines follow, each containing a test case. The test case consists of four strings S, E, D and C, in the form `HH:MM`. S, E ($S \leq E$) represent the time of the first and the last bus, respectively. D and C ($0 < D$, $0 < C$) each represent the time interval between buses, and the time it takes a bus to traverse the route.

In each string `HH:MM`, `HH` represents the hour part of the time (or time interval), and `MM` represents the minute part. Concretely, `HH` will be a zero-padded integer between 0 and 23 (00, 01, $\cdots$, 23). Likewise, `MM` will be a zero-padded integer between 0 and 59 (00, 01, $\cdots$, 59).

Output

Print a single line per each test case, containing the minimum number of buses required for the schedule.

Sample input and output

standard input	standard output
4 05:50 20:20 00:30 01:05 05:50 20:20 00:30 01:00 05:00 20:00 15:00 16:00 01:00 23:00 00:10 21:39	3 2 2 130

Problem J. Time to Chicken

알고치킨은 요즘 가장 떠오르는 스타트업 치킨 업체이다. 그 비결은 잘 튀겨낸 치킨에도 있지만, 국내 요식업에서 빼놓을 수 없는 배달 비용을 최소화하여 치킨 가격을 낮춰낸 것이 유효했다.

알고치킨의 치킨 배달 방식은 상당히 독특한데, 다음과 같다:

1. 알고치킨이 존재하는 도시는 그래프의 형태로 주어지며, 알고치킨의 체인점과 고객들은 각 정점 위에 존재할 수 있다.
2. 고객은 특정 알고치킨 체인점에 전화하여 치킨을 주문할 수 있다.
3. 모든 체인점의 알고치킨은 동일한 품질을 유지하는 것에 굉장히 노력을 기울여, 그 결과 어느 체인점에서 해당 고객에게 배달을 하더라도 고객은 늘 만족할 수 있게 되었다.
4. 물론 고객들은 자신이 지정한 특정 맛집에서 배달을 받는다고 믿고 있다. 따라서 만약 고객이 지정한 체인점에서 고객까지 배달할 수 있는 길이 없다면 해당 주문은 무시된다.
5. 각 체인점 사장님들의 수익 보전을 위해 각 체인점은 받은 주문 수 만큼의 발주를 하고 싶어한다.

위와 같은 성질을 이용하여 많은 주문이 들어올 경우 알고치킨은 고객이 선택한 체인점과 실제 배송을 담당하는 체인점을 잘 조율하여 배송 비용을 최적화 할 수 있다. 배송 비용이란 모든 주문들을 배송하는데 드는 체인점으로부터 고객까지의 거리의 합과 같다.

당신은 알고치킨의 데이터 분석가로써 현 알고치킨 알고리즘에 대해 개선안을 제안해야 하는 위치에 놓였다. 그 첫번째 단계로, 도시 정보와 주문 정보들이 주어질 때 각 경우에 대하여 최적의 배송 비용을 알아내자.

Input

첫 줄에는 도시를 이루는 정점의 크기 N $(1 \leq N \leq 30)$이 주어진다.

이어서 N줄에 걸쳐 N개의 정수가 입력된다. $i(1 \leq i \leq N)$번째 줄의 j $(1 \leq j \leq N)$번째 정수 $c_{i,j}$는 정점 i로부터 정점 j까지 연결된 길의 길이 $(1 \leq c_{i,j} \leq 1\,000$ 혹은 $c_{i,j} = -1)$를 나타내며 -1일 경우 두 정점 사이에는 직접 연결된 길이 없다.

이어서 주문의 수 Q $(1 \leq Q \leq 50\,000)$이 주어진다.

이어서 Q줄에 걸쳐 두 개의 정수 s, e $(1 \leq s, e \leq N)$이 주어지며, 이는 정점 e에 있는 고객이 정점 s에 있는 체인점에게 주문했음을 의미한다.

Output

Q줄에 걸쳐 최적의 배송 비용을 출력한다. 각 i $(1 \leq i \leq Q)$번째 줄에는 1번째 주문부터 i번째 주문까지를 모두 배송할 수 있는 최적의 배송 비용을 출력해야 한다.

Sample input and output

standard input	standard output
4 0 -1 1 50 -1 0 100 2 -1 -1 0 -1 -1 -1 -1 0 2 1 4 2 3	50 3

Problem K. Maximal K-edit Subset

강산이 한번 변하고도 남는, 무려 12년 반이라는 기간 동안 한 학교에서 학사부터 박사까지 마친 Hwan은 우연찮은 기회를 잡아 먼 타향의 어느 의대 부속 연구소에서 박사후과정을 하게 되었다. 2개월 가량이 지나 어느 정도 적응을 마친 어느 날, 그가 속한 연구실의 교수는 그의 자리에 들러 갑작스레 새로운 프로젝트를 진행하자고 이야기 하였고, 이에 대한 간략한 설명을 전달하고 자리를 떠났다. 하지만 너무나 새로운 분야의 프로젝트에 참가하게 되었기 때문에 당장은 자신이 해야 하는 일이 무엇인지 알 수 없었으나, 박사과정에서 익힌 며칠간의 꾸준한 구글 검색 엔진을 통해 이번 프로젝트를 전산학의 문제로 치환하여 풀 수 있다는 것을 알게 되었다. Hwan이 정의한 문제인 maximal k-edit subset이라는 문제는 다음과 같다.

우선 알파벳 `A`, `C`, `G` 그리고 `T`로 구성된 문자열들로 이뤄진 L과 S가 주어진다. 이때 L의 모든 원소의 문자열의 길이는 모두 같고, S의 모든 원소의 문자열의 길이는 같으며, 그리고 S 문자열의 길이는 L보다 길거나 같다고 가정한다. 여기서 S의 부분 집합 중, 모든 부분집합의 원소들이 적어도 한 개 이상의 L에 속한 문자열과 k 이하의 Edit Distance(두 문자열을 일치하도록 만들기 위한 연산의 수, 이 때 문자의 삽입, 삭제 혹은 변경을 하는 경우를 한번의 연산으로 간주한다)를 갖는 부분문자열(substring)을 포함하고 있으면, 해당 부분집합을 k-edit subset으로 정의할 수 있다. 이 문제에서는 가능한 k-edit subset이 되는 경우 중 최대 크기를 찾아야한다.

Hwan은 문제를 잘 정의했지만, 다른 프로젝트도 진행하고 있던 중이였기 때문에 이 문제를 풀 코드를 작성할 시간이 없었고, 아직 영어의 장벽을 느끼고 있는지라 다른 연구실 동료들에게 부탁하기도 힘든 상태이다. 결국 그는 한국에 있는 문제 해결과 코딩에 조예가 깊고 친한 후배였던 당신에게 도움을 구하기로 했으며, 당신은 그의 사탕발림에 넘어가게 되어 이 문제를 해결해야 한다. 그리고 그는 마지막으로 한마디를 더 덧붙였다. "데이터의 개수가 방대하니까 프로그램 수행시간이 아주 빨라야 할 거야!"

Input

첫 번째 줄에 세 개의 정수 N_L, N_S, k ($1 \le N_L \le 500$, $1 \le N_S \le 2\,500$, $1 \le k \le 4$)가 공백으로 구분되어 주어진다. N_L은 L의 원소의 수이고, N_S는 S의 원소의 수를 의미한다.

두 번째 줄부터 N_L개의 줄에 걸쳐 각 줄에 하나씩 L에 속한 문자열이 주어진다. 각 문자열의 길이는 $k+1$ 이상 30이하이다.

N_L+2번째 줄부터 N_L개의 줄에 걸쳐 각 줄에 하나씩 S에 속한 문자열이 주어진다. 각 문자열의 길이는 $k+1$이상 50이하이다.

입력으로 주어지는 모든 문자열은 반드시 `A`, `C`, `G`, `T`로만 구성된다.

Output

첫 번째 줄에 주어진 입력에 대한 maximal k-edit subset의 크기를 출력한다.

Sample input and output

standard input	standard output
3 6 0 ACGT ACCT GCGC AAACGTCC AAGCGCCC AAACGTCC AAACCTCC AAGCGCCC AAAATTTT	5
3 6 1 ACGT ACCT GCGC AAACTCCC AAGCGCGC AAACGTCC AAACCTCC AATTTTCC AAAATTTT	4

Problem L. PPAP

I have a pen
I have an apple
Apple pen!

I have a pen
I have a pineapple
Pineapple pen!

Apple pen,
Pineapple pen,
pen-pineapple-apple-pen

Piko just can't stop singing along to PPAP! PPAP is an extremely addictive song which became an internet meme. Piko is especially obsessed with the song's chorus part, which just repeats the words "pen-pineapple-apple-pen". One afternoon, Piko started repeatedly singing PPAP's chorus part, while writing down the lyrics in a notebook as he sang. Piko delimited words with a dash(-) and each chorus part with a forward slash(/). The first 30 characters Piko has written down can be seen below:

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	p	e	n	-	p	i	n	e	a	p	p	l	e	-	a	p	p	l	e	-	p	e	n	/	p	e	n	-	p	i	...

The song was so addictive, Piko spent the entire afternoon writing down the lyrics! While Piko has gone to the bathroom, his friend Taro came over and discovered Piko's notebook. He started wondering the following: how many times does a given pattern P appear as a substring of Piko's string, within a given range?

Input

Your program needs to solve the problem for a single test case. A test case is given in a single line as two integers L, R and a string P, separated by a single space. ($1 \leq L \leq R \leq 1\,000\,000$). L and R respectively represents the index of the first and the last character of the range Taro is interested in. (The first character of the string has index 1.) P is a string that only consists of lowercase alphabet letters, dashes, and forward slash, and its length will be between 1 and 10 000, inclusive.

Output

Print the number of times P appears as a substring of Pico's string, starting on or after index L and ending on or before index R.

Sample input and output

standard input	standard output
1 25 pen	2
1 48 -pen	2
1 999999 melon	0