UCPC 2021

Finals

Official Problemset

전국 대학생 프로그래밍 대회 동아리 연합
여름 대회 2021 본선

주최 전국 대학생 프로그래밍 대회 동아리 연합

후원 SOLVED.AC, 한빛미디어 Hanbit Media, Inc., ALGOSPOT, YoungJin.com 영진닷컴, DEVSISTERS, VUNO, kakao, SAMSUNG SOFTWARE MEMBERSHIP, NAVER D2, 오늘의집, STARTLINK

오시영 윤준하 이재열 Team Cafe Mountain

문제 목록

문제지에 있는 문제가 총 12문제가 맞는지 확인하시기 바랍니다.

A A+B와 쿼리
B Distance on Triangulation 2
C UCP-Clustering
D 츠바메가에시
E 가위바위보 버블 정렬
F 간단한 문제
G 돌 가져가기 2
H 봉화대
I 붉은색 푸른색
J 시험 문제 출제
K 은퇴한 자들의 게임
L Make Different

I번 문제를 제외한 모든 문제의 메모리 제한은 1GB로 동일합니다.

문제 A. A+B와 쿼리

시간 제한	2 초
메모리 제한	1024 MB

음이 아닌 N자리 10진수 정수 A와 B가 있다. 단, 각 수의 앞부분에 불필요한 0이 있을 수도 있다. 이때 $N+1$자리 정수 $C=A+B$를 생각하자. 마찬가지로 $A+B$의 값이 $N+1$자리가 안 될 경우 앞에 불필요한 0을 채워 $N+1$자리로 만든다.

Q개의 쿼리를 순서대로 처리해야 한다. 각 쿼리는 다음 중 하나이다.

- `A i d`: A의 오른쪽에서부터 i번째 자리 숫자를 d로 바꾼다.
- `B i d`: B의 오른쪽에서부터 i번째 자리 숫자를 d로 바꾼다.

각 쿼리를 수행할 때마다 $C=A+B$를 다시 계산하고, C의 각 자리 숫자 중 몇 개의 숫자가 바뀌었는지 출력하시오.

입력

첫째 줄에는 N과 Q가 주어진다. $(1 \le N, Q \le 300\,000)$

둘째 줄에는 A, 셋째 줄에는 B가 주어진다.

다음 Q줄에는 한 줄에 하나씩 쿼리가 주어진다. 모든 쿼리에서 $1 \le i \le N$, $0 \le d \le 9$이다.

출력

각 쿼리를 수행할 때마다 $C=A+B$를 다시 계산하고, C의 각 자리 숫자 중 몇 개의 숫자가 바뀌었는지 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
5 3 09905 80000 B 1 5 A 2 9 B 5 9	2 4 2

노트

쿼리를 실행하기 전과 각 쿼리를 실행한 후의 계산식은 차례로 다음과 같다.

- $09905 + 80000 = 089905$
- $09905 + 80005 = 089910$
- $09995 + 80005 = 090000$
- $09995 + 90005 = 100000$

문제 B. Distance on Triangulation 2

시간 제한	3 초
메모리 제한	1024 MB

뜨룹랜드는 정점이 $2N$개인 볼록다각형 모양의 왕국이다.

볼록다각형의 각 정점에는 집이 하나씩 있다. 집들은 시계 방향으로 1부터 $2N$까지 번호가 매겨져 있다.

다각형의 둘레를 따라 $2N$개의 양방향 도로가 지어져 있다. 즉 모든 $i(1 \le i < 2N)$에 대해 i번 집과 $i+1$번 집이 도로로 연결되어 있고, 1번 집과 $2N$번 집이 도로로 연결되어 있다.

뜨룹랜드에는 1번부터 N번까지 번호가 매겨진 N명의 사람들이 살고 있다. 각 사람은 정확히 2개의 집을 소유하고 있다. 즉 주인이 없는 집은 없다. i번 사람이 소유한 두 집의 번호를 x_i, y_i라 하자.

뜨룹랜드에 다음 조건을 만족하도록 정확히 $2N-3$개의 양방향 도로를 더 지으려고 한다.

- 도로는 서로 다른 두 집을 선분으로 연결해야 한다.
- 이미 지어져 있는 도로를 포함하여, 같은 쌍을 잇는 도로는 2개 이상 있을 수 없다.
- 두 도로는 양쪽 끝이 아닌 지점에서 교차할 수 없다.

a번 집에서 b번 집으로 갈 때 거쳐야 하는 최소 도로 개수를 $Dist(a,b)$라 할 때, 위 조건을 만족하면서 $\sum_{i=1}^{N} Dist(x_i,y_i)$가 최소가 되도록 하는 도로 배치를 구하여라.

입력

첫째 줄에 $N(2 \le N \le 200\,000)$ 이 주어진다.

그 후 N개의 줄에 i번 사람이 소유한 두 집의 번호인 x_i, y_i가 공백을 사이에 두고 주어진다. $(1 \le x_i, y_i \le 2N)$

출력

첫째 줄에 $\sum_{i=1}^{N} Dist(x_i,y_i)$의 최솟값을 출력한다.

그 후 $2N-3$개의 줄에 새로 지을 도로가 잇는 두 집의 번호를 공백을 사이에 두고 출력한다.

가능한 배치가 여러 가지라면 아무것이나 출력해도 된다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
3 1 3 2 5 6 4	5 1 3 1 4 6 4

노트

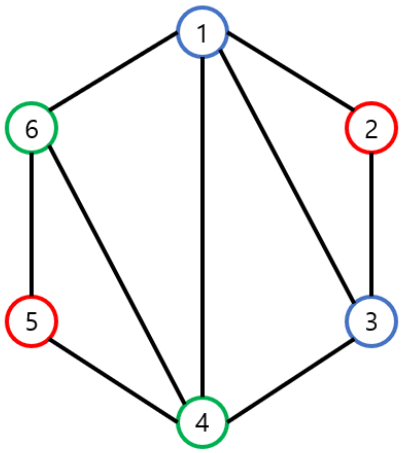


그림 B.1: 예제 1에 대해 정답으로 가능한 도로 배치 중 하나

$Dist(1,3) = 1, Dist(4,6) = 1, Dist(2,5) = 3$ 이므로 합은 5이다. 합을 4 이하로 만들 수 없으므로 해당 배치를 출력한다.

문제 C. UCP-Clustering

시간 제한	4 초
메모리 제한	1024 MB

군집화(clustering) 알고리즘은 같은 클러스터에는 최대한 비슷한 데이터가 묶이고, 서로 다른 클러스터 사이에는 다른 데이터들이 위치하도록 하는 알고리즘이다. UCP-Clustering은 군집화 알고리즘 중 하나로, 2차원 평면 위에 n개의 데이터가 주어질 때 다음과 같은 방식으로 데이터를 k개의 클러스터로 나눈다.

1. 각 클러스터는 구심점이 되는 중심 좌표를 하나씩 가진다. n개의 데이터 중 서로 다른 k개를 무작위로 선택하고, 각 데이터의 좌표를 중심 좌표로 갖는 k개의 빈 클러스터를 만든다.
2. 아래의 과정을 반복한다.
   (a) 각각의 데이터에 대하여 유클리드 거리가 가장 가까운 클러스터의 중심 좌표를 선택하고, 그에 해당하는 클러스터에 그 데이터를 포함시킨다. 가장 가까운 중심 좌표가 여러 개라면 그중에서 가장 작은 것을 선택한다. 좌표 (x_1, y_1)이 (x_2, y_2)보다 작다는 것은 $x_1 < x_2$거나, $x_1 = x_2$이고 $y_1 < y_2$인 경우를 의미한다.
   (b) 각각의 클러스터에 대하여 클러스터의 새로운 중심 좌표를 계산한다. 각 클러스터의 새로운 중심 좌표는 그 클러스터에 포함된 모든 데이터의 좌표들로부터 각 차원별로 중앙값을 계산하여 결정된다. 단, 값의 개수가 짝수일 경우, 중앙값은 값들을 정렬했을 때 가운데 두 값의 평균으로 계산된다.
   (c) 만약 모든 클러스터의 새로운 중심 좌표가 원래의 중심 좌표와 같다면 알고리즘을 종료한다. 그렇지 않다면 클러스터 구성을 초기화하고, 이 새로운 중심 좌표들을 중심 좌표로 갖는 k개의 빈 클러스터를 새로 만든다.

예를 들어, 3개의 데이터 $\{(1,2), (3,4), (5,6)\}$를 UCP-Clustering 알고리즘을 이용해 두 개의 클러스터로 나누는 상황을 생각해 보자. $(1,2)$와 $(3,4)$를 초기 중심 좌표로 선택하면 $\{(1,2)\}$와 $\{(3,4), (5,6)\}$ 두 개의 클러스터로 나뉘고, 두 클러스터의 중심 좌표는 각각 $(1,2)$, $(4,5)$가 된다. 다른 예시로 $(1,2)$와 $(5,6)$을 초기 중심 좌표로 선택하면 $\{(1,2), (3,4)\}$와 $\{(5,6)\}$으로 나뉘고, 두 클러스터의 중심 좌표는 각각 $(2,3)$, $(5,6)$이 된다. 예시에서 알 수 있듯이, 초기에 클러스터의 중심 좌표를 어떻게 선택하느냐에 따라 알고리즘의 실행 결과가 달라지며, 위의 두 경우에서는 2번 과정이 모두 2번 반복된다.

$k = 2$로 고정한다면 가능한 초기 중심 좌표의 경우의 수는 총 $n(n-1)/2$ 가지이다. 알고리즘을 완료했을 때 두 클러스터의 중심 좌표로 가능한 경우를 모두 찾고, 각 경우에 대해 수렴하는 데까지 걸리는 2번 과정의 반복 횟수의 기댓값을 구하시오.

입력

첫 번째 줄에 데이터의 수 N이 주어진다. $(2 \le N \le 512)$

그 후 N개의 줄에 i번째 데이터의 좌표인 (x_i, y_i)가 공백을 사이에 두고 주어진다. 모든 좌표는 정수이며, 모든 데이터의 좌표는 서로 다르다. $(-10^6 \le x_i, y_i \le 10^6)$

출력

두 클러스터의 최종 중심 좌표 $\mu_1 = (x_1^\mu, y_1^\mu)$, $\mu_2 = (x_2^\mu, y_2^\mu)$와 클러스터의 중심 좌표가 두 좌표로 수렴하는 데까지 걸리는 2번 과정의 반복 횟수의 기댓값을 공백을 사이에 두고 출력한다. 단, 두 좌표 중 더 작은 쪽을 μ_1으로 둔다. 가능한 좌표 쌍이 여러 개 존재하는 경우 μ_1이 작은 것부터, μ_1이 같은 경우는 μ_2가 작은 것부터 출력한다. 출력하는 값의 절대 오차 또는 상대 오차는 10^{-6}까지 허용한다.

주어지는 모든 데이터에서 처음에 클러스터의 중심 좌표가 되는 데이터를 어떻게 고르더라도 UCP-Clustering이 실패하는 상황, 즉 두 클러스터의 중심 좌표가 같아지거나, 한쪽 클러스터가 비거나, 클러스터의 중심 좌표가 수렴하지 않게 되는 상황이 발생하지 않음이 보장된다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
3 1 2 3 4 5 6	1.0 2.0 4.0 5.0 2.000000 2.0 3.0 5.0 6.0 2.000000
4 0 0 0 3 3 0 3 3	0.0 0.0 3.0 3.0 1.000000 0.0 1.5 3.0 1.5 2.000000 0.0 3.0 3.0 0.0 1.000000 1.5 0.0 1.5 3.0 2.000000

노트

첫 번째 예제에서, 알고리즘은 다음과 같이 동작한다.

- 초기 두 클러스터의 중심 좌표로 $(1,2)$, $(3,4)$를 선택한 경우: 첫 번째 반복을 통해 두 클러스터의 중심 좌표가 각각 $(1,2)$, $(4,5)$으로 바뀌고, 두 번째 반복에서 클러스터의 중심 좌표가 바뀌지 않아 알고리즘이 종료된다.
- 초기 두 클러스터의 중심 좌표로 $(1,2)$, $(5,6)$를 선택한 경우: 첫 번째 반복을 통해 두 클러스터의 중심 좌표가 각각 $(2,3)$, $(5,6)$으로 바뀌고, 두 번째 반복에서 클러스터의 중심 좌표가 바뀌지 않아 알고리즘이 종료된다.
- 초기 두 클러스터의 중심 좌표로 $(3,4)$, $(5,6)$를 선택한 경우: 첫 번째 반복을 통해 두 클러스터의 중심 좌표가 각각 $(2,3)$, $(5,6)$으로 바뀌고, 두 번째 반복에서 클러스터의 중심 좌표가 바뀌지 않아 알고리즘이 종료된다.

세 가지 경우를 모두 고려했을 때, 중심 좌표가 $\mu_1 = (1,2)$와 $\mu_2 = (4,5)$로 수렴하는 데에 걸리는 평균 반복 횟수는 2회이다. 마찬가지로, $\mu_1 = (2,3)$과 $\mu_2 = (5,6)$으로 수렴하는 데에 걸리는 평균 반복 횟수 또한 2회이다. 두 μ_1 중 $(1,2)$가 $(2,3)$보다 작으므로 $(1,2)$와 $(4,5)$로 수렴하는 경우부터 출력한다.

문제 D. 츠바메가에시

시간 제한	4 초
메모리 제한	1024 MB

"츠바메가에시"(つばめがえし)는 일본의 검사 "사사키 코지로"가 날아가는 제비를 베었다고 전해지는 검 초식의 이름이다. 기록상으로는 세 번 연속 칼질을 했다고 전해지나, 실제로 기술을 재연해 본 바에 따르면 두 번 연속까지가 한계라고 한다. 하지만 세 번 연속 베는 것이 더 멋있어 보이므로 게임이나 애니메이션 등의 미디어에서 이 기술을 묘사할 때는 검격을 세 번 하는 모습을 주로 볼 수 있다.

미디어에 영향을 많이 받은 pichulia는 세 번 연속 베는 기술을 재연하고자 한다. 피나는 노력 끝에 pichulia는 마침내 **X 축 또는 Y 축과 평행한 무한한 길이의 직선 형태의 검격**을 구사할 수 있게 되었다.

연습은 끝났고 이제는 실전이다. 하지만 pichulia는 야생생물 보호 및 관리에 관한 법률 제19조 1항에 의거해 야생의 제비를 벨 수 없었고, 부득이하게 제비뽑기의 제비를 베기로 결심했다.

2차원 평면상에 N개의 제비가 있고, 각 제비마다 베었을 때 얻을 수 있는 점수를 가지고 있다. **정확히 세 번의 검격을 통해** 얻을 수 있는 점수의 최댓값을 구해보자.

점수는 검격들에 베인 제비의 점수들의 합이다. 단, 한 제비를 여러 번 베어도 점수는 한 번만 더해진다.

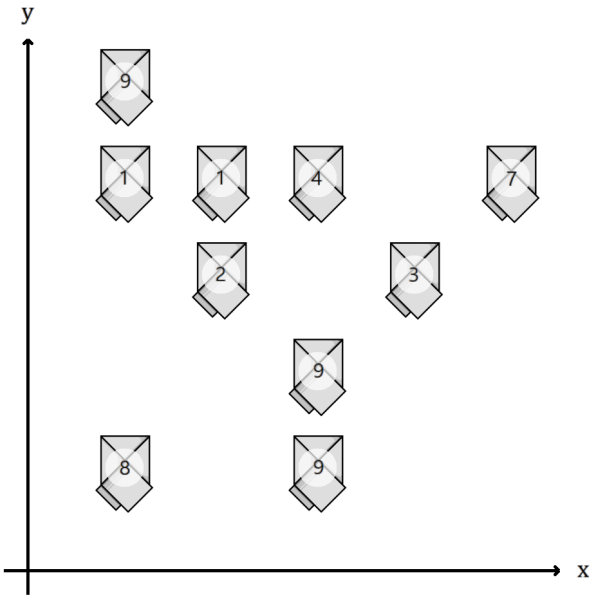


그림 D.1: 2차원 평면상에 배치된 제비

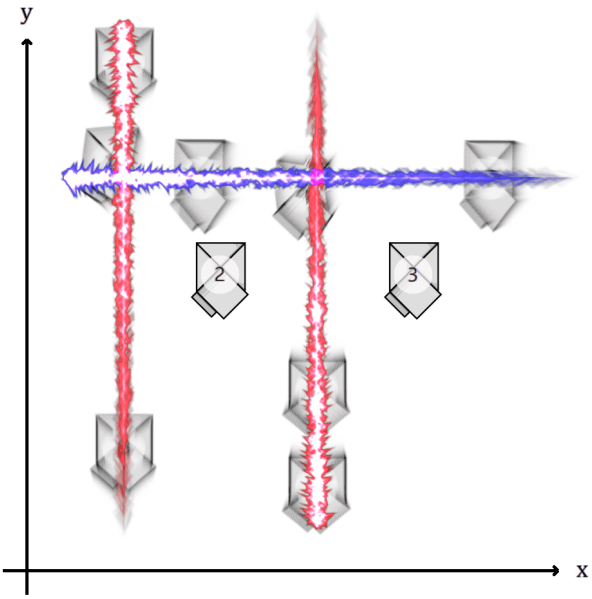


그림 D.2: 세 번의 검격으로 제비를 베는 모습

입력

첫 번째 줄에는 제비의 개수 $N(1 \le N \le 300\,000)$이 주어진다.

두 번째 줄부터 $N+1$번째 줄까지 N줄에 걸쳐서 세 정수 x, y, v가 공백으로 구분되어 주어진다. 이는 2차원 평면상의 좌표 (x, y) 에 점수가 v 인 제비가 있음을 의미한다. $(0 \le x, y \le 1\,000\,000,\ 1 \le v \le 7\,000)$

모든 제비의 위치는 서로 다르다.

출력

정확히 세 번의 검격을 통해 얻을 수 있는 점수의 최댓값을 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
10 1 1 8 1 4 1 1 5 9 2 3 2 2 4 1 3 1 9 3 2 9 3 4 4 4 3 3 5 4 7	48
8 1 0 1 1 1000000 1 2 1 1 2 999999 1 3 2 1 3 999998 1 4 3 1 4 999997 1	6
1 1 1 3	3

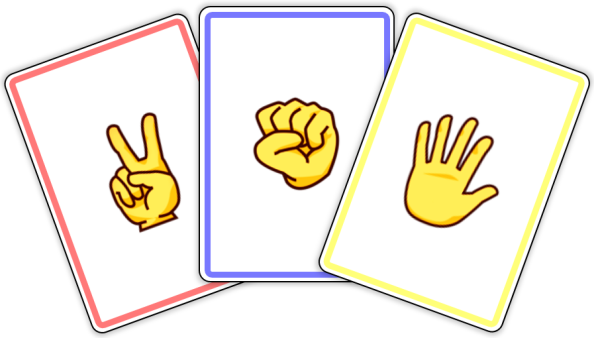
문제 E. 가위바위보 버블 정렬

시간 제한	1 초
메모리 제한	1024 MB

이환이는 4차 산업혁명 시대에 살고 있는 천재 5살 아기이다. 가위바위보를 즐기는 이환이는 집에서도 혼자 가위바위보를 하고 싶어서 창의력을 발휘해 카드를 이용한 게임을 만들었다. 게임을 하려면 먼저 '가위', '바위' 또는 '보'가 그려진 카드 N장을 일렬로 나열해 놓아야 한다. 그러고 나서, 가장 왼쪽에 있는 카드부터 차례대로 보면서 만약 그 카드가 바로 오른쪽에 있는 카드를 이긴다면 두 카드의 위치를 바꾸고, 그렇지 않다면 그대로 두는 것을 반복한다. 이렇게 마지막에서 두 번째 카드까지 한 번씩 훑어보면 놀이 한 번이 끝난다.

이환이는 이 놀이의 과정이 버블 정렬과 비슷하기 때문에 '가위바위보 버블 정렬'이라고 부르기로 했다. 카드의 승패 관계는 다음과 같다.

- '가위'가 그려진 카드는 '보'가 그려진 카드를 이긴다.
- 바위'가 그려진 카드는 '가위'가 그려진 카드를 이긴다.
- '보'가 그려진 카드는 '바위'가 그려진 카드를 이긴다.



계산이 아주 빠른 이환이는 이 놀이를 무려 T번이나 했다! 피곤해진 이환이는 내일 이 놀이를 계속하기로 하고 잠을 자러 가려고 했으나, 카드 위를 지나가다 그만 카드들이 뒤섞여 버렸다. 이환이는 뛰어난 기억력을 가진 천재이기 때문에 맨 처음 카드들의 배열을 기억해 냈지만, 마지막 카드 배열은 기억하지 못했다. 놀이를 T번 한 후의 배열을 복원해서 이환이를 도와 주자.

입력

첫 번째 줄에는 카드의 개수 $N(2 \le N \le 500\,000)$과 놀이를 한 횟수 $T(1 \le T \le 1\,000\,000\,000)$가 주어진다.

두 번째 줄에는 맨 처음 카드들의 배열을 나타내는 길이 N의 문자열이 주어진다. i번째 문자는 i번째 위치에 놓인 카드의 종류를 나타낸다. '가위'가 그려진 카드는 S, '바위'가 그려진 카드는 R, '보'가 그려진 카드는 P이다.

출력

T번 놀이를 한 후 카드들의 배열을 길이 N의 문자열로 출력한다. 마찬가지로 '가위'는 S, '바위'는 R, '보'는 P이다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
5 1 RSPSP	SRPPS
10 3 RSRRRRRRSR	SRRRRSRRRR

노트

첫 번째 예제에서, 이환이는 한 번의 놀이를 한다.

1. 첫 번째 카드 R(바위)은 두 번째 카드 S(가위)를 이기므로 두 카드를 서로 바꾼다. 이때 배열은 SRPSP가 된다.
2. 두 번째 카드 R(바위)은 세 번째 카드 P(보)에게 지므로 위치를 바꾸지 않는다.
3. 세 번째 카드 P(보)는 네 번째 카드 S(가위)에게 지므로 위치를 바꾸지 않는다.
4. 네 번째 카드 S(가위)는 마지막 카드 P(보)를 이기므로 두 카드를 서로 바꾼다. 이때 배열은 SRPPS가 된다.

따라서 놀이가 끝난 후 카드들의 배열은 SRPPS이다.

두 번째 예제에서, 이환이는 세 번의 놀이를 한다. 각 놀이가 끝난 후 카드들의 배열은 다음과 같이 변한다.
RSRRRRRRSR → SRRRRRRSRR → SRRRRRSRRR → SRRRRSRRRR
따라서 놀이가 끝난 후 카드들의 배열은 SRRRRSRRRR이다.

문제 F. 간단한 문제

시간 제한	4 초
메모리 제한	1024 MB

길이가 N인 두 수열 $(p_1, p_2, \ldots, p_N)$, $(q_1, q_2, \ldots, q_N)$ 이 주어진다.
이때 다음 값을 구하여라.

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\min(|p_i - p_j|, |q_i - q_j|)$$

입력

첫째 줄에 수열의 길이 $N(1 \le N \le 1\,000\,000)$ 이 주어진다.
둘째 줄에는 정수 $p_1, p_2, \ldots, p_N$ 이 공백으로 구분되어 주어진다. $(1 \le p_i \le 1\,000\,000)$
셋째 줄에는 정수 $q_1, q_2, \ldots, q_N$ 이 공백으로 구분되어 주어진다. $(1 \le q_i \le 1\,000\,000)$

출력

첫째 줄에 문제의 답을 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
3 1 3 2 1 2 3	6
4 1 1 1000000 1000000 1000000 1000000 1 1	7999992

문제 G. 돌 가져가기 2

시간 제한	2 초
메모리 제한	1024 MB

N 개의 돌이 일렬로 나열되어 있다. 각 돌은 흰색 또는 검은색의 색상을 갖는다.

돌들을 왼쪽부터 1번 돌, 2번 돌,..., N번 돌이라 하자. i번째 돌의 무게는 A_i이다.

당신은 나열된 돌들 중 하나를 골라 가져가는 것을 모든 돌이 없어질 때까지 총 N번 반복하려고 한다.

돌을 가져갈 때, 만약 그 돌이 현재 나열된 돌 중 가장 왼쪽이나 가장 오른쪽이 아니며, 가져간 돌에 인접한 두 돌 모두 가져간 돌과 다른 색인 경우 당신은 가져간 돌의 무게만큼의 점수를 얻는다.

돌을 가져가는 방법은 순서에 따라 총 $N!$가지 서로 다른 방법이 존재한다. 가능한 $N!$가지 방법에서 얻을 수 있는 점수의 총합을 구하여라.

입력

첫 줄에 양의 정수 N이 주어진다. $(1 \le N \le 2 \times 10^5)$

둘째 줄에 `B` 또는 `W`로만 이루어진 길이 N의 문자열 S가 주어진다.

S의 i번째 문자 S_i는 왼쪽에서 i번째 돌의 색을 나타낸다. `B`는 돌이 검은색임을, `W`는 돌이 흰색임을 뜻한다.

셋째 줄에 N개의 정수 $A_1, A_2, \cdots, A_N$이 주어진다. $(1 \le A_i \le 10^9)$

A_i는 i번 돌의 무게를 나타낸다.

출력

첫째 줄에 $N!$가지 방법에서 얻을 수 있는 점수의 총합을 998 244 353으로 나눈 나머지를 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
4 WBWB 6 4 5 3	72

노트

왼쪽에서 두 번째 돌을 가져가면서 점수를 얻는 방법과 세 번째 돌을 가져가면서 점수를 얻는 방법이 각각 8가지씩 있다. 따라서 총 점수는 $8 \times 4 + 8 \times 5 = 72$ 점이다.

문제 H. 봉화대

시간 제한	1 초
메모리 제한	1024 MB

N개의 마을이 일렬로 1번부터 N번까지 순서대로 산등성이를 따라 있다. 각 마을의 높이는 1 이상 N 이하의 자연수 중 하나이며 서로 다르다. 외침을 막기 위해 마을을 여러 구간으로 나누어, 각 구간의 가장 높은 마을에 봉화대를 설치하려 한다. 각 구간은 하나 이상의 연속된 마을을 포함해야 하고, 각 마을은 정확히 하나의 구간에 포함되어야 한다. 봉화대의 효율적인 상호 통신을 위해, 봉화대가 설치된 마을의 높이는 번호의 오름차순으로 봤을 때 증가해야 한다. 전략 도모를 위해 가능한 구간 배치의 개수를 파악해보고자 한다.

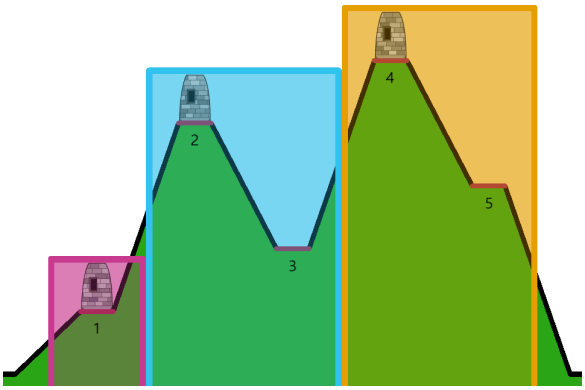


그림 H.1: 조건을 만족한 봉화대 설치 예시

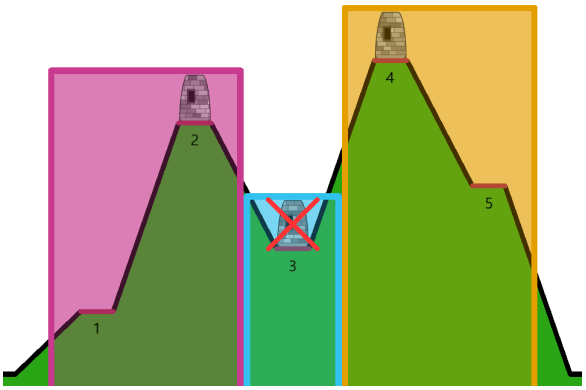


그림 H.2: 조건을 만족하지 않은 봉화대 설치 예시

입력

첫째 줄에는 마을의 개수를 의미하는 정수 N이 주어진다. $(1 \le N \le 500\,000)$

둘째 줄에는 각 마을의 높이를 의미하는 N개의 정수 $h_1, h_2, \cdots, h_N$이 공백으로 구분되어 주어진다. $(1 \le h_i \le N)$

h_i는 i번째 마을의 높이를 의미하며 서로 다르다.

출력

조건을 만족하며 N개의 마을을 구간으로 나누는 방법의 가짓수를 1 000 000 007로 나눈 나머지를 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
5 1 4 2 5 3	6
3 3 2 1	1
8 6 3 1 7 2 5 4 8	20

노트

첫 번째 예제의 가능한 모든 배치는 (1 / 4 / 2 5 3), (1 / 4 2 / 5 3), (1 / 4 2 5 3), (1 4 / 2 5 3), (1 4 2 / 5 3), (1 4 2 5 3) 이다. /는 구간의 경계이며, 봉화대는 밑줄로 강조된 높이의 마을에 설치된다.

문제 I. 붉은색 푸른색

시간 제한	2 초
메모리 제한	32 MB

메모리 제한에 유의하세요.

어린 시절 제연이는 1부터 N까지의 번호가 쓰여 있는 N개의 구슬을 가지고 있었다. 각 구슬은 붉은색 또는 푸른색을 띠고 있었으며, 제연이는 구슬들을 N개의 서로 다른 주머니에 하나씩 넣어 보관했다.

제연이는 구슬들을 소중히 여겼다. 그래서 매일매일 구슬들을 가지고 한 행동들을 모두 일기장에 순서대로 기록했다. 제연이가 일기장에 기록한 일은 다음 세 종류 중 하나이다.

- `1 i j` : i번 구슬이 들어 있는 주머니와 j번 구슬이 들어 있는 주머니를 합친다. $(1 \le i, j \le N)$
- `2 i` : i번 구슬을 갖고 놀다가 잃어버린다. $(1 \le i \le N)$ 슬프게도 잃어버린 구슬은 영영 다시 찾지 못했다.
- `3 i l h` : i번 구슬이 들어 있는 주머니를 관찰한다. 그 주머니에는 붉은색 구슬이 l개 이상 h개 이하 들어 있었다. $(1 \le i \le N, 0 \le l \le h \le N)$

10년이 지난 뒤, 제연이는 M개의 기록이 적혀 있는 일기장을 발견했다. 그 시절의 순수함을 잃어버린 제연이는 일기장을 보고 각 구슬의 색깔을 복원하기로 했다. 혼자서 색깔을 복원할 생각을 하니 눈앞이 샛노래진 제연이를 도와주자.

단, 구슬은 항상 주머니에 넣어서 보관했으며, 일기장에 기록되지 않은 구슬의 이동이나 변화는 없었다고 가정한다.

입력

첫째 줄에 정수 N과 $M(2 \le N \le 2\,000, 1 \le M \le 4\,000)$이 주어진다.

둘째 줄부터 M개의 줄에 일기장의 기록이 위에서 설명한 형식으로 주어진다. 이전에 잃어버리지 않은 구슬만 기록에 등장하며, 1번 종류의 기록에서 i번 구슬과 j번 구슬은 다른 주머니에 속해 있는 상태였음이 보장된다.

출력

M개의 기록을 모두 만족하는 N개 구슬의 색깔 조합이 존재한다면 `YES`를, 존재하지 않는다면 `NO`를 출력한다.

그러한 색깔 조합이 존재한다면, 다음 줄에 길이 N의 문자열을 출력한다. 문자열의 i번째 문자는 i번 구슬이 붉은색이면 `R`, 푸른색이면 `B`이다. 답이 여러 가지라면 아무것이나 출력해도 된다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
5 9 1 2 4 1 1 5 3 4 1 2 3 5 0 1 1 4 1 3 4 2 5 2 1 3 4 0 1 3 3 1 1	YES RBRRB
3 4 1 1 2 3 2 1 2 1 2 3 3 3 0 0	NO

문제 J. 시험 문제 출제

시간 제한	1 초
메모리 제한	1024 MB

덕인 교수는 알고리즘 과목을 강의하는 교수다. 덕인 교수는 다가오는 중간고사 시험 문제를 준비하면서, 얼마 전 학생들에게 가르친 최대 부분합 문제를 추가하기로 했다. 최대 부분합 문제는 다음과 같다.

- 배열 $A = [a_1, a_2, \ldots, a_n]$가 주어질 때, $max_{1 \le i \le j \le n}(\sum_{k=i}^{j} a_k)$를 구하여라.

덕인 교수는 수열 하나를 주고, 해당 수열에서 최대 부분합을 구하도록 할 심산이다. 덕인 교수는 수열 제작의 대가로, 다음과 같은 과정으로 수열을 만든다고 알려져 있다.

- 정수만으로 구성된 $N \times N$ 크기의 이차원 배열을 준비한다.
- $(1,1)$에서 출발해 아래 혹은 오른쪽으로 이동하는 것을 반복하여 (N,N)에 도착한다. 참고로 (i,j)에서 아래로 이동하면 $(i+1,j)$에 도달하고, 오른쪽으로 이동하면 $(i,j+1)$에 도달한다.
- 지나온 칸에 적힌 정수를 지나온 순서대로 나열하여 길이 $2N-1$의 수열을 완성한다.

덕인 교수는 '답이 K인 문제는 아름다운 문제'라는 신념을 가지고 있어서, 최대 부분합이 K인 수열을 만들려고 한다. 여기에 더해서, 최대 부분합이 K인 수열을 최대한 많은 방법으로 만들려고 한다. 덕인 교수의 조교인 당신은 $N \times N$ 크기의 이차원 배열이 주어지면, 최대 부분합이 K인 수열을 만드는 서로 다른 방법의 수를 구해야 한다. 어떤 두 방법이 서로 다르다는 것은 두 수열을 구성하는 이차원 배열상의 경로가 서로 다르다는 것을 의미한다.

입력

첫 번째 줄에는 이차원 배열의 크기 N과 만들고자 하는 최대 부분합 K가 주어진다. $(1 \le N \le 20, -4 \times 10^{10} \le K \le 4 \times 10^{10})$

그 다음 N개의 줄에 걸쳐 각 줄에 N개의 정수가 공백으로 구분되어 주어진다. 구체적으로, i번째 줄의 j번째 정수는 이차원 배열 A의 (i,j)에 위치하는 $A_{i,j}$를 의미한다. $(-10^9 \le A_{i,j} \le 10^9)$

출력

주어진 배열에서 최대 부분합이 K인 수열을 만드는 서로 다른 방법의 수를 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
3 3 1 2 -5 -2 3 0 -1 -1 1	2
4 3 1 -1 1 1 1 1 1 1 1 1 1 1 1 1 -1 1	4

노트

첫 번째 예제에서 만들 수 있는 수열들은 다음과 같다.

- $[1, 2, -5, 0, 1]$
- $[1, 2, 3, 0, 1]$
- $[1, 2, 3, -1, 1]$
- $[1, -2, 3, 0, 1]$
- $[1, -2, 3, -1, 1]$
- $[1, -2, -1, -1, 1]$

이 중에서 최대 부분합이 3인 수열은 1번째와 5번째 수열뿐이므로 답으로 2를 출력한다.

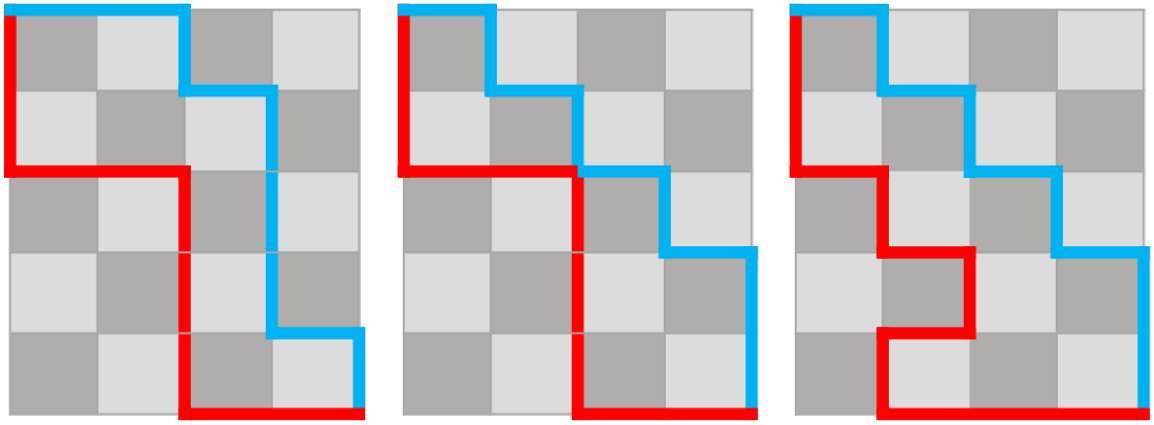
두 번째 예제에서는 만들 수 있는 수열들 중 $[1, -1, 1, 1, 1, -1, 1]$만이 최대 부분합이 3이고, 이러한 수열을 만들기 위해서는 $(1, 2)$와 $(4, 3)$을 반드시 지나야 하므로 경우의 수는 4가 된다.

문제 K. 은퇴한 자들의 게임

시간 제한	1 초
메모리 제한	1024 MB

PS계를 성공적으로 은퇴한 제연이와 덕인이는 허전한 마음을 달래기 위해 새로운 보드게임을 개발했다. 이 게임은 K개의 판과 K개의 말로 구성된다. $i(i=1,\cdots,K)$번째 판은 세로 길이 N_i, 가로 길이 M_i 크기의 격자판인데, 게임을 시작하기 전 울타리를 사용해 경계를 만든다.

울타리는 격자의 **변**을 따라 짓는다. 격자판의 왼쪽 위 꼭짓점에서부터 오른쪽(R) 또는 아래(D)로만 길이 1씩 움직여서 오른쪽 아래 꼭짓점에 도달하는 길이 N_i+M_i의 경로들을 고려하자. 이 중 시작점과 끝점을 제외하고는 만나지 않는 **두 경로**를 골라 이들을 따라 울타리를 지었을 때 이를 **좋은 울타리 배치**라고 한다.



(a) (b) (c)

그림 K.1: 좋은 울타리 배치와 나쁜 울타리 배치

예를 들어 $N_i=5, M_i=4$인 경우를 살펴보자. (a)의 경우 두 경로 RRDRDDDRD, DDRRDDDRR이 시작점과 끝점을 제외하고는 만나지 않으므로 좋은 울타리 배치이다. 한편 (b)의 경우 선택된 두 경로 RDRDRDRDD, DDRRDDDRR이 시작점도 끝점도 아닌 곳에서 만나므로 좋은 울타리 배치가 아니다. (c)의 경우 두 경로 중 하나가 오른쪽 또는 아래로만 움직이는 경로가 아니므로 마찬가지로 좋은 울타리 배치가 아니다.

게임은 K개의 격자판에 각각 좋은 울타리 배치로 경계를 만든 뒤, 각 격자판의 가장 왼쪽 위 **칸**에 말을 하나씩 놓은 상태로 시작된다. 게임은 제연이가 먼저 시작해 한 턴씩 번갈아 가며 진행된다. 제연이는 자신의 차례에 K개의 말 중 하나를 골라 **오른쪽으로 한 칸** 이동시켜야 하고, 덕인이는 자신의 차례에 K개의 말 중 하나를 골라 **아래로 한 칸** 이동시켜야 한다. 울타리를 넘어서 말을 이동시킬 수는 없으며, 자신의 차례에 움직일 수 있는 말이 없는 사람이 게임에서 지게 된다.

안타깝게도 이 게임은 행운 요소가 하나도 없는 실력겜이라 두 사람이 최선을 다한다면 게임을 하기 전에도 이길 사람을 알 수 있다. 둘은 더 이상 PS 문제를 풀 의지가 남아있지 않기 때문에 여러분이 이길 사람을 알려줘야 한다.

입력

첫째 줄에 정수 $K(1 \le K)$가 주어진다.

두번째 줄부터 $i=1,\cdots,K$ 번째 격자판의 정보가 각각 세 줄씩 주어진다. 첫째 줄에 정수 N_i, M_i가 주어지고, 이후 두 개의 줄에는 울타리를 짓는 두 경로가 각각 길이 N_i+M_i의 문자열로 주어진다. 각 문자열은 R 또는 D로만 이루어져 있으며, 좋은 울타리 배치를 표현한다. $(1 \le N_i,\ 1 \le M_i,\ \sum_{i=1}^{K}(N_i+M_i) \le 500\,000)$

출력

두 사람이 최선을 다해서 게임을 진행할 때, 제연이가 이기면 `First`, 덕인이가 이기면 `Second`를 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
2 3 2 RRDDD DRDDR 1 2 DRR RRD	First
2 2 2 DRDR RRDD 4 5 RDRDDRRRD DDDRDRRRR	Second

노트

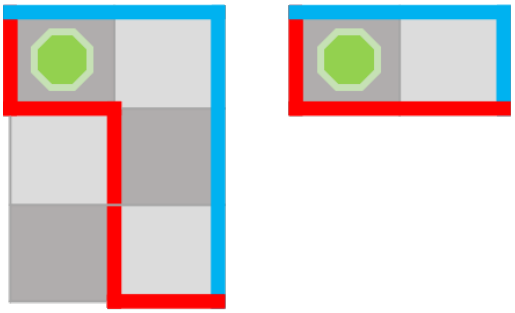


그림 K.2: 예제 1 참고

예제 1의 초기 게임판은 위와 같다. 제연이가 첫 번째 턴에 두 번째 판의 말을 오른쪽으로 움직이면 덕인이는 더 이상 말을 움직일 수 없으므로 제연이가 이긴다.

문제 L. Make Different

시간 제한	3 초
메모리 제한	1024 MB

원형 게임판에 N개의 스프링이 있다. 스프링은 시계 방향으로 1번 부터 N번 까지 번호가 붙어 있다. 각 스프링은 빨간색 또는 파란색으로 칠해져 있는데, 빨간 스프링을 밟으면 바로 양옆에 있는 스프링으로 점프할 수 있으며 파란 스프링을 밟으면 양옆으로 두 칸 거리에 있는 스프링으로 점프할 수 있다.

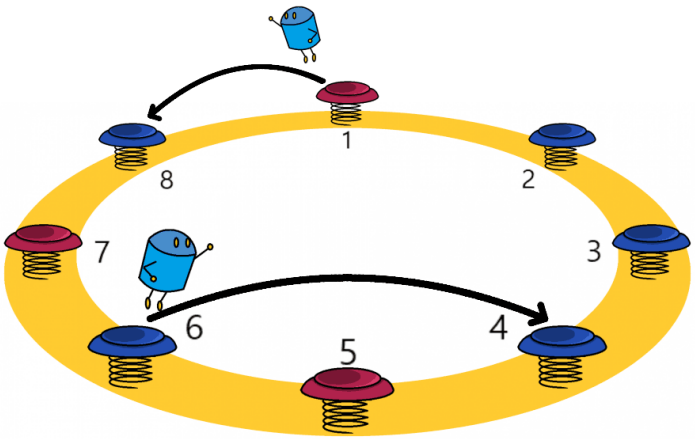


그림 L.1: 두 로봇이 1번과 6번 스프링에 있고, 반시계 방향 점프 명령을 한 경우 로봇의 움직임

당신은 두 로봇을 이용하여 게임을 할 것이다. 우선 게임판에 두 로봇을 서로 다른 번호의 스프링 위에 놓는다. 당신은 로봇들에게 시계 방향 또는 반시계 방향으로 움직이라고 명령할 수 있다. 명령을 내리면 두 로봇이 **동시에** 당신이 명령한 방향으로 점프한다. 즉 로봇을 하나만 움직이거나 두 로봇이 다른 방향으로 점프하도록 할 수는 없다.

게임은 두 로봇이 서로 다른 색깔 스프링을 밟으면 끝난다. 로봇의 시작 위치가 주어지면 게임을 끝내기 위해 필요한 최소 명령 수를 구하여라.

입력

첫 번째 줄에 스프링의 수 $N(3 \le N \le 100\,000)$과 질문의 수 $Q(1 \le Q \le 100\,000)$가 주어진다.

두 번째 줄에 스프링의 종류가 1번 스프링부터 N번 스프링까지 순서대로 주어진다. 빨간 스프링은 1, 파란 스프링은 2이다.

세 번째 줄부터 Q개의 줄에는 두 로봇이 게임을 시작할 스프링의 번호 $R1$, $R2$가 주어진다. $(1 \le R1, R2 \le N, R1 \ne R2)$

출력

Q개의 줄에 걸쳐 두 로봇이 다른 색깔 스프링에 위치하기 위해 필요한 최소 명령 수를 출력한다. 처음부터 두 로봇이 다른 색깔 스프링에 있었으면 `0`을, 아무리 명령하더라도 두 로봇이 다른 색깔 스프링에 있게 할 수 없다면 `-1`을 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
8 3 1 2 2 2 1 2 1 2 1 2 1 5 3 6	0 -1 1