2020
서울대학교 프로그래밍 경시대회

Division 1

주최 및 주관

후원

NAVER

서울대학교 컴퓨터공학부
Seoul National University
Dept. of Computer Science and Engineering

STARTLINK

2020년 9월 12일

참가자를 위한 도움말

- 대회 시간은 13:00부터 17:00까지입니다. 대회가 진행되는 동안 인터넷 검색 및 라이브러리 코드 사용은 **전면 허용됩니다.** 하지만 다른 참가자나 외부인과 풀이, 코드를 공유할 수 없습니다. 또한, 대회 시간 동안 이 문제지를 외부로 유출하거나 공유할 수 없습니다.
- 대회는 Baekjoon Online Judge(https://www.acmicpc.net/) 플랫폼을 이용하여 진행됩니다. 별도로 제공되는 계정 정보를 이용하여 로그인하신 뒤 코드 제출 및 결과 확인 등을 하실 수 있습니다.
- 제출하실 답안 코드는 C11, C++14, C++17, Java, Java (OpenJDK), Python 3, PyPy3 **로만** 작성하셔야 합니다. 단, C++14와 C++17을 제외한 언어의 경우 시간 제한 및 메모리 제한 안에 문제를 풀 수 있음이 보장되지 않습니다.
- 모든 입력은 **표준 입력**으로 주어지고, 모든 출력은 **표준 출력**으로 합니다.
- 프로세스가 0이 아닌 값을 리턴하거나, 표준 오류 스트림 (`stderr`)에 값을 출력할 경우 런타임 에러를 받습니다.
- 문제에 대한 질의 사항은 대회 페이지의 질문 기능을 사용해 주시기 바랍니다. 이 때 대답해 드리기 어려운 질문에 대해서는 "답변을 드릴 수 없습니다"로 답변할 수 있습니다.
- 이 문제지는 참고용입니다. 문제지에 적힌 문제와 플랫폼에 올라온 문제가 다를 경우, 플랫폼의 문제가 우선합니다.

채점 결과

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

틀렸습니다 제출하신 답안 프로그램이 테스트 데이터에 대해 생성한 출력이 출제자의 정답과 일치하지 않음을 의미합니다.

컴파일 에러 제출하신 답안 프로그램을 컴파일하는 도중 오류가 발생하였음을 의미합니다. 제출 기록을 클릭하여 오류가 발생한 부분을 볼 수 있습니다.

런타임 에러 제출하신 답안 프로그램을 실행하는 도중 프로세스가 비정상적으로 종료되었음을 의미합니다.

시간 초과 제출하신 답안 프로그램이 정해진 시간 안에 종료되지 않았음을 의미합니다.

출력 형식이 잘못되었습니다 답을 올바르게 구했으나, 답안 프로그램의 출력 형식이 문제에 나와 있는 출력 형식과 다름을 의미합니다. 대표적으로 줄 뒤에 의미 없는 공백을 두 칸 이상 출력하거나 의미 없는 빈 줄을 출력할 경우 이 결과를 받을 수 있습니다.

출력 초과 답안 프로그램이 정답에 비해 너무 많은 출력을 했음을 의미합니다. 이 결과는 **틀렸습니다**와 같은 의미를 갖습니다.

만약 여러 원인으로 인해 **맞았습니다!!**가 아닌 다른 결과를 얻으셨다면, 그 중 어떤 것도 결과가 될 수 있습니다. 예를 들어 답도 잘못되었고 비정상적인 동작도 수행하는 코드를 제출하신 경우 대부분 **런타임 에러**를 받으시게 되지만, 경우에 따라서 **틀렸습니다**를 받을 수도 있습니다.

문제 정보

문제 번호	제목	시간 제한	메모리 제한
A	빈 문자열 만들기	1.0s	1024MB
B	공정한 회의	2.0s	
C	전국일주	1.0s	
D	버거운 버거	3.0s	
E	직선형 분자 만들기	5.0s	
F	$2 \times M$ 타일링	1.0s	
G	3분 그래프 리턴즈	3.0s	
H	계주 코스 정하기	2.0s	
I	연금술사	1.0s	

문제지와 플랫폼의 제한 조건이 다를 경우 플랫폼을 우선으로 참고하세요.

Problem A. 빈 문자열 만들기

0과 1로만 이루어져 있으며, 0의 개수와 1의 개수가 동일한 문자열 S가 주어진다. 당신은 S에 다음과 같은 작업을 여러 번 수행할 수 있다:

S의 길이 $2k$인 연속한 부분문자열이 앞 k개 문자가 모두 동일하고, 또한 뒤 k개 문자가 서로 동일하며, 0과 1을 모두 포함할 때, 그 부분문자열을 제거할 수 있다.

예를 들어, $S =$ "0111000011"인 경우, S의 2번째 문자부터 7번째 문자까지인 "111000"을 제거하는 것이 가능하다. 이 작업 후에는 제거된 부분의 앞부분과 뒷부분이 연결되어 $S =$ "0011"이 된다. 그러면 이제 한 번의 작업을 통해 "0011"을 제거할 수 있으므로 초기 $S =$ "0111000011"는 두 번의 작업을 통해 빈 문자열로 만들 수 있다.

여러분의 목표는 최소 횟수의 작업을 통해 S를 빈 문자열로 만드는 것이다. 최소 횟수의 작업으로 S를 빈 문자열로 만드는 과정을 구하여라.

입력 형식

첫 줄에 0과 1로만 이루어진 문자열 S가 주어진다. S에 포함된 0의 개수와 1의 개수는 동일하다.

출력 형식

작업들을 통해 S를 빈 문자열로 만드는 것이 불가능하다면 첫 줄에 `-1`을 출력한다.

그렇지 않은 경우, 첫 줄에 필요한 작업의 최소 횟수 K를 출력한다. 그 다음 K줄에 걸쳐 수행한 작업에 대한 정보를 출력한다.

K개 줄 중 i번째 줄에는 두 정수 b_i와 e_i를 공백을 사이에 두고 출력한다. 이는 i번째 작업에서 S의 b_i번째 문자부터 e_i번째 문자로 이루어진 문자열을 제거하였음을 뜻한다.

최소 횟수의 작업으로 문자열을 지우는 방법이 여러 가지인 경우에는 그 중 아무 것이나 출력해도 된다.

제약 조건

- S에 포함된 문자의 개수는 1 이상 $500,000$ 이하이다.
- S에 포함된 0의 개수와 1의 개수는 동일하다.

예제

표준 입력(stdin)	표준 출력(stdout)
0101110100	4 2 3 4 5 3 6 1 2

예제 설명

처음에 $S =$ "0101110100"인 상태이다.

S의 i번째 문자부터 j번째 문자까지의 문자열을 $S[i : j]$라 하자.

$S[2 : 3] =$ "10"을 제거하면 $S =$ "01110100"이 된다.

그 후 $S[4 : 5] =$ "10"을 제거하면 $S =$ "011100"이 되고,

다음에 $S[3 : 6] =$ "1100"을 제거하면 $S =$ "01"이 된다.

마지막으로 $S[1 : 2] =$ "01"을 제거하면 S는 빈 문자열이 된다.

$S =$ "0101110100"을 4번 미만의 작업을 통해 빈 문자열로 만드는 방법은 존재하지 않는다.

Problem B. 공정한 회의

먼 미래, SNUPS는 알고리즘 교육 열풍에 힘입어 총 N명의 회원이 존재하는 큰 동아리가 되었다. 모든 서로 다른 두 회원 사이에는 친밀한 정도를 나타내는 "친밀 거리"를 정의할 수 있다. i번 회원과 j번 회원 사이의 친밀 거리는 정수 $C(i,j)$로 나타내며, 값이 작을수록 더 친밀함을 의미한다. 친밀 거리 $C(i,j)$는 모든 $1 \le i,j \le N, i \ne j$에 대해 $1 \le C(i,j) \le 10^7$, $C(i,j) = C(j,i)$를 만족한다.

SNUPS는 모든 회원의 의견을 모두 반영하면서도 효율적으로 동아리를 운영하기 위해, 안건이 발생할 때마다 N명의 회원 중 랜덤하게 뽑힌 세 명만이 회의를 진행하는 시스템을 도입했다. 하지만 이 시스템 하에서는 뽑힌 세 명 중 친한 두 명이 연합하면 나머지 한 명의 의견이 제대로 반영되지 않는 문제가 발생했다. 구체적으로, 회의를 위해 뽑힌 서로 다른 세 회원이 각각 i,j,k번 회원이라고 할 때 $C(i,j) < C(j,k)$이고 $C(i,j) < C(i,k)$이면 k번 회원의 의견이 반영되지 못하고, 이러한 경우 회의가 **불공정**하다고 한다.

제연이는 어떤 세 명이 회의를 진행하더라도 불공정한 경우가 생기지 않게끔 하고 싶다. 다행히도 SNUPS는 커질 대로 커져, M쌍의 회원을 제외하고는 아직 서로 모르는 상태이다. 이미 서로 아는 회원들 사이의 친밀 거리를 바꿀 수는 없지만, 서로 모르는 회원들 간의 친밀 거리를 원하는 대로 조정하는 것은 인간 심리에 통달한 제연이에게는 무척 간단한 일이다. 물론 최종적으로 정해진 친밀 거리는 $1 \le i,j \le N, i \ne j$에 대해 $1 \le C(i,j) \le 10^7$, $C(i,j) = C(j,i)$를 만족하는 정수여야 한다.

만약 이러한 계획이 실현 가능하다면, 친밀한 SNUPS를 위해 $\sum_{i=1}^{N}\sum_{j=i+1}^{N} C(i,j)$를 최소로 만들려고 한다. SNUPS의 밝은 미래를 위해 제연이의 계획이 실현 가능한지 판단하고, 가능하다면 $\sum_{i=1}^{N}\sum_{j=i+1}^{N} C(i,j)$의 최솟값을 구하자.

입력 형식

입력의 첫 줄에는 회원의 수 N, 이미 서로를 알고 있는 관계의 수 M이 주어진다.

둘째 줄부터 M개의 줄에 걸쳐 회원들의 관계에 대한 정보가 주어진다. $i+1$번째 줄에는 A_i, B_i, D_i가 공백을 사이에 두고 주어지며, A_i번 회원과 B_i번 회원이 이미 알고 있는 관계이며 이들의 친밀 거리는 D_i임을 의미한다.

출력 형식

첫째 줄에 제연이의 계획이 가능하지 않다면 `-1`을, 가능하다면 $\sum_{i=1}^{N}\sum_{j=i+1}^{N} C(i,j)$의 최솟값을 출력한다.

제약 조건

- $3 \le N \le 300,000$
- $0 \le M \le 300,000$
- $1 \le A_i, B_i \le N$, $A_i \ne B_i$, $i \ne j$에 대해 $\{A_i, B_i\} \ne \{A_j, B_j\}$
- $1 \le D_i \le 10^7$

예제

표준 입력(stdin)	표준 출력(stdout)
4 2 1 2 5 2 4 3	14
4 4 1 2 10 1 3 20 2 4 30 3 4 40	-1

Problem C. 전국일주

승현이는 오랜만에 휴가를 얻어 오랜 꿈이었던 전국일주를 할 계획을 세웠다.

승현이가 살고 있는 나라는 1번부터 N번까지의 N개 마을과 서로 다른 두 마을간을 잇는 $N(N-1)/2$개의 도로로 이루어져 있다. 승현이는 각 마을을 정확히 한 번씩 지나 여행을 시작한 마을로 돌아오려고 한다. 문제는 도로 종류가 자갈 도로, 진흙 도로의 두 가지라는 점이다.

자갈 도로와 진흙 도로를 지나갈 수 있는 타이어는 서로 다르다. 따라서 승현이는 지나는 도로의 종류가 바뀔 때마다 타이어를 갈아 주어야 한다. 승현이는 타이어를 딱 한 번 바꿀 수 있을 만큼 여유가 있기 때문에, 여행을 하는 도중 도로 종류가 최대 한 번만 바뀌는 여행 경로를 찾고자 한다.

승현이는 현재 마을들을 잇는 도로가 어떤 종류인지 전혀 모르고 있다. 승현이는 디지털 문명과는 담을 쌓고 사는 사람이기 때문에, 친구인 제연이에게 도로의 종류에 대해 물어보고자 한다. 승현이는 한 번의 질문으로 두 마을을 잇는 도로의 종류를 알아낼 수 있다. 하지만 $2N$번의 질문 후에는 제연이의 인내심이 바닥나 더 이상 질문을 할 수 없다.

다행히도, 각 도로의 종류를 어떻게 정하든 간에 $2^{N(N-1)/2}$가지의 모든 경우에 대해서 승현이의 조건을 만족하는 여행 경로가 존재함이 보장된다고 한다. 하지만 승현이는 어떤 질문을 해야 할지 결정하지 못해 당신에게 도움을 청했다. 승현이가 무사히 전국일주를 할 수 있도록 효율적으로 질문을 던져 조건을 만족하는 여행경로를 찾아 주자.

중요: 채점 시스템은 맨 처음에 각 도로의 종류를 정해놓지 않고, 여러분의 질문에 따라 도로의 종류를 결정할 수 있다. 단, 이전까지의 답변들과 모순되는 답변은 하지 않는다.

상호 작용(Interaction)

입력의 첫 줄에 마을의 수 N이 주어진다.

그 뒤로는 여러분의 프로그램과 채점 시스템이 interaction하면서 진행된다.

i번 마을과 j번 마을을 잇는 도로의 종류는 `? i j`을 표준 출력 스트림(stdout)으로 출력하여 물어볼 수 있다. ($1 \le i, j \le N, i \ne j$) 출력한 후에는 표준 출력 버퍼를 flush해 주어야 한다. 언어별로 표준 출력 버퍼를 flush하는 방법은 다음과 같다.

- C: `fflush(stdout)`
- C++: `std::cout << std::flush`
- Java: `System.out.flush()`
- Python: `sys.stdout.flush()`

각각의 질문 이후에는 채점 시스템이 질문에 대한 답을 줄 것이다. `Gravel`은 자갈 도로, `Mud`는 진흙 도로를 뜻한다. 이러한 질문은 최대 $2N$번까지만 할 수 있다.

조건을 만족하는 경로를 찾은 경우, 순서대로 a_1번 마을, a_2번 마을, $\cdots$, a_n번 마을을 차례로 지나 다시 a_1번 마을로 돌아오는 경로가 조건을 만족한다고 하면 `!` a_1 a_2 $\cdots$ a_n의 형태로 답을 출력하고 프로그램을 종료해야 한다.

채점 프로그램은 현재까지 알려진 정보와 모순되지 않는 모든 경우의 도로망에서 참가자가 제시한 경로가 조건을 만족할 때 이를 정답으로 처리한다. 현재까지 알려진 정보와 모순되지 않으면서 참가자가 제시한 경로가 두 번 이상 도로 종류가 바뀌도록 도로들의 종류를 설정할 수 있다면 채점 프로그램은 이를 오답으로 처리한다.

제약 조건

- $3 \leq N \leq 2,000$

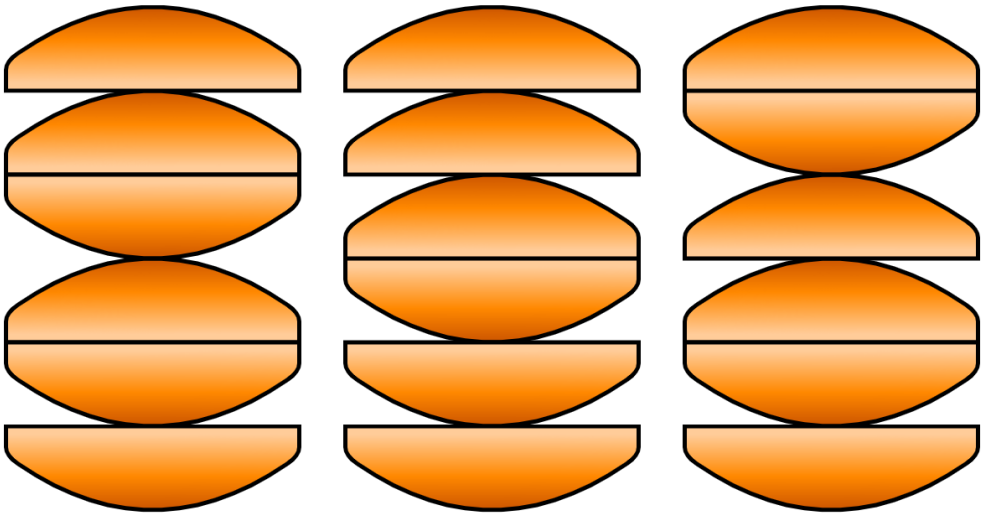
예제

입출력 예제는 채점 프로그램과의 상호 작용 방식을 보여주기 위한 것으로, 실제 채점 프로그램과의 작동 방식과는 다를 수 있다.

표준 입력(stdin)	**표준 출력**(stdout)
4 Mud Gravel Gravel	 ? 1 2 ? 2 3 ? 3 4 ! 2 3 4 1

Problem D. 버거운 버거

드디어 산업기능요원 복무를 마친 키파는 *버거*운 직장에서 벗어나 새로운 직업에 도전하고자 햄*버거*집을 차렸다. 키파는 케이크를 여러 차례 만들면서 빵은 좀 구워 봤지만 햄버거를 만드는 것은 처음이었기 때문에, 위아래의 구분이 있는 빵을 적당히 구워서 햄버거 패티로 햄버거가 들어간 탄수화물 폭탄의 **버거운 버거**를 만들어 팔기로 했다.



버거운 버거의 예시.

버거운 버거의 엄밀한 정의는 다음과 같다.

- 속이 위로 온 빵 X 위에 속이 아래로 온 빵 Y를 올린 것은 버거운 버거이다. 이때 X를 Y의 **대응하는 쌍**, Y를 X의 **대응하는 쌍**이라 정의하자.
- 속이 위로 온 빵 X 위에 버거운 버거를 올리고, 그 위에 속이 아래로 온 빵 Y를 올린 것은 버거운 버거이다. 마찬가지로 이때 X를 Y의 **대응하는 쌍**, Y를 X의 **대응하는 쌍**이라 정의하자.
- 버거운 버거 위에 버거운 버거를 올린 것은 버거운 버거이다.
- 위 세 규칙으로 만들 수 없는 것은 버거운 버거가 아니다.

키파는 총 N개의 빵 굽는 기계를 일렬로 세워 두고 동시에 다루고 있다. 하나의 기계는 빵을 하나만 구울 수 있다. 가장 왼쪽의 것부터 오른쪽으로 순서대로 1부터 번호를 붙이자. 키파가 가게를 운영하는 도중에 다음 둘 중 하나의 상황이 Q번 발생할 수 있다.

- a번 기계부터 b번 기계까지의 빵이 곧 타려고 하기 때문에, 각각 뒤집어 줘야 한다.
- 손님이 a번 기계부터 b번 기계까지의 빵을 차곡차곡 쌓아 주기를 원한다. 즉, 이 과정 중 **빵을 뒤집어 쌓으면 안 되고**, 가장 아래에 a번 기계에서 나온 빵, 그 위에 $(a+1)$번 기계에서 나온 빵, 이렇게 하여 가장 위에 b번 기계에서 나온 빵이 **순서대로** 쌓여야 한다. 키파는 기계에 빵이 없으면 재료가 다 떨어진 것처럼 보인다고 생각했기 때문에, 한 주문이 끝난 이후 그 주문을 받기 이전의 상태대로 빵의 위아래를 맞춰서 채워 둔다.

그러나 손님들이 햄버거를 만들기를 원하는 빵을 쌓았을 때, 어떤 빵은 대응하는 쌍이 존재하지 않아 버거운 버거로서 실격일 수 있다. 키파는 각 손님의 주문대로 빵을 쌓은 뒤, 이 빵을 버거운 버거로 만들기 위해 쌓아둔 빵의 순서를 유지하면서 미리 구워 둔 빵을 최소한으로 집어넣고자 한다. 키파는 손님들의 건강에 관심이 많기 때문에 각 주문마다 버거운 버거의 높이, 즉 버거운 버거에 들어간 빵의 개수를 알고자 한다. 이를 구하는 프로그램을 작성해서 키파를 도와 주자.

입력 형식

첫 줄에 양의 정수 N이 주어진다.

둘째 줄에 길이가 N이고 (와)로만 구성된 문자열이 주어진다. 모든 $1 \le i \le N$에 대해 i번째 문자가 (인 경우 속이 위로 온 빵이 i번 기계에,)인 경우 속이 아래로 온 빵이 i번 기계에 들어 있다는 의미이다.

셋째 줄에 양의 정수 Q가 주어진다.

넷째 줄부터 Q개의 줄에 세 개의 양의 정수 t, a, b로 상황이 주어진다. t는 2 이하이며, $1 \le a \le b \le N$이다.

$t = 1$인 경우 a번 기계부터 b번 기계까지의 빵을 뒤집어 줘야 함을 의미한다.

$t = 2$인 경우 a번 기계부터 b번 기계까지의 빵을 꺼내 버거운 버거로 만들어 달라는 주문이 들어왔음을 의미한다.

출력 형식

각 주문($t = 2$)마다 버거운 버거의 높이를 한 줄에 하나씩 출력한다.

제약 조건

- $1 \le N \le 1,000,000$
- $1 \le Q \le 300,000$

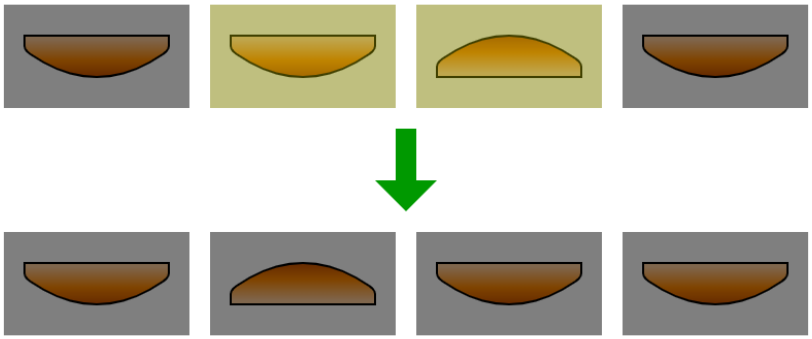
예제

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

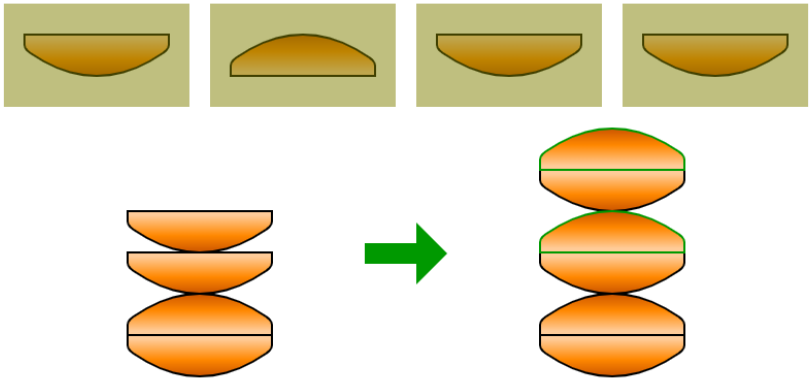
예제 설명

입력 예제에서는 총 다섯 번의 상황이 발생한다.

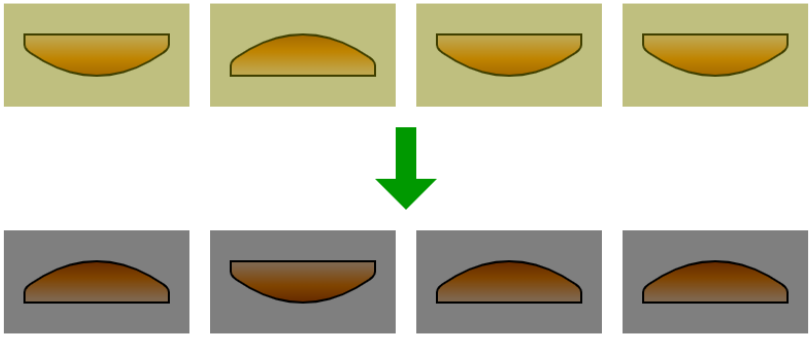
첫 번째 상황은 2번 기계부터 3번 기계까지에 들어 있는 빵을 모두 뒤집어 주어야 하는 상황이다. 기계 안에 있는 빵을 하나씩 뒤집음에 유의하라.



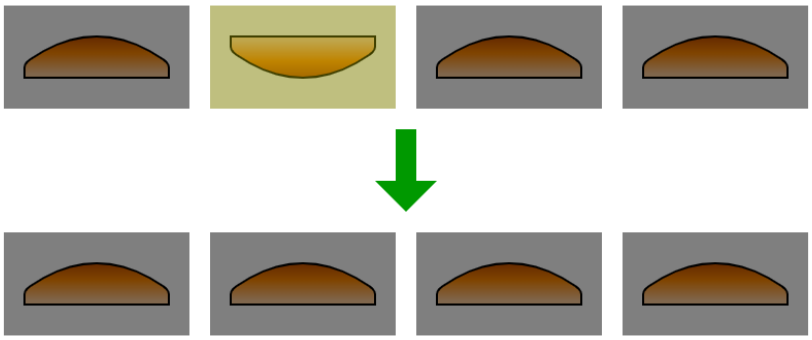
두 번째 상황은 손님이 1번 기계부터 4번 기계까지의 빵을 모두 꺼내 버거운 버거를 만들어 달라는 주문이다. 빵을 꺼내서 쌓으면 왼쪽 아래와 같이 되고, 두 개의 빵을 추가해 높이 6의 버거운 버거를 만들 수 있으며, 이보다 높이가 더 낮은 버거운 버거를 만들 수는 없다. 빵을 추가할 때는 기존에 쌓여 있던 빵의 순서는 유지되어야 하며, 기존 빵을 뒤집을 수 없음에 유의하라.



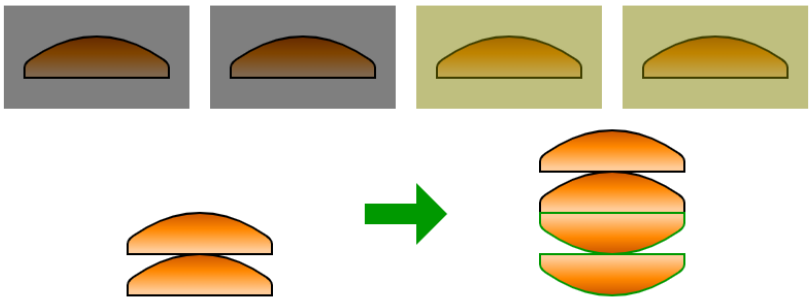
세 번째 상황은 1번 기계부터 4번 기계까지에 들어 있는 빵을 모두 뒤집어 주어야 하는 상황이다. 첫 번째 주문을 처리했지만, 키파는 기계에 빵이 비는 것을 싫어하기 때문에 모두 원래 방향대로 채워 두었음에 유의하라.



네 번째 상황은 2번 기계에 들어 있는 빵을 뒤집어 주어야 하는 상황이다.



다섯 번째 상황은 손님이 3번 기계부터 4번 기계까지의 빵을 모두 꺼내 버거운 버거를 만들어 달라는 주문이다. 빵을 꺼내서 쌓으면 왼쪽 아래와 같이 되고, 두 개의 빵을 추가해 높이 4의 버거운 버거를 만들 수 있으며, 이보다 높이가 더 낮은 버거운 버거를 만들 수는 없다. 빵을 추가할 때 기존에 쌓아둔 빵과 빵 사이뿐만 아니라 가장 위쪽 혹은 아래쪽에도 넣을 수 있으며, 빵을 넣는 방향은 상관없음에 유의하라.



Problem E. 직선형 분자 만들기

레프는 화학실험 시간에 실험 재료로 분자를 받았다. 분자를 구성하는 원자는 총 N개로 1번부터 N번까지 서로 다른 번호가 붙어 있고, 두 원자를 연결하는 총 M개의 서로 다른 화학 결합이 존재한다. 각 결합은 서로 다른 두 원자를 연결한다. 즉, 분자는 방향성이 없는 그래프로 생각할 수 있다. 실제 화학에서의 예시와 다르게, 분자가 꼭 연결되어 있거나, 2개 이상의 원자로 이루어져 있거나, 4개 이하의 다른 원자와 연결되어 있지는 **않을 수 있음**에 주의하라.

레프는 실험을 위해 최첨단 기계를 사용할 것이다. 레프가 기계에 두 정수 $1 \le L \le R \le N$을 입력하면, 기계는 번호가 L 이상 R 이하인 원자를 **제외한 다른 모든 원자**를 제거한다. x번 원자를 제거한다는 것은, x번 원자 및 x번 원자와 연결된 화학 결합을 모두 제거하는 것을 말한다.

레프는 이렇게 만든 분자를 보고서에 적어야 하는데, 보고서에 적을 분자는 **직선형**이어야 한다. 즉, 직선 위에 분자를 구성하는 원자들을 적당히 재배치하여 인접한 원자들끼리는 모두 화학 결합으로 이어져 있고, 인접하지 않은 원자들끼리는 결합이 없도록 할 수 있어야 한다.

숙련된 화학도인 레프는 이런 분자는 얼마든지 많이 만들 수 있지만, 보고서 추가 점수를 받기 위해 만들 수 있는 직선형 분자가 총 몇 가지인지도 적어 내려고 한다. 즉, 레프는 직선형 분자를 만들기 위해 실험 기계에 입력할 수 있는 정수 쌍 (L, R)의 개수를 알고 싶다. 레프를 위해 가능한 분자의 개수를 구하는 프로그램을 작성해주자.

입력 형식

입력의 첫째 줄에는 두 정수 N, M이 공백을 사이에 두고 주어진다. 분자에 포함되는 원자의 개수가 N개이고, 화학 결합의 개수가 M개임을 의미한다.

둘째 줄부터 M개의 줄에 걸쳐 화학 결합에 대한 정보가 주어진다. $(i+1)$번째 줄에는 두 정수 A_i, B_i가 공백을 사이에 두고 주어지는데, 이는 A_i번 원자와 B_i번 원자를 잇는 화학 결합이 존재한다는 의미이다.

출력 형식

첫째 줄에 실험 기계에 입력할 수 있는 정수 쌍의 개수를 출력한다.

제약 조건

- $1 \le N \le 250,000$
- $0 \le M \le 250,000$
- $1 \le A_i, B_i \le N$, $A_i \ne B_i$, $i \ne j$에 대해 $\{A_i, B_i\} \ne \{A_j, B_j\}$

예제

표준 입력(stdin)	표준 출력(stdout)
3 3 1 2 2 3 3 1	5
8 7 2 1 1 4 4 3 3 8 8 7 7 5 5 6	17
12 12 1 2 3 4 5 6 7 8 9 10 11 12 2 4 4 6 6 8 8 10 10 12 12 2	28

예제 설명

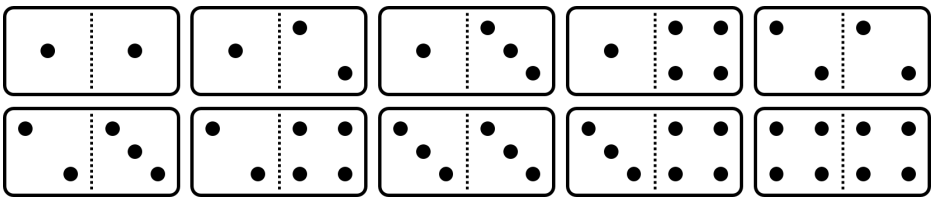
예제 1: 가능한 (L, R) 쌍은 $(1,1), (1,2), (2,2), (2,3), (3,3)$이다. $(1,3)$의 경우, 어떻게 배치해도 인접하지 않은 원자 간의 결합이 존재하게 된다.

예제 3: 벤젠 분자이다. 1개, 2개, 3개, 4개의 원자만 남길 수 있는 경우의 수는 각각 12가지, 6가지, 5가지, 5가지이다.

Problem F. $2 \times M$ 타일링

상수는 프로그래밍 대회 상품으로 타일들이 들어 있는 상자를 받았다. 이 타일들은 두 개의 정사각형 칸이 변을 맞대고 붙어 있는 직사각형 모양이고, 각 칸에는 정해진 개수만큼의 점이 찍혀 있다. 편의상 한 칸에는 a개, 다른 칸에는 b개의 점이 찍힌 타일을 (a, b) 타일이라고 하자.

타일 상자는 크기에 따라서 안에 들어 있는 타일들의 구성이 달라지며, 상수가 받은 타일 상자는 크기가 K인 상자이다. 이 상자에는 $1 \le a \le b \le K$를 만족하는 모든 정수 쌍 a, b에 대해 (a, b) 타일이 각각 하나씩, 총 $\frac{K(K+1)}{2}$개의 타일이 들어 있다.



$K = 4$인 경우의 예시

머릿속이 온통 알고리즘 문제뿐인 상수는 타일의 모양을 보자마자 유명한 "2×N 타일링" 문제가 생각났다. 그래서 상수도 상자에 들어 있는 타일들 중 M개의 타일을 골라서 $2 \times M$ 격자판을 채워 보기로 했다.

타일을 놓을 때는 반드시 격자판의 칸에 맞춰서 놓아야 하며, 가로로 놓을지 세로로 놓을지는 자유롭게 정할 수 있다. 같은 방향이더라도 타일을 180도 돌려서 두 칸의 순서를 바꿔서 놓을 수도 있다. 당연히 타일이 격자판을 벗어나거나 두 타일이 겹쳐서는 안 되고, 따라서 M개의 타일을 올바르게 놓으면 $2M$개의 칸을 전부 덮게 된다.

그러나 단순히 격자판을 채우는 것은 너무 쉽다고 생각한 상수는 타일들에 찍혀 있는 점들을 이용해서 다음과 같은 조건들을 추가로 만들었다.

- 격자판의 두 가로줄에는 서로 같은 개수만큼의 점이 찍혀 있어야 한다.
- 격자판의 모든 세로줄에는 점이 정확히 $K + 1$개씩 찍혀 있어야 한다.

상수가 주어진 조건에 맞게 격자판을 채울 수 있을지 알아보자.

입력 형식

첫 줄에 두 개의 정수 K과 M이 공백을 사이에 두고 주어진다. K는 상수가 받은 타일 상자의 크기, M은 상수가 채우려는 격자판의 너비를 의미한다.

출력 형식

상수가 조건에 맞게 격자판을 채울 수 있다면 첫 줄에 `YES`를 출력한다. 그리고 다음 두 줄에 걸쳐 각 줄에 격자판의 각 칸에 찍혀 있는 점의 개수를 의미하는 M개의 정수를 공백으로 구분하여 출력한다. 단, 해당 칸을 덮는 타일이 세로로 놓여 있을 경우는 앞에 `-`를 붙여 음수로 출력한다.

격자판을 채울 수 없는 경우는 첫 줄에 `NO`를 출력한다.

제약 조건

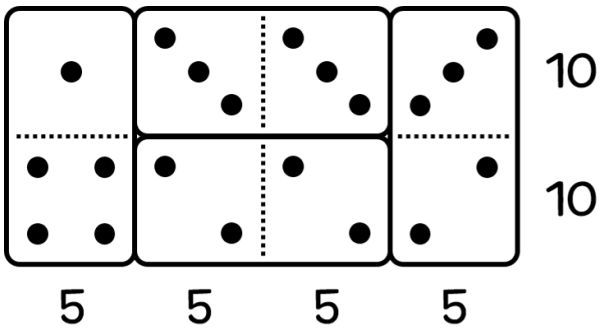
- $1 \leq K \leq 15$
- $1 \leq M \leq \frac{K(K+1)}{2}$

예제

표준 입력(stdin)	표준 출력(stdout)
4 4	YES -1 3 3 -3 -4 2 2 -2
2 3	NO

예제 설명

첫 번째 예제의 출력은 다음과 같은 배치를 나타낸다.



Problem G. 3분 그래프 리턴즈

3분 그래프라는 음식을 아는가? 3분 그래프는 서울대학교 컴퓨터공학부 학생들의 지친 심신을 위로해 주는 보양식으로, 이름에서 알 수 있듯 간편한 조리 방식과 깊은 풍미를 가지고 있어 널리 사랑받는 음식이다. 작년에 출시된 3분 그래프 평면맛의 인기에 힘입어, 3분 그래프 제조업체 스눕스는 야심작 **3분 그래프 구간맛**을 출시했다.

3분 그래프 구간맛. 국산 정점을 쓴다.

3분 그래프 구간맛은 이름에서 알 수 있듯 정점이 N개인 **구간 그래프**의 형태이다. 각 정점은 위치가 고정된 수직선 상의 구간 형태이며, 겹치는 구간끼리는 간선으로 이어져 고정되어 있다. 끝점끼리 겹치는 구간도 겹치는 구간으로 생각한다. i번째 정점은 $[s_i, e_i]$의 구간으로 나타나며, t_i의 맛을 가지고 있다.

서울대학교 컴퓨터공학부는 아니지만 미식가인 규민이는 큰 마음을 먹고 3분 그래프 구간맛을 샀다. 그런데 규민이는 사이클 알러지가 있기 때문에, **정점을 몇 개 빼내서** 모든 사이클을 제거한 뒤에 먹어야 한다. 물론 정점을 빼낼 때는 그 정점에 연결된 간선도 자연스럽게 제거된다.

규민이는 3분 그래프 구간맛을 최대한 맛있게 즐기고 싶기 때문에, 몇 개의 정점을 빼낸 뒤 남아 있는 정점들의 맛을 합한 값이 최대가 되었으면 한다. 규민이가 3분 그래프를 3분 안에 먹을 수 있도록 여러분이 이 작업을 대신 수행해주자.

입력 형식

입력의 첫째 줄에는 구간의 개수 N이 주어진다.

둘째 줄부터 N개의 줄에 걸쳐 각 구간을 나타내는 정보가 주어진다. $(i+1)$번째 줄에는 세 정수 s_i, e_i, t_i가 공백을 사이에 두고 주어지는데, 이는 i번째 구간이 $[s_i, e_i]$이며, t_i의 맛을 가지고 있다는 의미이다.

출력 형식

첫째 줄에 규민이가 먹을 수 있는 정점들의 맛을 합한 값의 최댓값을 출력한다.

제약 조건

- $1 \le N \le 250,000$
- $1 \le s_i \le e_i \le 500,000,\ 1 \le t_i \le 10^{12} \quad (1 \le i \le N)$

예제

표준 입력(stdin)	표준 출력(stdout)
3 1 3 10 3 5 20 5 7 30	60
3 1 3 1 2 3 2 3 3 3	5

Problem H. 계주 코스 정하기

승현이는 한 학교에서 선생님으로 근무하고 있다. 이번에 운동회를 담당하게 된 승현이는 릴레이 계주의 코스 중 하나를 어떻게 만들지 고심 중이다.

이 코스는 N행 M열의 구역으로 나뉜 공간에서 치러질 것이며, 위에서부터 i번째 행, 왼쪽에서부터 j번째 열의 구역은 (i, j)로 표현된다. 승현이는 1열의 한 구역을 시작 지점으로, M열의 한 구역을 도착 지점으로 하는 코스를 만들고자 한다. 이때, 도착 지점은 시작 지점보다 위쪽 구역은 아니어야 한다. 즉, 코스가 구역 $(S, 1)$에서 시작하여 (T, M)에서 끝난다고 할 때, $1 \le S \le T \le N$가 성립해야 한다. 따라서 가능한 시작 지점과 도착 지점의 후보는 $\frac{N(N+1)}{2}$가지가 존재한다.

학생들은 최단 경로로 도착 지점에 도달하고 싶어 하기 때문에, 시작 지점에서 출발하여 오른쪽이나 아래쪽으로 인접한 구역으로 이동하는 것만을 반복하여 도착 지점에 도달하고자 할 것이다.

승현이네 학교의 전통은 릴레이 계주에서 바통 대신 선인장을 들고 달리는 것이다. 그러나 선인장은 추위에 약하기 때문에, 만약 학생이 선인장을 들고 기온이 0도 미만인 곳을 지나가게 된다면 선인장은 시들고 만다. 이 릴레이 경주는 선인장이 시들면 탈락 처리되기 때문에, 학생들은 기온이 0도 이상인 구역만 지나가야 한다.

i행의 구역은 북서풍의 영향으로 A_i도의 영향을 받고, j열의 구역은 동남풍의 영향으로 B_j도의 영향을 받는다. 따라서, 구역 (i, j)의 기온은 $(A_i + B_j)$도이다.

시작 지점과 도착 지점이 정해졌을 때, 기온이 0도 이상인 구역만 지나면서 시작 지점부터 오른쪽 또는 아래쪽으로 인접한 구역으로 이동하는 것을 반복하여 도착 지점에 도달하는 것이 가능하다면 이 경우는 릴레이 계주 코스로 가능한 경우이다. 이때, 시작 지점과 도착 지점 역시 기온이 0도 이상이어야 한다.

$\frac{N(N+1)}{2}$가지의 총 후보 중, 가능한 시작 지점과 도착 지점 쌍의 경우의 수를 구하여라.

입력 형식

첫 줄에 두 정수 N과 M이 공백을 사이에 두고 주어진다.

둘째 줄에 N개의 정수 $A_1, A_2, \cdots, A_N$이 공백을 사이에 두고 주어진다.

셋째 줄에 M개의 정수 $B_1, B_2, \cdots, B_M$이 공백을 사이에 두고 주어진다.

출력 형식

첫 줄에 가능한 코스의 개수, 즉 가능한 (시작 지점, 도착 지점) 쌍의 개수를 출력한다.

제약 조건

- $1 \le N, M \le 200,000$
- $-10^9 \le A_i \le 10^9$ $(1 \le i \le N)$
- $-10^9 \le B_i \le 10^9$ $(1 \le i \le M)$

예제

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

Problem I. 연금술사

각고의 노력 끝에 현자의 돌을 얻은 연금술사는 모든 물질로 금을 만들 수 있는 것은 아니라는 사실을 깨닫고 절망하였다. 그러나, 그는 현자의 돌을 이용해 가치가 낮은 광물로 가치가 높은 광물을 만들 수 있다는 사실을 발견하였다. 물론, 가치가 높은 광물을 사용해서 얻은 결과물의 가치가 낮을 수도 있다. 이 때 투입한 광물들은 사라지고, 결과 광물 하나만 남는다.

구체적으로, 광물 $k(k \geq 1)$개와 현자의 돌을 이용해 새로운 광물을 만드는 경우를 생각해 보자. 사용한 광물 k개의 가치를 각각 $x_1, x_2, \cdots, x_k$라 할 때, $S = \{x_1, x_2, \cdots, x_k\}$에 포함되지 않는 가장 작은 음이 아닌 정수 $\mathrm{MEX}(S)$가 최종 산물의 가치가 된다. 예를 들어, 가치가 각각 $0, 1, 4, 2, 1$인 광물과 현자의 돌을 이용해 새로운 광물을 만드는 경우, $\mathrm{MEX}(\{0, 1, 4, 2, 1\}) = 3$이므로 결과 광물의 가치는 3이 된다.

현재 연금술사는 가치가 0, 1, 2, $\cdots$, $(N-1)$인 광물들을 각각 c_0, c_1, c_2, $\cdots$, c_{N-1}개씩 가지고 있다. 연금술사는 이 광물들과 현자의 돌을 이용해 새로운 광물을 만드는 과정을 여러 번 반복할 수 있다. 결과물로 나온 광물도 다시 재료로 사용할 수 있다. 이 과정에서 현자의 돌은 무한히 사용할 수 있다.

연금술사의 목표는 최종적으로 단 하나의 광물만을 남기면서 그 광물의 가치를 최대로 하는 것이다. 단, 광물을 사용하지 않고 버릴 수는 없다.

연금술사가 최종적으로 얻게 될 단 하나의 광물의 가치의 최댓값을 구하여라.

입력 형식

첫 줄에 정수 N이 주어진다.

두 번째 줄에 N개의 정수 $c_0, c_1, \cdots, c_{N-1}$이 공백으로 구분되어 주어진다. 단, 연금술사는 적어도 하나 이상의 광물을 갖고 있다.

출력 형식

첫 줄에 최종적으로 남은 광물 하나의 가치의 최댓값을 출력한다.

제약 조건

- $1 \leq N \leq 100,000$
- $0 \leq c_i \leq 10^9$
- $c_0 + c_1 + \cdots + c_{N-1} \geq 1$

예제

표준 입력(stdin)	표준 출력(stdout)
1 1	1
2 0 1	1
6 1 0 0 0 0 1	2
5 0 0 0 0 1	4
5 1 0 1 0 1	3

예제 설명

다음은 다섯 번째 예제에 대한 설명이다.

처음에는 가치가 0, 2, 4인 광물이 존재한다.

가치가 0인 광물과 현자의 돌을 이용해 가치가 $\mathrm{MEX}(\{0\}) = 1$인 광물을 만들면 남은 광물들의 가치는 1, 2, 4가 된다.

가치가 4인 광물과 현자의 돌을 이용해 가치가 $\mathrm{MEX}(\{4\}) = 0$인 광물을 만들면 남은 광물들의 가치는 0, 1, 2가 된다.

가치가 0, 1, 2인 광물과 현자의 돌을 이용해 가치가 $\mathrm{MEX}(\{0, 1, 2\}) = 3$인 광물을 만들면 가치가 3인 광물 하나만 남고, 이것이 남은 광물의 가능한 최대 가치이다.