'23 PIMM ALGORITHM PARTY

CONTEST COMMENTARY EDITORIAL

PIMM Algorithm Study

PIMM the Game Developing Club @ Chonnam National University

목차

대회 운영진

출제진

- 박종현 // belline0124
- 고민규 // jjkmk1013
- 김근성 // onsbtyd
- 이윤수 // lys9546
- 정영도 // 0do
- 최정환 // jh01533

검수진

- eric00513
- jk410
- kiwiyou
- lycoris1600
- parkky
- tony9402
- rustiebeats
- utilforever

후원

- utilforever
- belline0124
- jjkmk1013
- lys9546
- onsbtyd
- jh01533
- and 

문제 목록

문제	의도한 난이도	출제자
A. 학점계산프로그램	쉬움	lys9546
B. 알파빌과 베타빌	보통	lys9546
C. 인기투표	조금 어려움	belline0124
D. 동전 탑 게임	어려움	jh01533
E. 미술 시간	어려움	jjkmk1013
F. 나무나무나 심어야지	매우 어려움	onsbtyd
G. PIMM 파티	매우 어려움	0do

- 문제의 난이도 표기는 대회 문제 사이의 상대적인 난이도 차이를 기준으로 작성되었습니다.
  난이도 색상은 solved.ac 문제 레이팅의 동일 색 계열 문제와 같은 난이도임을 담보하는 것은 아닙니다.

대회 결과

2023 전남대학교 PIMM 알고리즘 파티

- 대회 시간: KST 2023-09-03 (일) 15:00 ~ 18:00 (3시간)
- 대회 주소: https://www.acmicpc.net/contest/view/1095
- 참여 방법: 대회 기간 중에 대회 주소에 접속하여 1회 이상 답안 제출
- 문제
  - 문제 구성: 7문제 (solved.ac 기준 Bronze ~ Diamond 난이도 의도됨)
  - 난이도 기준 오름차순 정렬됨
- 스코어보드: 대회 종료 30분 전 프리즈, 대회 종료 시 해제
- 개최자: 전남대학교 게임개발동아리 PIMM
- 운영진: 출제자 6명, 검수자 8명
- solved.ac 프로필 보상
  - "2023 전남대학교 PIMM" 뱃지 (1문제 이상 해결 시)
  - "오늘 즐거웠지" 배경 (2문제 이상 해결 시)

오픈 컨테스트

1. 1위. xiaowuc1
2. 2위. nick832
3. 3위. edenooo
4. 4위. moonrabbit2
5. 5위. jhuni

전남대학교 재학생

- BOJ 핸들 (솔브 수, 페널티 점수)

1. 1위. speciling (3, 74)
2. 2위. lja3723 (3, 274)
3. 3위. zmxncbv777 (3, 289)
4. 4위. sda5129 (2, 20)
5. 5위. stupy3on (2, 40)

특별상 / 추첨상

모든 특별상 / 추첨상 상품은 "스타벅스 아이스 카페 아메리카노 T" 기프티콘입니다.

모든 랜덤 지급 / 추첨상 추첨 기록은 부록에서 확인할 수 있습니다.

특별상

동일한 특별상을 달성한 참가자가 둘 이상일 경우 랜덤으로 1명에게 지급됩니다.

3대 500

- 제목 공개: 공개
- 달성: goop2027
- 조건: 세 문제만 해결해 500점에 가장 가까운 페널티 점수를 획득했다.

언어의 마술사

- 제목 공개: 공개
- 달성: wizardrabbit
- 조건: 가장 다양한 언어로 답안을 제출해 문제를 해결했다.

이 몸 《등장》

- 제목 공개: 비공개
- 달성: saywoo
- 조건: 대회가 시작되고 20분이 지나고 처음으로 답안을 제출했다.

한 번만 투표할 수 있습니다.

- 제목 공개: 비공개
- 달성: kaorin
- 조건: 처음으로 "인기투표" 문제를 한 번에 통과했다.

추첨상

pk661 delena0702 YunGoon
cinador xiaowuc1 ncy09
kkgojn chang1021

'23 PIMM ALGORITHM PARTY

PROBLEM STATEMENTS

A. 학점계산프로그램

- 시간 제한: 1초
- 메모리 제한: 512MB

문제

종현은 영도와 함께 학교 포털 사이트에서 데이터를 가져와 본인의 학점을 계산하는 프로그램을 만들고 있다. 종현은 영도가 사이트에서 추출한 등급 데이터를 사용해서 평균 학점을 출력하고자 한다.

등급별 학점은 다음의 표를 따른다.

등급	학점
A+	4.5
A	4.0
B+	3.5
B	3.0
C+	2.5
C	2.0
D+	1.5
D	1.0
F	0.0

입력

첫째 줄에 과목별 등급이 나열된 문자열 S가 주어진다. 등급 사이에는 별도의 구분자가 없다. 문자열은

표에 있는 문자들로만 이루어져 있으며, 최대 $1\,000$ 글자로 이루어져 있다.

출력

문자열 S에 나열된 등급으로 구한 학점의 산술평균을 첫째 줄에 출력한다. 정답과 출력값의 절대/상대 오

차는 10^{-4}까지 허용한다.

예제 입력 1

```
A+A
```

예제 출력 1

```
4.25
```

예제 입력 2

```
A+AB+B+C
```

예제 출력 2

```
3.5
```

예제 입력 3

```
CA+ABBB+A
```

예제 출력 3

```
3.42857
```

B. 알파빌과 베타빌

- 시간 제한: 1초
- 메모리 제한: 512MB

문제

민규와 친구들은 바로 옆에 붙어있는 두 빌라, 알파빌과 베타빌에 살고 있다. 이 두 빌라 중 알파빌은 싼값에 좋은 빌라라서 너무 인기가 많아 입주하려는 사람들이 줄을 선다. 민규의 친구들 역시 대기 번호를 받아 알파빌의 대기 명단에 적혀있다. 알파빌 입주에 실패한 친구들은 어쩔 수 없이 조금 더 비싼 베타빌에 들어가게 될 것이다. 이를 안타까워한 민규는 더 많은 친구를 알파빌에 입주시키기 위해 집주인 몰래 대기 명단을 바꾸려고 한다.

대기 명단에는 입주하려는 사람들의 대기 번호가 입주하는 순서대로 왼쪽에서 오른쪽으로 적혀 있으며, 대기 번호는 1번부터 N번까지의 서로 다른 정수이다.

민규는 한 번 명단을 바꿀 때 번호 두 개를 선택해서 서로 위치를 교환할 수 있다. 대기 명단을 너무 많이 바꾸면 집주인이 눈치를 챌 수 있기 때문에, 민규는 가능한 한 최소한으로 명단을 바꾸려고 한다. 민규의 모든 친구가 친구가 아닌 사람들보다 먼저 입주하도록 명단을 바꿀 때, 최소 교환 횟수를 출력하자.

입력

첫 번째 줄에 대기 명단에 적힌 수의 개수 N과 민규 친구의 수 M이 공백으로 구분되어 주어진다. ($1 \leq M \leq N \leq 1\,000$)

두 번째 줄에 대기 명단에 적힌 N개의 정수가 주어진다.

세 번째 줄에 민규 친구의 대기 번호를 나타내는 M개의 정수가 주어진다.

출력

모든 친구들이 먼저 입주할 수 있도록 명단을 바꾸는 최소 횟수를 출력한다.

예제 입력 1

```
5 2
1 2 3 4 5
3 4
```

예제 출력 1

```
2
```

1번과 3번을 바꾸고 2번과 4번을 바꾸면 3 4 1 2 5가 되어 모든 친구들이 먼저 입주 할 수 있다.

예제 입력 2

```
5 2
3 1 2 4 5
3 4
```

예제 출력 2

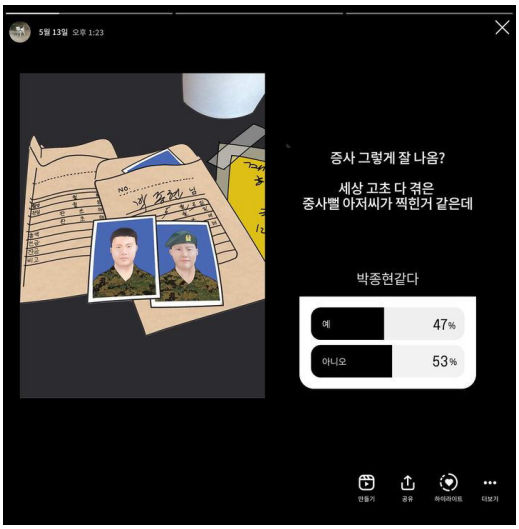
```
1
```

C. 인기투표

- 시간 제한: 1초
- 메모리 제한: 512MB

문제

종현은 인생에서 한 번 있을 군 생활을 기념 겸 추억거리로 남겨두기 위해 증명사진을 찍어보았다. 그런데 웬걸! 사진관으로부터 받은 증명사진 속엔 대학생 상병 박종현이 아니라 세상 고초 다 겪은 직업 군인 중사 박종현이 찍혀있었다. 사진에 불만이 생긴 종현은 생활관 동기들에게 하소연 했지만, 동기들은 종현을 놀리기에 바빴다.



억울한 종현은 SNS에 이 사진이 정말로 자신과 닮았는지 공개 투표를 부쳐, 사진관에서 사진을 이상하게 편집했다고 주장하려고 한다. 투표는 백분율로 공개되고, 철회하거나 번복할 수 없다.

근성은 공개 투표가 친구가 많은 사람의 전유물이라고 생각했기 때문에, 감히 친구가 적은 종현이 SNS에 자신의 증명사진과 공개 투표를 올린 것을 용서하지 못한다. 하지만 근성의 생각과 달리 종현이 새로 사귄 친구가 많을 수도 있으므로, 투표 상황을 지켜보고 종현의 인기를 추정해 보기로 했다.

백분율로 나타나는 투표 결과로부터 전체 투표수를 추측할 수 있으므로, 과거 어느 시점에서의 투표 결과와 현재의 투표 결과를 모두 만족하는 현재의 총투표수를 추정한다.

근성은 종현이 친구가 많으리라 생각하지 않기 때문에, 가능한 총 투표수가 여러 가지라면 근성은 가장 낮은 값을 결과로 받아들일 것이다.

입력

첫째 줄에 투표 항목의 개수 N ($1 \leq N \leq 100$) 과 투표 결과가 표현되는 소수점 자릿수 P ($0 \leq P \leq 6$)가 공백으로 구분되어 주어진다.

둘째 줄에 이전 시점, 백분율로 표현되는 각 항목의 투표 결과에 10^P를 곱한 값 N개가 공백으로 구분되어 주어진다.

셋째 줄에 현재 시점, 백분율로 표현되는 각 항목의 투표 결과에 10^P를 곱한 값 N개가 공백으로 구분되어 주어진다.

둘째 줄과 셋째 줄에 주어지는 각 값은 반올림이 이루어지지 않은 값이다. 각 시점에서 주어지는 수의 합은 10^{P+2}임이 보장된다.

출력

근성이 추정하는 이전 시점의 총 투표수와 현재 시점의 총 투표수를 공백을 간격으로 순서대로 출력한다.

예제 입력 1

```
2 0
50 50
90 10
```

예제 출력 1

```
2 10
```

이전 시점의 투표 결과는 50%, 50%였고, 현재 시점의 투표 결과는 10%, 90%이다. 따라서 이전 시점 투표자는 최소 2명, 현재 시점 투표자는 최소 10명이다.

예제 입력 2

```
3 2
2000 5000 3000
8000 1000 1000
```

예제 출력 2

```
10 50
```

이전 시점의 투표 결과는 20.00%, 50.00%, 30.00%였고, 현재 시점의 투표 결과는 80.00%, 10.00%, 10.00%이다. 따라서 이전 시점 투표자는 최소 10명, 현재 시점 투표자는 최소 50명이다.

D. 동전 탑 게임

- 시간 제한: 1초
- 메모리 제한: 512MB

문제

동전 탑 게임은 두 명의 플레이어가 각각 A, B개의 동전으로 쌓인 **두 개의 탑**으로 하는 게임이다. 두 플레이어는 서로 턴을 번갈아 가면서 게임을 진행하게 되고 각각의 턴에는 비어있지 않은 동전 탑 하나를 골라 한 개 이상의 동전을 가져가야 한다. 더 이상 가져갈 동전이 없으면 게임은 끝나고 **더 많은 동전을 가져간 플레이어**가 승리한다. 만약 서로 같은 양의 동전을 가져갔다면 후공이 승리한다.

민규는 너무나도 단순한 이 게임을 흥미진진하게 만들고 싶었다. 그래서 게임이 끝날 때까지 승패를 알 수 없도록 정수 K를 두어 **마지막으로 동전을 가져가는 플레이어**에 대한 다음과 같은 규칙을 추가했다.

K가 음이 아닌 정수라면 K개의 동전을 추가로 얻는다. K가 음수라면 $|K|$개의 동전을 잃는다. 만약 보유하고 있던 동전이 $|K|$개보다 작다면 게임에서 패배한다. 민규는 영도와 함께 개선한 게임을 하려 한다. 영도는 게임을 보자마자 필승법을 눈치채고 재빠르게 선후공 선택권을 가져갔다. 두 플레이어가 최적의 방법으로 게임을 진행한다고 가정했을 때 영도가 이기려면 선공과 후공 중 어느 것을 선택해야 할까?

입력

첫째 줄에 A, B, K가 공백으로 구분되어 주어진다.

출력

선공을 선택해야 한다면 `First`, 후공을 선택해야 한다면 `Second`를 출력한다.

제한

- $0 \le A, B \le 1\,000$
- $A + B > 0$
- $|K| \le 2\,000$
- 입력으로 주어지는 모든 수는 정수이다.

예제 입력 1

```
10 7 4
```

예제 출력 1

```
First
```

예제 입력 2

```
4 4 -2
```

예제 출력 2

```
Second
```

E. 미술 시간

- 시간 제한: 1초
- 메모리 제한: 512MB

문제

민규는 어느 날 문득 그림이 그리고 싶어졌다. 마침 옆에는 미술과 PS에 통달한 정환이 백준 문제를 풀고 있었다. 민규는 정환에게 그림 잘 그리는 법을 물어보았고, 정환은 길이가 N칸인 긴 직사각형 모양의 색칠되지 않은 종이를 주며 다음과 같은 자신의 지시를 따르면 멋진 그림이 완성될 것이라고 얘기했다.

- a b x: a번째 칸부터 b번째 칸까지, 색칠되지 않은 칸을 x번째 색으로 칠한다.

하지만 민규는 이 과정이 너무 지루해 누군가 대신해 주길 바라고 있다. 여러분이 대신 민규의 그림을 완성해 주자.

입력

첫째 줄에 칸의 개수 N ($1 \le N \le 10^5$)과 쿼리의 개수 Q ($1 \le Q \le 10^5$)가 공백으로 구분되어 주어진다.

둘째 줄부터 Q개의 줄에는 쿼리가 한 줄에 하나씩 주어진다. ($1 \le a \le b \le N; 1 \le x \le 10^9$)

출력

모든 쿼리를 수행한 후 각 칸의 색을 한 줄에 공백을 사이에 두고 출력한다. 이때 색칠되지 않은 칸은 0을 출력한다.

예제 입력 1

```
6 2
4 4 3
1 5 1
```

예제 출력 1

```
1 1 1 3 1 0
```

F. 나무나무나 심어야지

- 시간 제한: 2초
- 메모리 제한: 512MB

문제

근성은 나무에 관심이 많다.

비록 지금은 개발을 하고 있지만, 그렇다고 나무에 대한 애정이 식은 것은 아니다. 어느 날 이진 트리를 가지고 놀던 근성은 이진 트리는 나무임에도 열매가 안 열린다는 사실을 깨닫고 큰 충격에 빠졌다. 근성은 나무는 열매가 반드시 열려야 한다 생각하는 나무 열매..(중략) 론을 밀고 있었기에 나무 열매가 열리는 트리 그래프를 만들었고 이에 "**나무나무**"라 이름 지었다.

나무나무의 특징은 다음과 같다.

1. 이 트리의 1번 정점은 뿌리를 의미한다. 이 트리는 뿌리로부터 위로 뻗어 나간다.
2. 1번 정점을 제외한 정점은 가지가 갈라지거나 끝나는 지점을 의미한다. 정점에는 가지가 연결될 수 있다. 단, 다른 정점에서 갈라진 가지가 정점에서 합쳐지지는 않는다.
3. 간선은 가지를 의미한다. 가지 양 끝에는 반드시 정점이 존재한다.
4. 정점에는 최대 1개의 열매가 열릴 수 있다. 뿌리에는 열매가 열리지 않는다. 매 쿼리에 앞서 열매가 열릴 수 있는 정점에는 열매가 열린다.

트리를 만든 후 무엇을 할 수 있을까 고민하던 중 아래와 같은 두 가지를 생각해 냈다!

- `1 i j w` : (접목) 임의의 정점 i에 가지를 붙인다.
  - 정점 i는 뿌리와 같은 그래프에 속하는 정점이다.
  - 가지 끝에는 정점이 항상 존재하기에 j번으로 번호를 붙인 정점이 새로 생긴 가지 반대쪽에 같이 붙는다.
  - 정점 j에는 w 무게의 열매가 달린다. $w = 0$은 열매가 열리지 않는다는 뜻이다.
  - $(1 \leq i, j \leq N + M; 0 \leq w \leq 500)$
- `2 i` : (수확) 정점 i와, 그 위로 연결된 모든 정점의 열매를 떨어트리려면 몇의 힘으로 흔들어야 할지 출력한다.
  - 임의의 정점 i를 흔들면 해당 정점 위로 연결된 가지, 정점들이 모두 같은 힘으로 흔들린다.

- 정점 i를 흔들어 모든 열매를 떨어뜨리기 위해서는 힘이 정점 i와 그 위로 연결된 모든 정점에 달린 열매 무게의 합이 되어야 한다.
- 0의 힘으로 흔들 수는 없기에 만약, 무게 합이 0이라면 -1 을 출력한다.
- $(1 \leq i \leq N + M)$

쿼리에 주어지는 수는 모두 정수이고, 올바른 입력임을 보장한다. 또한 수확 쿼리는 1회 이상 주어진다.

그런데 근성은 이 쿼리를 만들다 갑자기 동아리방에 가야 한다며 떠났다. 여러분이 대신 풀어주자.

입력

첫째 줄에 최초 정점의 수 N, 쿼리의 수 M이 공백으로 구분되어 주어진다. 최초의 정점에는 1 이상 N 이하의 번호가 중복되지 않게 붙어있다. $(2 \leq N, M \leq 100\,000)$

둘째 줄에 1번부터 N번까지, 각 정점이 어느 정점의 바로 위에 가지로 연결되어 있는지 공백으로 구분되어 주어진다. 1번은 뿌리이므로 `-1`` 이 주어진다. 모든 정점은 최종적으로 뿌리와 같은 그래프에 속하지만, 입력 도중에는 속하지 않을 수 있다.

셋째 줄에 1번부터 N번까지, 각 정점이 가지고 있는 열매의 무게 w_i 가 공백으로 구분되어 주어진다. ($0 \leq w_i \leq 500$) 0이라면 열매가 열리지 않는 것이고 뿌리에는 열매가 열리지 않는다.

이후 넷째 줄부터 M개의 줄에 걸쳐 쿼리가 주어진다.

출력

수확 쿼리가 들어올 때 몇의 힘으로 흔들어야 할지 출력한다. 단, 흔들어야 할 힘이 0이라면 `-1`을 출력한다.

예제 입력 1

```
3 7
-1 1 1
0 0 3
2 1
1 2 7 4
1 7 4 5
```

```
1 7 5 0
2 4
2 7
2 1
```

예제 출력 1

```
3
5
9
12
```

예제 입력 2

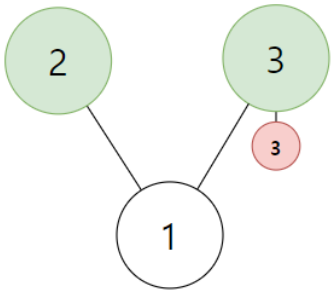
```
4 3
-1 1 1 1
0 0 0 0
2 1
1 2 7 500
2 1
```

예제 출력 2

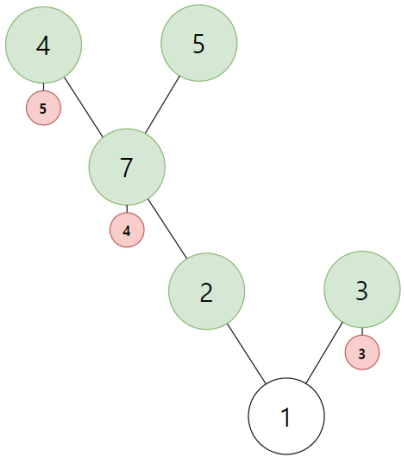
```
-1
500
```

힌트

1번 예제 테스트 케이스를 확인해보자.

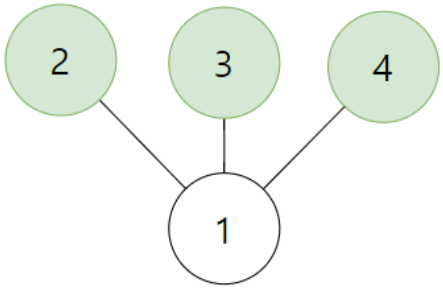


최초 트리의 형태이다. 첫 번째 쿼리 시 1번 정점을 흔들면 2, 3번 정점이 같이 흔들리고 3번 정점에 달린 열매를 떨어트리기 위해 3의 힘으로 흔들어야 한다.

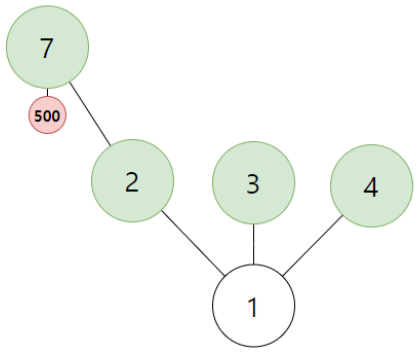


3개 접목을 진행한 모습이다. 4번 정점을 흔들면 4번만 흔들리고, 4번에 달린 열매를 떨어트리기 위해 5의 힘으로 흔들어야 한다. 마찬가지로 7번 정점을 흔들면 7, 4, 5번이 흔들리고 4번과 7번의 열매를 떨어트리기 위해 9의 힘으로 흔들어야 한다.

2번 예제를 확인해보자.



첫 번째 쿼리 시 1번 정점을 흔들면 2, 3, 4번이 같이 흔들리지만, 열매가 달려 있지 않다. 흔들어야 할 힘이 0이기에 -1을 출력한다.



1번 정점을 흔들면 2, 3, 4, 7번이 같이 흔들리고, 7번의 열매를 떨어트리기 위해 500의 힘으로 흔들어야 한다.

G. PIMM 파티

- 시간 제한: 1초
- 메모리 제한: 1024MB

문제

PIMM 회장인 민규는 여름방학을 맞이해 파티를 열기로 했다! 회장 역할에 진심인 민규는 **모두가 행복한 파티**를 만들고자 한다.

수석 인간관계 연구원인 종현의 말에 따르면, PIMM 회원들은 근성과 같은 **인싸(I)**와, 윤수와 같은 **아싸(A)**로 분류할 수 있다. **인싸**들은 다른 사람과 말하는 것을 좋아하며, 상하좌우로 인접한 각 1칸 안에 1명 이상의 사람이 있으면 행복해 한다. **아싸**들은 다른 사람의 쓸데없는 말을 계속 듣는 것을 싫어하며, 상하좌우로 인접한 1칸 안에 **인싸**가 없으면 행복해 한다.

민규는 행복한 파티를 위해, $N \times M$의 직사각형 형태로 이루어진 파티룸을 빌렸다. 민규는 정환에게서 **인싸** X명과, **아싸** Y명이 파티에 참여한다고 전달받았다.

영도는 민규를 대신하여, 파티룸에 인원들의 자리를 배정하는 역할을 맡게 되었다. 영도를 도와서 **아싸**들이 먼저 배치된 파티룸의 정보가 주어졌을 때, 모두가 행복한 파티를 만드는 경우의 수를 구해보자.

입력

첫 줄에는 N, M, X, Y가 공백으로 구분되어 한 줄에 주어진다. ($1 \leq N, M \leq 7$; $1 \leq X, Y \leq N \times M$; $X + Y \leq N \times M$)

다음 N개의 줄에는 아싸가 배치된 파티룸의 정보가 길이 M의 문자열로 주어진다. *는 빈칸이고, A는 아싸가 있는 위치이다.

출력

파티룸에서 인싸를 배치할 X개의 칸을 순서에 상관없이 선택할 때, 해당하는 경우가 모두가 행복한 파티가 되는 경우의 수의 합을 $1\,000\,000\,007$로 나눈 나머지를 한 줄에 출력한다.

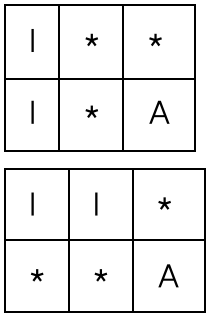
예제 입력 1

```
2 3 2 1
***
**A
```

예제 출력 1

```
2
```

예제 1에서 가능한 경우는 다음과 같이 총 두 가지이다. 'I'는 **인싸**이다.



예제 입력 2

```
1 4 2 1
*A**
```

예제 출력 2

```
0
```

예제 2에서는 인원을 어떻게 배치하더라도 모든 인원의 행복을 충족시킬 수 없다.

예제 입력 3

```
3 4 3 2
A***
A***
****
```

예제 출력 3

```
12
```

'23 PIMM ALGORITHM PARTY

PROBLEM SOLVING DESCRIPTIONS

A. 학점계산프로그램

쉬움

태그	#implementation #math #string
출제자	lys9546
검수자	eric00513 jk410 kiwiyou lycoris1600 parkky rustiebeats tony9402 utilforever
첫 해결	asdf1705 (1분)
정답률	69.302%

입력받은 문자열을 파싱하여 표에 있는 값으로 변환한 후 평균을 구해주면 됩니다.
+ 가 붙은 과목의 경우에 안 붙은 과목의 학점보다 0.5점 높다는 것을 이용하여 모든 과목을 파싱하는 것보다 효율적으로 구현할 수 있습니다.

B. 알파빌과 베타빌

보통

태그	#greedy
출제자	lys9546
검수자	eric00513 jk410 kiwiyou lycoris1600 parkky rustiebeats tony9402 utilforever
첫 해결	xiaowuc1 (3분)
정답률	65.761%

대기 명단의 왼쪽에서부터 M개의 값들이 민규의 친구인지 확인하면서 개수를 세어주면 됩니다. 민규의 친구인지 확인하는 방법에 따라 시간복잡도가 달라집니다.

1. 모든 친구를 순회하면 $O(M^2)$ 안에 해결할 수 있습니다.
2. `set`, `map` 등의 자료구조를 이용하면 $O(M\,logM)$으로 해결할 수 있습니다.
3. 친구의 번호 x가 위치 y에 있다는 정보를 `arr[x] = y` 형태로 저장하면 $O(N+M)$으로 해결할 수 있습니다.

C. 인기투표

조금 어려움

태그	#implementation #gcd #math
출제자	belline0124
검수자	eric00513 jk410 kiwiyou lycoris1600 parkky rustiebeats tony9402 utilforever
첫 해결	xiaowuc1 (12분)
정답률	18.248%

두 개 이상의 정수가 주어졌을 때, 이 정수들을 정수들의 최대공약수로 나누면 주어진 정수들의 비율 관계를 만족하는 최소 정수를 획득할 수 있습니다. 다시 말해 획득한 수는 주어진 수 사이의 비율 관계를 최소 정수로 나타냅니다. 백분율에서 유추할 수 있는 모든 투표수는 이 획득한 수의 배수입니다.

이전 시점은 가능한 투표수의 최소 정수를 그대로 출력하면 되지만, 현재 시점의 추정 결과는 그대로 출력하면 안 됩니다. 투표는 철회하거나 번복할 수 없으므로, 각 항목의 투표수는 각각 현재 시점이 이전 시점보다 항상 크거나 같아야 합니다.

즉, 다음 출력은 틀렸습니다.

입력

```
2 0
10 90
50 50
```

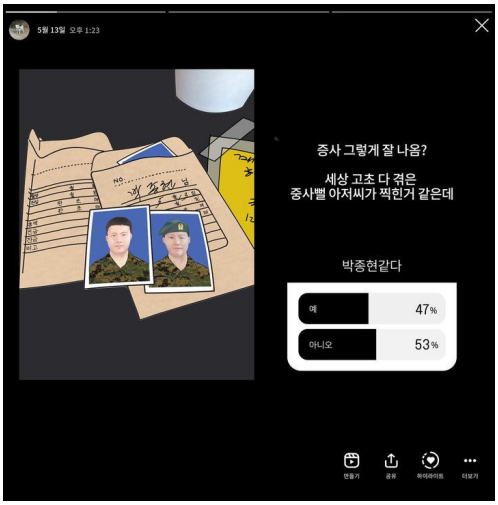
출력

```
10
2
```

정리하면 이전 시점의 투표수는 입력받은 수들을 이 수들의 최대공약수로 나눈 수입니다. 그리고 현재 시점의 투표수는 "각 항목의 투표수가 이전 시점보다 크거나 같은", "현재 시점의 수들의 최대 공약수로 나눈 수의 배수"입니다.

여담

원래 이 문제는 입력을 실수로 제시할 계획이었습니다. 하지만 정밀도 문제로 몇몇 입력이 일부 언어에서 실수에 10^P를 곱해 정수로 만드는 과정이 제대로 수행되지 않을 수 있다는 사실을 확인했습니다. 결국 원래 제시할 실수 입력 대신 실수에 10^P를 곱한 정수를 입력으로 제시하는 것으로 변경하였습니다.



Special Thanks to 강도희 @ PIMM: 이 그림은 PIMM의 디자인 전공 멤버, 강도희 씨의 도움을 받았습니다.

D. 동전 탑 게임

어려움

태그	#game_theory
출제자	jh01533
검수자	eric00513 jk410 kiwiyou lycoris1600 parkky rustiebeats tony9402 utilforever
첫 해결	nick832 (23분)
정답률	10.274%

K가 음수일 때와 K가 음이 아닌 정수일 때로 나눠서 접근하면 됩니다.

K가 음이 아닌 정수일 때 (sub 1)

우선 마지막에 동전을 가져가는 것, 즉 K를 얻는 것을 목표로 두 플레이어가 최선을 다했을 때 어떤 플레이어가 K를 가져갈 수 있을 지 생각해 봅시다. 만약 $A = B$ 라면 선공이 어떻게 가져가든지 후공이 같은 양의 동전을 다른 탑에서 가져가면 항상 마지막에 동전을 가져갈 수 있습니다.

ex: (10, 10) > (10, 7) > (7, 7) > (2, 7) > (2, 2) > (0, 2) > (0, 0)

이때 중요한 관찰은 게임이 어떻게 진행되든 후공이 위와 같은 규칙을 지킨다면 K를 제외하고 서로 가져가는 동전의 양이 항상 똑같다는 것입니다. 따라서 $A = B$ 일 때 후공이 규칙을 지킨다면 선공은 A개 후공은 $A + K$ 개를 가져가므로 후공이 항상 승리하게 됩니다.

그렇다면 $A \neq B$ 개일 때는 어떨까요? 일반성을 잃지 않고 $A > B$ 라고 해봅시다. 처음에 선공이 $A - B$개의 동전을 A개의 동전이 쌓인 탑에서 가져가게 되면 아까와 같은 $A = B$ 구조가 되고, 선후공이 바뀐 상태로 $A = B$ 규칙을 지키며 플레이하면 됩니다.

선공이 규칙을 지킨다면 선공은 $(A - B) + B + K$ 개의 동전을 가져가고 후공은 B개의 동전을 가져갑니다. $(A - B) > 0$ 이므로 선공이 항상 승리하게 됩니다. 따라서 K와 상관없이 $A = B$ 일 때 후공이 필승법을 가지고, $A \neq B$ 일 때 선공이 필승법을 가집니다.

K가 음수일 때 (sub 2)

예외를 제외하고 K가 음이 아닌 정수일 때와 똑같이 플레이하다가 `(1, 0)` 구조를 만들 수 있을 때 `(1, 0)` 구조로 만들면 됩니다.
2가지 예외가 있습니다.

- $A = 1; B = 0$ 일 때 이미 `(1, 0)` 구조인 상황입니다. K가 음수일 때 항상 후공이 승리합니다.
- $K = -1$ 일 때 선공이 첫 턴에 A, B 중 동전이 많이 쌓여있는 탑의 동전을 모두 가져가게 된다면 항상 선공이 승리하게 됩니다.

본 문제는 돌무더기가 2개인 님게임과 같습니다. 만약 동전 탑이 N개일 때도 풀 수 있을까요?

E. 미술 시간

어려움

태그	#disjoint_set #implementation
출제자	jjkmk1013
검수자	eric00513 jk410 kiwiyou lycoris1600 parkky rustiebeats tony9402 utilforever
첫 해결	kcm1700 (3분)
정답률	29.630%

이 문제에 제시되는 쿼리를 그대로 수행할 경우 총 시간복잡도는 $O(QN)$으로 시간초과를 받게 됩니다.

이를 해결하기 위해서는 이미 색칠된 칸을 효과적으로 건너뛰는 방법이 필요합니다.

각 칸의 번호 i에 대해서, 비어있는 칸의 번호 중 i이상인 가장 작은 수 P_i를 정의합시다. 처음에는 모든 $i(1 \le i \le N)$에 대해 $P_i = i$임을 알 수 있습니다. 이 P_i를 i의 비어있는 칸이라고 칭하겠습니다. 또한, 빈칸 k가 색칠될 경우, k의 비어있는 칸 P_k는 그다음 칸인 $k+1$의 비어있는 칸 P_{k+1}이 됨을 알 수 있습니다. 이렇게 정의했을 때, 연속된 칸을 색칠하는 쿼리의 특성상 한번 색칠한 칸을 만나게 되는 경우 P_k를 방문하여 바로 다음 빈칸으로 접근할 수 있게 됩니다.

따라서 분리 집합을 이용해 k번째 칸에 방문했을 때 빈칸이라면 색칠한 후 P_k를 P_{k+1}과 연결하고, 색칠된 칸이라면 P_k를 방문하여 색칠된 구간을 건너뜁니다. 이때 경로 압축을 이용하면 각 칸을 최대 2번씩만 방문하게 되어 총 시간복잡도는 $O(N+Q)$가 됩니다. 주의해야 할 점은 부모를 정하는 기준을 숫자가 높은 순으로 설정해야 올바르게 다음 빈칸으로 연결됩니다.

이외에도 느리게 갱신되는 세그먼트 트리를 통해 쿼리를 $O(logN)$에 처리하거나, BBST가 내장된 언어의 경우 1부터 N까지 삽입 후 색칠된 칸을 제거하는 방식으로 각 과정을 $O(logN)$에 처리할 수 있습니다.

$O(N+Q)$, $O(Q\,logN)$, $O((N+Q)\,logN)$세가지 방법 모두 문제를 해결할 수 있습니다.

F. 나무나무나 심어야지

매우 어려움

태그	#dfs #euler_tour_technique #offline_query
출제자	onsbtyd
검수자	eric00513 jk410 kiwiyou lycoris1600 parkky rustiebeats tony9402 utilforever
첫 해결	gggkik (19분)
정답률	56.667%

나무나무의 특징을 분석해 봅니다. 1, 2, 3번 설명으로 나무나무가 일반적인 트리 구조라는 것을 알 수 있고, 4번 설명으로 각 정점에 최대 한번 값이 더해질 수 있다는 걸 알 수 있습니다. 이 상태에서 수확 쿼리가 들어올 때 주어진 정점과 그 정점의 서브 트리에 있는 열매의 무게 합을 구해야 합니다.

$N + M$이 최대 20만이기에 쿼리를 그때그때 바로 그래프 탐색을 통해 답을 구하려 하면 시간초과가 발생합니다. 그렇기에 조금 더 빠르게 가능하도록 처리를 해둬야 하지만, 접목 쿼리로 인해 트리의 구조가 변경되는 것도 고려해야 합니다.

처리를 위해 접목 쿼리를 먼저 해결합니다. 접목의 특징은 아래와 같습니다.

1. 뿌리와 같은 그래프에 속하는 정점에 붙는다.
2. 한번 접목된 가지는 제거되지 않는다.
3. 열매는 접목 쿼리 시에 붙는다.

1번 특징으로 모든 접목 쿼리는 뿌리가 속한 그래프에서 일어나고, 수확도 마찬가지인 걸 알 수 있습니다. 또 2번 특징으로 인해 해당 트리는 쿼리가 들어올수록 항상 커집니다. 그리고 트리의 규모는 수확 쿼리의 결과인 열매의 무게 합에 영향을 주지 않습니다. 그래서 접목 쿼리를 먼저 수행해 트리를 구성해 두고 후에 열매만 매다는 전략이 가능해집니다.

모든 쿼리를 미리 받아둔 뒤 한번 돌면서 접목 쿼리 만날 때까지만 연결해 최종 상태의 트리를 미리 만들어 둡니다. 이러면 트리 구조 변경이 없어지고 접목 쿼리를 단순히 정점에 값만 추가하는 쿼리로 바꿀 수 있습니다.

다음으로 수확 쿼리를 진행합니다. 접목 쿼리를 "트리 구조의 변경이 없는 값 추가"로 바꿨기에 트리에서 정점과 그 정점의 서브 트리에 있는 열매 무게 합을 빠르게 구할 수 있도록 처리해야 합니다. 만약 정점과 그 정점의 서브 트리를 구간으로 만들 수 있다면 세그먼트 트리를 사용하여 구간 합을 $O(logN)$에 구할 수 있습니다. 트리를 구간으로 만들기 위해서는 DFS와 오일러 경로 테크닉(ETT)을 사용하면 됩니다.

이 과정을 모두 거치면 접목 쿼리와 수확 쿼리가 세그먼트 트리에서의 값 업데이트와 구간 합으로 바뀌게 됩니다. 이제 입력받은 쿼리를 다시 한번 돌면서 실행하면 됩니다.

- 접목 -> 정점에 열매 무게만큼 값을 더해줌
- 수확 -> 정점과 그 서브 트리의 구간 합을 출력

최대로 존재할 수 있는 정점이 $(N + M)$ 개이기에 시간복잡도는 $O(M\,log(N + M))$가 나오게 됩니다.

G. PIMM 파티

매우 어려움

태그	#dp #dp_connection_profile #dp_bitfield
출제자	0do
검수자	jk410 kiwiyou lycoris1600 tony9402
첫 해결	xiaowuc1 (57분)
정답률	25.926%

문제에서 주어진 조건을 먼저 보겠습니다. 인싸와 아싸가 서로 접해있다면, 인싸는 행복해지는 조건을 충족하게 되지만 아싸는 행복하지 못하게 되므로 모두가 행복한 파티의 조건을 충족할 수 없습니다. 즉, 모든 인싸와 모든 아싸끼리 서로 떨어진 상태에서, 모든 인싸가 최소 한 명 이상의 인싸를 붙어있게만 한다면 모두가 행복한 파티의 조건을 충족할 수 있습니다. 이때, 문제에서는 인싸만을 배치할 것을 요구합니다. 따라서, 이 문제는 아싸와 인접한 칸이 아닌 곳에 인싸를 배치하는 경우의 수를 찾는 문제로 변형할 수 있습니다.

이렇게 문제를 변형하고 나면, 다이나믹 프로그래밍 기법을 사용하여 이 문제를 해결할 수 있음을 알게 됩니다. 이 문제는 격자에서 비트마스킹 DP를 사용하는 문제들과 동일하게 접근할 수 있습니다. 이러한 문제들은, 최근 M(또는 N)개 칸의 상태만 저장해도 그 이전 칸의 상태는 경우의 수를 계산하는 데 영향을 주지 않습니다. 문제에서 X명의 인싸를 배치해야 한다는 조건이 추가적으로 주어지는데, 이 부분은 DP 배열에 한 개의 차원을 추가하는 것으로 해결할 수 있습니다.

일반적인 비트마스킹 DP 문제와는 다르게, 이 문제에서는 각 칸의 상태를 3개로 분류해서 저장하여야 합니다. 즉, 3진법 구현이 이 문제에서 요구됩니다. 인싸가 배치되지 않은 칸의 상태는 0, 아직 행복하지 않은 인싸가 배치된 칸의 상태는 1, 행복한 인싸가 배치된 칸의 상태는 2로 저장합니다. 새로운 칸에 2를 배치할 경우, 1이 저장된 인접 칸의 값을 2로 바꿔주어야 합니다. 모든 과정이 끝난 후 정답을 출력할 때 1이 저장된 칸을 답에 고려해서는 안 됩니다.

정답 코드의 시간 복잡도는 $O(3^M NMX)$입니다. 문제에서 주어진 N과 M의 크기가 작기 때문에 비트마스킹 기법을 이용한 $O(4^M NMX)$ 구현 등 다양한 구현 방법이 정답 처리를 받을 수 있으며, 이 부분은 출제 과정에서 의도된 것입니다.

'23 PIMM ALGORITHM PARTY

APPENDIX

특별상 통계 자료, 특별상 대상자 랜덤 선택 및 추첨상 추첨 기록

In [ ]:
```
from random import choices
from json import loads
from collections import defaultdict
from datetime import datetime, timezone, timedelta
from time import mktime

submits_loaded: list = loads(open('submits.json', encoding='utf-8').read())
```

특별상: 3대 500

달성자: goop2027

- 달성자 확인에 코드를 사용하지 않았습니다.

특별상: 언어의 마술사

달성자: wizardrabbit

- Python 3와 PyPy3는 하나의 언어로 판단합니다.

In [ ]:
```
language_submits: defaultdict = defaultdict(set)
for each in submits_loaded:
    submit_id, handle, problem, result, memory, elapsed, language, byte_len, invoked =
    if result == '맞았습니다!!':
        language_submits[handle].add(
            language
            # (''.join([x for x in language if not x.isdigit()])).strip()
        )

print(sorted(language_submits.items(), key=lambda each: len(each[1]), reverse=True)[:5
```

```
[('moonlit', {'Python 3', 'PyPy3', 'C++17'}), ('wizardrabbit', {'Python 3', 'node.js',
'C++17'}), ('tnsgh9603', {'Python 3', 'PyPy3', 'C++17'}), ('yup0927', {'Python 3', 'C++
17'}), ('cinador', {'Python 3', 'PyPy3'})]
```

특별상: 이 몸 《등장》

달성자: nflight11

In [ ]:
```
kst = timezone(timedelta(hours=9))
benchmark = int(mktime(datetime(2023, 9, 3, 15, 20, 0, 0, tzinfo=kst).timetuple()))
invokes: list[tuple] = []

for each in submits_loaded:
    submit_id, handle, problem, result, memory, elapsed, language, byte_len, invoked =
    invokes.append((handle, int(invoked)))
invokes.sort(key=lambda each: each[1])

already_invoked = set()
after_benchmark = []
for each in invokes:
    handle, invoked = each
    if benchmark <= invoked and handle not in already_invoked:
        after_benchmark.append((handle, invoked))
    already_invoked.add(handle)
after_benchmark.sort(key=lambda each: each[1])
after_benchmark[:3]
```

Out[ ]:
```
[('nflight11', 1693722021),
 ('rhgustmd123', 1693722057),
 ('urin6695', 1693722067)]
```

특별상: 한 번만 투표할 수 있습니다.

달성자: kaorin

- 달성자 확인에 코드를 사용하지 않았습니다.

추첨상

pk661 , delena0702 , YunGoon , cinador , xiaowuc1 , ncy09 , kkgojn , chang1021

In [ ]:
```
solves = defaultdict(set)
for each in submits_loaded:
    submit_id, handle, problem, result, memory, elapsed, language, byte_len, invoked =
    if result == '맞았습니다!!':
        solves[handle].add(problem)

print(choices([handle for handle in solves], [len(solves[handle]) ** 2 for handle in s
```

```
['pk661', 'delena0702', 'YunGoon', 'cinador', 'xiaowuc1', 'ncy09', 'kkgojn', 'chang102
1']
```

https://github.com/pimm-dev

Contest hosted at Baekjoon Online Judge by Startlink.

https://startlink.io