UCPC 2023 예선 해설

Official Solutions

전국 대학생 프로그래밍 대회 동아리 연합 · UCPC 2023 운영진

- 이종서 leejseo
- 김도훈 99asdfg
- 김태훈 amel
- 최은규 asdarwin03
- 박신욱 bnb2011
- 곽우석 bubbler
- 김도훈 dohoon
- 양성준 egod1537
- 이상헌 evenharder
- 배근우 functionx
- 유상혁 golazcc83
- 이혜아 ho94949
- 임병찬 hyperbolic
- 최재민 jh05013
- 구재원 jjaewon9
- 권혁준 juny2
- 이경찬 kclee2172
- 이성호 puppy
- 한동규 queued_q
- 박상훈 qwerasdfzxcl
- 장우성 saywoo
- 신용명 tlsdydaud1
- 박수찬 tncks0121
- 이한길 wapas
- 정명준 wider93
- 정우경 man_of_learning
- 채이환 benedict0724
- 이지후 silverwolf
- 윤창기 TAMREF
- 시제연 tlwpdus

Sponsors

SAMSUNG SOFTWARE MEMBERSHIP

MOLOCO

Sponsors

STARTLINK

SOLVED.AC

DEVOCEAN

Sponsors

Sponsors

HYUNDAI AutoEver

Jane Street®

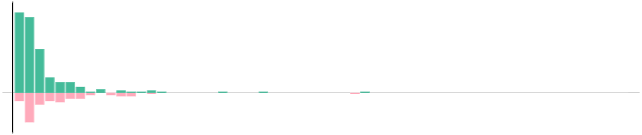
문제		의도한 난이도	출제자
A	체육은 코딩과목 입니다	**Easy**	leejseo
B	물류창고	**Hard**	jjaewon9
C	차량 모듈 제작	**Medium**	wapas, saywoo
D	더 흔한 타일 색칠 문제	**Easy**	bnb2011
E	반전수	**Hard**	kclee2172
F	응원단	**Hard**	golazcc83
G	은하 온라인 마케팅 프로젝트	**Hard**	asdarwin03, 99asdfg
H	팔찌	**Hard**	wider93
I	자석	**Medium**	evenharder
J	다섯 용사의 검	**Challenging**	queued_q
K	세미나 배정	**Medium**	puppy

A. 체육은 코딩과목 입니다

`implementation`

출제진 의도 – **Easy**

- 제출 320번, 정답 219팀 (정답률 70.938%)
- 처음 푼 팀: **25세 김동현 마지막 UCPC -많은 응원 부탁드려요-** (김동현, 안정현, 이하린), 1분
- 출제자: `leejseo`



- “좌향좌”를 오른쪽으로 270도 도는 것으로 생각할 수 있습니다.
- 도는 각도의 총합을 계산해서 360도로 나눈 나머지를 구하면 문제를 해결할 수 있습니다.
- 참고로, 출제자의 정해는 다음과 같습니다:

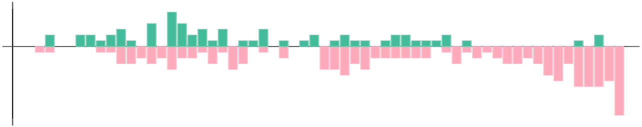
```
print("NESW"[sum(int(input()) for _ in range(10)) % 4])
```

B. 물류창고

`smaller_to_larger`

출제진 의도 – **Hard**

- 제출 214번, 정답 64팀 (정답률 29.907%)
- 처음 푼 팀: **대회장에늦게도착할수록강한팀이다** (대회시작시간을착각한사람, 침대에서뛰어내려서무릎이까진사람, 지하철을반대로탄사람), 9분
- 출제자: `jjaewon9`



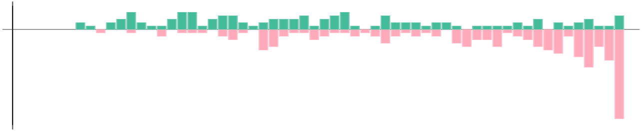
- 빈 그래프에서 시작해, 도로를 이동 상한선의 내림차순으로 하나씩 추가해 봅시다.
- 이번 단계에서 추가하는 이동 상한선이 W인 도로가 서로 다른 연결 요소 T_1, T_2를 연결한다고 합시다. 이는 각 연결 요소에서 하나씩 고른 쌍의 배송 상한선이 W임을 의미합니다.
- 따라서, $W \times$(T_1 속 i번 회사 소유 물류창고 개수)$\times$(T_2 속 i번 회사 소유 물류창고 개수) 만큼 i번 회사에 대한 답이 증가합니다.

- 각 연결 요소마다 (회사 번호, 회사 소유 물류창고 개수)의 map을 관리하고, 두 연결 요소를 합칠 때는 크기가 더 큰 map에 작은 map의 원소들을 하나씩 추가합시다. 이렇게 하면 map의 각 원소들이 최대 $\mathcal{O}(\log N)$ 번 이동하게 됩니다.
- 총 시간복잡도는 $\mathcal{O}(M \log M + N \log^2 N)$ 입니다.

C. 차량 모듈 제작

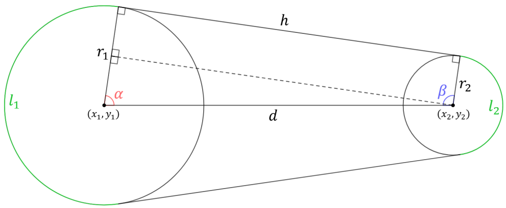
math, geometry, mst
출제진 의도 – **Medium**

- 제출 278번, 정답 115팀 (정답률 41.367%)
- 처음 푼 팀: **25세 김동현 마지막 UCPC -많은 응원 부탁드려요-** (김동현, 안정현, 이하린), 18분
- 출제자: wapas, saywoo



- N개의 기어를 각각의 정점으로 생각해봅시다.
- 2개의 기어 a, b에 대하여 a가 회전하고 있을 때, b가 회전하기 위해 필요한 벨트의 길이를 가중치로 하는 간선을 생각할 수 있습니다.
  - 2개의 기어 a, b가 접하거나 겹치면 벨트가 필요하지 않습니다.

- 접하거나 겹치지 않는다면, 필요한 벨트의 길이를 구하는 방법은 다음과 같습니다.



- 다음의 그림과 같이 2개의 기어가 있을 때, 공통 외접선의 길이인 h는 다음과 같이 구할 수 있습니다.

$$h = \sqrt{d^2 - (r_1 - r_2)^2}$$

- l_1 과 l_2 의 길이를 구하기 위해서는 각 α, β 의 크기를 구해야합니다.
- $\sin\alpha = \frac{h}{d}$ 이므로 다음과 같이 각 α, β 의 크기를 구할 수 있습니다.

$$\alpha = \arcsin\frac{h}{d} \qquad \beta = \pi - \alpha$$

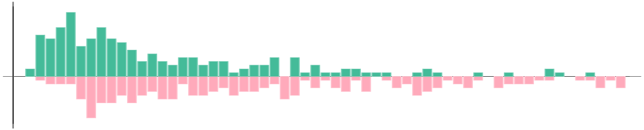
- 각 기어의 반지름과 각 α, β 의 크기를 이용하여 l_1 과 l_2 의 길이를 구할 수 있습니다.
- 최종적으로 $2 \times h + l_1 + l_2$ 가 필요한 벨트의 길이입니다.

- N개의 정점을 모두 연결하는 트리를 만들면 차량 모듈이 정상적으로 작동하므로, 사용하는 벨트의 길이를 최소화하게 간선을 연결하여 트리를 만들어야 합니다.
- 즉, 최소 스패닝 트리 문제로 바뀌게 됩니다.
- N개의 정점에서 만들 수 있는 $\frac{N(N-1)}{2}$개의 간선을 모두 만든 후, 이 간선들로 만든 최소 스패닝 트리에서 간선의 가중치의 합이 필요한 벨트의 총 길이가 됩니다.

D. 더 흔한 타일 색칠 문제

ad_hoc
출제진 의도 – **Easy**

- 제출 380번, 정답 190팀 (정답률 50.526%)
- 처음 푼 팀: **당신을 대신해 UCPC 팀명을 정해드릴게요** (프로그래밍 용사, 디버그의 달인, 컴파일의 지배자), 5분
- 출제자: bnb2011

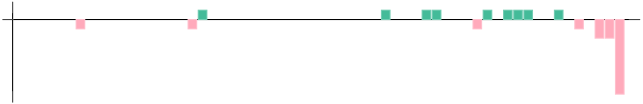


- $K \times K$ 크기로 가능한 타일의 수는 26^{K^2} 개로 너무 많습니다.
- 따라서 한 타일의 색상 배치를 고정해두고, 이를 모든 $K \times K$ 크기의 타일에 적용하는 식으로는 풀이를 전개하기 어렵습니다.
- 대신 $K \times K$ 타일에서 **같은 상대적 위치**에 있는 칸들은 모두 색상이 동일해야 한다는 점을 이용할 수 있습니다. 즉 $(i \bmod K, j \bmod K)$ 가 같은 칸들의 색상이 모두 동일해야 합니다.
- 같은 상대적 위치에 어떤 문자들이 몇 개 있는지 센 뒤, 그 중 가장 많이 등장한 문자로 타일을 다시 칠해주면 최소한의 색상을 사용할 수 있습니다. 시간복잡도는 $O(NM)$ 입니다.

E. 반전수

combinatorics, linearity_of_expectation
출제진 의도 – **Hard**

- 제출 26번, 정답 9팀 (정답률 34.615%)
- 처음 푼 팀: **DDT** (두, 둥, 탁), 55분
- 출제자: kclee2172



풀이 1

- $1, 2, \cdots, N$ 으로 이루어진 수열 p 가 존재할 때, 이 수열의 반전수 I 는 다음과 같이 작성할 수 있습니다.
  $I = X_1 + X_2 + \cdots + X_N$, 여기서 X_i 는 i 에 의하여 발생하는 반전수로, $s < t$ 면서 $p_s > i, p_t = i$ 를 만족하는 (s, t) 쌍의 개수를 의미합니다.
- 여기서, 관찰을 해보면 X_i 를 이 문제의 $N = 2$ 인 특수한 경우로 생각할 수 있습니다.
  원래 수열 p 에서 i 이상인 수들만 모은 부분수열을 생각해봅시다. 그리고, 그 부분수열에서 i 보다 큰 값을 2, i 를 1로 치환해줍니다. 그러면, 원래 수열에서 $s < t$ 면서 $p_s > i, p_t = i$ 를 만족하는 (s, t) 쌍의 개수와 부분수열에서 $s < t$ 면서 $p_s = 2, p_t = 1$ 인 (s, t) 쌍의 개수는 같게 되며, 이 값은 1 이 a_i 개, 2 가 $a_{i+1} + a_{i+2} + \cdots + a_N$ 개인 수열의 반전수와 같게 됩니다.

- $Y(n,m)$를 n개의 1, m개의 2로 이루어진 수열의 반전수로 정의를 하겠습니다. 그러면,
  $X_N = 0,\ X_{N-1} = Y(a_{N-1}, a_N),\ X_{N-2} = Y(a_{N-2}, a_{N-1}+a_N), \cdots, X_1 = Y(a_1, a_2+a_3+\cdots+a_N)$
  이 성립합니다.
- 또한, X_i 들은 독립적입니다. 즉, $\mathbb{E}(X_iX_j) = \mathbb{E}(X_i)\mathbb{E}(X_j)$가 성립합니다. 그러므로,
  $\mathbb{E}(I^2) = \mathbb{E}(X_1 + \cdots + X_N)^2$은 다음과 같은 귀납과정으로 계산할 수 있습니다.
- $\mathbb{E}(X_1 + \cdots + X_{i-1}), \mathbb{E}(X_1 + \cdots + X_{i-1})^2$을 알고 있다고 가정하겠습니다. 이때, 다음이 성립합니다.
  $\mathbb{E}(X_1 + \cdots + X_i) = \mathbb{E}(X_1 + \cdots + X_{i-1}) + \mathbb{E}(X_i)$
  $\mathbb{E}(X_1 + \cdots + X_i)^2 = \mathbb{E}(X_1 + \cdots + X_{i-1})^2 + 2\mathbb{E}(X_1 + \cdots + X_{i-1})\mathbb{E}(X_i) + \mathbb{E}(X_i^2)$
- 결론적으로, $\mathbb{E}(X_i)$와 $\mathbb{E}(X_i^2)$을 계산할 수 있다면, 이 문제를 풀 수 있습니다.

- $X = Y(n,m)$ 이라고 가정을 하겠습니다. 이때, 다음이 성립합니다.

$$\mathbb{E}(X) = \mathbb{E}(\sum_{i<j} \mathbb{1}_{p_i > p_j}) = \sum_{i<j} \mathbb{E}(\mathbb{1}_{p_i > p_j}) = \sum_{i<j} \mathbb{E}(\mathbb{1}_{p_i = 2, p_j = 1})$$

$$= \sum_{i<j} \frac{(n+m-2)!}{(n-1)!(m-1)!} \times \frac{n!m!}{(n+m)!} = \frac{nm}{2}$$

비슷하게 계산을 해보면,

$$\mathbb{E}(X^2) = \mathbb{E}((\sum_{i<j} \mathbb{1}_{p_i > p_j})^2) = \sum_{i_1 < j_1, i_2 < j_2} \mathbb{E}(\mathbb{1}_{p_{i_1} = 2, p_{j_1} = 1, p_{i_2} = 2, p_{j_1} = 1})$$

$$= \frac{n(n-1)m(m-1)}{4} + \frac{nm^2 + n^2 m - 2nm}{3} + \frac{nm}{2}$$

UCPC 2023

풀이 2

- 문제의 조건을 만족하는 수열을 $p_1, p_2, ..., p_s$ ($s = \sum_{i=1}^{n} a_i$) 라고 할 때 반전수의 기댓값은 $\mathrm{E}[(\sum_{i<j} \mathbb{1}_{p_i > p_j})^2]$ 로 나타낼 수 있습니다.
- 해당 식을 기댓값의 선형성을 이용해서 정리할 경우 a_i에 대한 4차 이하의 다항식이 나옴을 확인할 수 있습니다.
- 특정 k에 대해서 a_k와 a_{k+1}을 교환할 경우 답이 변하지 않습니다.
- 그 이유는, 수열에서 모든 k와 $k+1$을 각각 $k+1$, k로 바꾼 뒤에 모든 k와 $k+1$의 위치를 서로 뒤집을 경우 반전수가 변하지 않기 때문입니다.

- 즉, 반전수의 제곱의 기댓값은 a_i 들의 차수가 4 이하인 대칭다항식이 됨을 알 수 있습니다.
- $s_k = \sum_{i=1}^{n} a_i^k$ 라고 할 때, 반전수의 제곱의 기댓값은 상수 t_{k_1,k_2,k_3,k_4} 에 대해 $\sum_{k_1+2k_2+3k_3+4k_4 \le 4} t_{k_1,k_2,k_3,k_4} {s_1}^{k_1} {s_2}^{k_2} {s_3}^{k_3} {s_4}^{k_4}$ 의 형태로 나타낼 수 있습니다.
- 가능한 선형 독립인 항이 11개가 되고, 따라서 가능한 모든 수열의 반전수를 전부 계산하는 시뮬레이션 풀이를 이용해서 a_i 의 합이 충분히 작은 경우를 전부 계산하는 방식으로 t_{k_1,k_2,k_3,k_4} 를 찾을 수 있습니다.
- 시간복잡도는 $k \le 4$ 에 대해서 s_k 를 계산하는데 걸리는 시간 $O(n)$ 이 됩니다.

F. 응원단

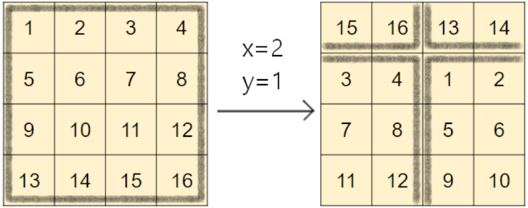
ad_hoc, implementation

출제진 의도 – **Hard**

- 제출 209번, 정답 55팀 (정답률 26.316%)
- 처음 푼 팀: **샤레롱은 미국갔어** (엉엉, 가지마, 군대도), 20분
- 출제자: golazcc83

F. 응원단

- 문제에서 주어진 응원 패턴의 홀짝성을 제외하고 풀이를 생각해봅시다.
  - 모든 단원들의 열을 1 증가시킨다. 열이 N을 초과한 단원은 동일한 행의 1열로 이동한다.
  - 모든 단원들의 행을 1 증가시킨다. 행이 N을 초과한 단원은 동일한 열의 1열로 이동한다.
- 각 단원의 초기 위치를 i행 j열이라고 했을 때, 응원 패턴을 모두 수행한 뒤 x행 y열을 움직였다면 최종 위치는 $((i+x-1) \mod N)+1$행 $((j+y-1) \mod N)+1$열입니다.



- 각 응원 패턴은 모든 단원의 위치를 수정하는 대신 x값, y값을 수정하는 것으로 $\mathcal{O}(1)$ 의 시간에 해결할 수 있습니다.
- 교체 패턴이 나올 때마다 모든 단원의 위치를 업데이트한 후 두 단원의 위치를 서로 바꿀 수 있으나, 모든 단원의 위치를 업데이트하는 데 $\mathcal{O}(N^2)$ 의 시간이 소요됩니다.
- 교체 패턴에서 주어지는 행, 열과 현재 시점에서의 x, y를 이용하여 두 단원의 초기 위치를 구할 수 있습니다. 두 단원의 초기 위치를 계산한 다음 초기 응원단의 상태에서 두 위치를 서로 바꿔주면 모든 단원의 위치를 업데이트할 필요가 없으므로 $\mathcal{O}(1)$ 의 시간에 해결할 수 있습니다.

UCPC 2023

- 이제 문제에서 주어진 응원 패턴의 홀짝성을 고려하면서 풀이를 생각해봅시다.
  - RO, RE : 행이 홀수 또는 짝수인 단원들의 열을 1 증가시킨다.
  - CO, CE : 열이 홀수 또는 짝수인 단원들의 행을 1 증가시킨다.
  - $S\ r_1\ c_1\ r_2\ c_2$: 두 단원의 위치를 서로 바꾼다.
- RO, RE, CO, CE에서 단원의 행, 열을 증가시키는 기준은 행과 열의 홀짝성입니다.
- 각 단원을 4개의 그룹으로 묶어서 생각해봅시다. 각 단원의 초기 위치가 i행 j열이면

1. i가 홀수, j가 홀수인 단원
2. i가 짝수, j가 홀수인 단원
3. i가 홀수, j가 짝수인 단원
4. i가 짝수, j가 짝수인 단원

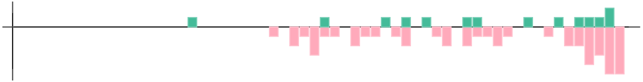
- 4개의 그룹에 아래의 항목을 관리해줍니다.
  - 현재 행의 홀짝성, 현재 열의 홀짝성
  - 초기 상태로부터 이동한 행의 값 x, 초기 상태로부터 이동한 열의 값 y
- 위에서 관리한 항목을 이용하면 모든 응원 패턴을 $\mathcal{O}(1)$ 의 시간에 해결할 수 있습니다.
  - RO, RE : 현재 행의 홀짝성이 일치하는 그룹 2개의 y를 1 증가시키고 현재 열의 홀짝성을 반전시킵니다.
  - CO, CE : 현재 열의 홀짝성이 일치하는 그룹 2개의 x를 1 증가시키고 현재 행의 홀짝성을 반전시킵니다.
  - $S\ r_1\ c_1\ r_2\ c_2$: (r_1, c_1) 의 현재 행과 열의 홀짝성이 일치하는 그룹을 찾습니다. 찾은 그룹의 x, y 값을 이용하여 초기 위치를 계산합니다. (r_2, c_2) 도 동일한 과정으로 초기 위치를 계산합니다. 초기 응원단의 상태에서 두 위치를 서로 바꿉니다.

- 모든 응원 패턴을 수행했다면, 4개의 그룹에서 관리한 항목들로 응원단의 최종 상태를 계산할 수 있습니다.
- 각 그룹 별 단원의 초기 위치가 i행 j열이고 응원 패턴을 모두 수행한 뒤 x행 y열을 움직였다면 최종 위치는 $((i+x-1) \mod N)+1$행 $((j+y-1) \mod N)+1$열입니다.
- 최종 시간복잡도는 $\mathcal{O}\left(N^2+Q\right)$ 입니다.

G. 은하 온라인 마케팅 프로젝트

greedy, sorting, data_structures, offline_query
출제진 의도 – **Hard**

- 제출 63번, 정답 14팀 (정답률 22.222%)
- 처음 푼 팀: **김앤장** (azberjibiou, maruii, etyu), 52분
- 출제자: asdarwin03, 99asdfg



- N개의 각 국가별로 광고할 도시를 하나씩 골랐을 때 나올 수 있는 모든 N개 도시 조합을 전부 확인하면 M^N개의 경우를 모두 확인해야 합니다. 문제를 시간 제한 내에 해결하기 위해서는 확인할 도시 조합의 개수를 줄여야 합니다.
- 임의의 $A_{i,j}$ 값을 최대 유입 수로 가지는 도시 조합들 중 최적의 도시 조합은 각 국가별로 $A_{i,j}$ 이하이면서 최댓값을 가진 도시를 하나씩 골라 도시 조합을 구성하는 것으로, 문제 상황에서 $A_{i,j}$에 대해 항상 최적인 도시 조합을 찾아낼 수 있습니다.
- 그 도시 조합은 이벤트를 통해 유입시킬 유저 수를 고려하더라도 $A_{i,j}$를 최대 유입 수로 가지는 모든 도시 조합 중 유입시킨 유저 수의 총합을 최대로 가지며, 동시에 최소 불균등도를 가질 수 있습니다. 이는 증명할 수 있습니다.
- 따라서 각 $A_{i,j}$에 대해 이러한 최적의 도시 조합을 구성하면 확인해야 할 도시 조합의 개수를 NM개 이하로 줄일 수 있습니다.

- 이제 오프라인 이벤트를 개최할 도시와, 이를 통해 유입시킬 적절한 유저 수를 생각해야 합니다.
- 임의의 $A_{i,j}$ 에 대해 최적의 도시 조합을 이루는 도시들 중 광고를 통해 유입시킬 유저 수가 가장 작은 도시에서 오프라인 이벤트를 개최하는 것이 항상 최적이며, 이는 증명할 수 있습니다.
- 이벤트를 통해 유입시킬 유저 수를 결정하기 위해 각 도시 조합에서 고려해야 할 대푯값은 총 4 개로, 광고를 통해 유입시킬 유저 수들의 총합, 그 중 가장 작은 값, 두번째로 작은 값, 가장 큰 값입니다. 앞으로 이를 각각 s, a, b, c 라 하겠습니다.
- 임의의 도시 조합에서 오프라인 이벤트를 개최해 $t(0 \le t \le C)$ 만큼의 유저를 추가로 유입시킨다고 했을 때, 광고와 이벤트를 통해 유입시킬 유저 수의 총합의 목표량 P 와 4 개의 대푯값에 따라 최적의 t 값이 달라집니다.
- P 가 주어졌을 때 한 도시 조합에서 최적의 t 값은 $P \le s + t$ 를 만족시키면서 $a + t$ 가 b 에 최대한 가깝도록 하는 t 값입니다.

- 따라서 이벤트로 유입시킬 유저 수의 총합 $t(0 \le t \le C)$ 에 따른 불균등도 $f(t)$ 는 다음과 같이 나타낼 수 있습니다.
- $$f(t) = \begin{cases} c-(a+t) & \text{if } a+t<b \\ c-b & \text{if } b \le a+t \le c \\ a+t-b & \text{if } c<a+t \end{cases}$$
- 이전에 t 값에 따른 불균등도 $f(t)$ 는 다음과 같이 P 값과 도시 조합의 대푯값 s, a, b, c 에 따라 최적의 t 값이 달라질 수 있다고 했는데, 이를 좀 더 구체적으로 생각해 봅시다.
- 임의의 P 에 대해, $P \le s+t$ 를 만족시키는 t 중 $f(t)$ 의 최솟값을 계산하기 위해 두 경우로 분리해 생각할 수 있습니다.

1. $P \le c - a + s$일 때 최소 $f(t)$는 $t = \min(a + C,\ b - a)$인 경우입니다.
2. $c - a + s < P \le s + C$일 때 최소 $f(t)$는 $t = P - s$인 경우입니다. 이 경우는 $c - a < C$일 때에만 존재할 수 있습니다.

- 한 도시 조합에 대해 광고와 이벤트를 통해 유입시킬 유저 수의 총합을 x라 하고, y를 x에 대해 만들 수 있는 최소 불균등도라 정의하면 가능한 x의 구간 $[s, s + C]$에 대해 위 내용을 다음과 같이 다시 쓸 수 있습니다.
- $y = \begin{cases} f(\min(a + C, b - a)) & \text{if } s \le x \le \min(s + C,\ c - a + s) \\ f(x - s) & \text{if } c - a + s < x \le s + C \end{cases}$
- 임의의 P값에 대해 $P \le x$을 만족하는 x를 만들 수 있는 모든 도시 집합의 y값 중 최솟값이 실제 마케팅을 진행할 도시 조합의 불균등도입니다.

- Q개의 P값에 대한 최소 불균등도를 구하기 위해 각 도시 조합의 유효한 구간을 고려해야 하며, 이때 정렬과 오프라인 쿼리를 활용할 수 있습니다.
- $s \le x \le \min(s+C, c-a+s)$를 만족하는 각 도시 조합의 x에 대해, $P \le s$만 만족하면 $0 \le P < s$인 P값에 대해서도 그 도시 조합의 최소 불균등도 $f(\min(a+C,\ b-a))$를 사용할 수 있습니다. 따라서 해당 경우에 대해 Q개의 P값을 내림차순 정렬하고 $\min(c-a+s,\ s+C)$값이 큰 순서대로 모든 도시 조합을 탐색하여 최솟값을 갱신 및 기록하는 풀이를 적용할 수 있습니다.
- $c-a+s < x \le s+C$를 만족하는 각 도시 조합의 x에 대해서는 임의의 P에 대해 $x = P$일 때 최소 불균등도를 구할 수 있습니다. 하지만 앞선 경우와 달리 P의 값에 따라 y의 값$(= f(P-s))$이 달라지므로 P가 여러개 주어질 때 하나의 고정된 불균등도 값 y를 사용할 수 없습니다.
- 하지만 이 경우에서 x가 증가할때마다 y가 항상 일정하게 증가하는 성질을 활용하면 이 문제를 해결할 수 있습니다.

- 임의의 P값에서의 y값은 $P-(c-a+s)+f(c-a)$로 나타낼 수 있으므로 y값 대신 $c-a+s$값과 $f(c-a)$값을 저장해두면 각 P에 대해 이를 계산해서 최소 불균등도를 계산할 수 있습니다.
- 이때 각 도시 조합의 x와 y는 항상 일정하게 증가하므로 $c-a+s<P\leq s+C$를 만족시키는 모든 도시 조합에 대해 $c-a+s-f(c-a)$가 가장 큰 도시 조합의 불균등도가 최소 불균등도임은 자명합니다.
- xy평면에 여러 도시 조합의 x에 대한 불균등도 y를 모두 그려서 비교해보면 이를 직관적으로 이해할 수 있습니다.
- 해당 경우의 유효 구간은 각 도시 조합마다 $(c-a+s, s+C]$임에 유의합시다.
- $c-a+s<P\leq s+C$를 만족하는 모든 도시 조합 중 최대 $c-a+s-f(c-a)$를 가진 도시 조합을 고르기 위해 우선순위 큐나 multiset을 활용할 수 있습니다.

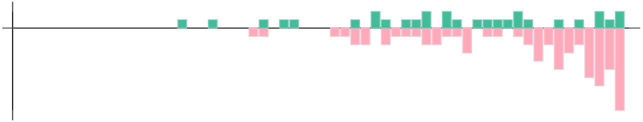
- 최종 시간복잡도는 $\mathcal{O}\left((NM+Q)logNM+QlogQ\right)$ 입니다.
- 별해로 세그먼트 트리를 이용하여 각 쿼리를 처리하는 풀이가 있습니다.

H. 팔찌

`ad_hoc, simulation`

출제진 의도 – **Hard**

- 제출 103번, 정답 30팀 (정답률 29.126%)
- 처음 푼 팀: **R3** (Red, Ruby, Red), 49분
- 출제자: `wider93`



H. 팔찌

- 팔찌로서 뒤집고 돌려도 같다는 조건을 잠시 뒤로 제쳐둡시다.
- 구슬을 색에 따른 변수 x, y, z로 생각합시다. 팔찌의 이어붙임은 (결합법칙이 성립하는) 곱셈으로 이해할 수 있습니다.
- 주어진 조작은 다음과 같은 등식으로 요약됩니다.
  - $xy = z = yx$
  - $yz = x = zy$
  - $zx = y = xz$
- 임의의 두 변수 a, b에 대해 $ab = ba$임을 관찰합니다($a = b$일 때 포함). 따라서, 교환법칙이 성립합니다.
- 모든 팔찌는 $x^a y^b z^c$로 표현됩니다. 즉, 각 구슬 색의 개수가 같으면 같은 팔찌입니다.
- 따라서 뒤집고 돌리는 조작을 추가해도 팔찌의 동일 유무에 실제로 의미가 없음을 알 수 있습니다.

H. 팔찌

- $[x, y, z]$의 순열 $[a, b, c]$에 대해, $a^2bc = (ab)(ac) = cb = a$를 만족합니다.
- 따라서 xyz는 (팔찌가 xyz인 경우를 제외하면) 소거될 수 있는 식입니다.
- $x^2 = y^2 = z^2 = xyz$에도 주목합시다.
- 가능한 길이 2 이하의 팔찌는 $x^2 = y^2 = z^2$, $xy = z$, $yz = x$, $zx = y$의 4종류가 전부입니다.
- 길이 3 이상의 모든 팔찌는 길이를 줄일 수 있는 식을 포함하고 있으므로 위 4종류 중 하나로 만들 수 있습니다.
- 위 4종의 팔찌가 다르다는 것은, $x = 1, y = 2, z = 3$을 대입하고 곱셈을 xor로 대신한 값이 팔찌의 변환에 대한 불변량이라는 점으로 확인할 수 있습니다.

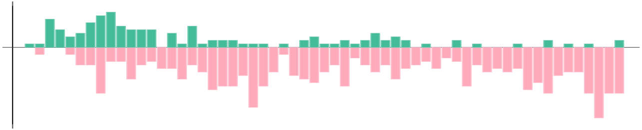
H. 팔찌

- 이제 같은 팔찌를 실제로 동일하게 만드는 법을 제시해야 합니다.
- 입력으로 제시된 두 팔찌를 각각 x, y, z, x^2 중 하나로 축소하면서, 축소과정과 그 역과정을 하나씩 출력하면 답이 됩니다.
- $a^n b \to a^{n-1} c \to a^{n-2} b \to \cdots$ 를 적절히 활용하면, $O(n)$ 횟수의 조작으로 길이 n의 팔찌를 위 넷 중 하나로 축소할 수 있습니다.
- 실제로 시뮬레이션할 때는 삽입·삭제를 $O(n)$에 수행하는 $O(n^2)$ 풀이를 허용하도록 제한이 주어져 있습니다.

I. 자석

math, prefix sum
출제진 의도 – **Medium**

- 제출 573번, 정답 125팀 (정답률 21.990%)
- 처음 푼 팀: **티어가 등차수열** (다이아 1, 다이아 4, 플래티넘 2), 4분
- 출제자: evenharder



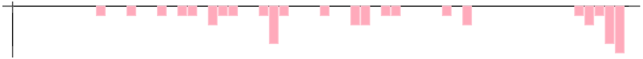
- i 번 칸에 N극이, j 번 칸에 S극이 놓이도록 자석을 설치하면 $a_i - a_j - K|i - j|$ 의 에너지를 충전할 수 있습니다.
- $i > j$ 라고 가정하면 식이 $a_i - a_j - K(i - j) = (a_i - Ki) - (a_j - Kj)$ 로 표현됩니다.
- i 를 2부터 N 까지 증가시키면서, $1 \le j < i$ 를 만족하는 j 중 $a_j - Kj$ 의 최솟값을 구하면 됩니다. 누적 최솟값을 이용하면 모든 i 에 대해 $O(N)$ 에 계산할 수 있습니다.
- $i < j$ 일 때는 수열을 거꾸로 뒤집어서 $i > j$ 일 때처럼 풀면 되고, 이 중 최댓값이 답입니다.

J. 다섯 용사의 검

`dp, bitmask, two_pointer`

출제진 의도 – **Challenging**

- 제출 43번, 정답 0팀 (정답률 0.000%)
- 처음 푼 팀: –
- 출제자: queued_q



가장 먼저 공격력을 좌표 압축하고 시작합니다. 서로 다른 공격력의 수를 N으로 둡시다.

m의 단단함을 가진 바위로 각 검을 테스트해 보면, 바위에 균열을 낼 수 있는 검 (이하 강한 검)과 그렇지 않은 검 (이하 약한 검)으로 나뉩니다.

결과에 따라 다음과 같이 최고의 검 후보와 공격력 범위를 줄여나갈 수 있습니다.

- 모든 검이 약한 검인 경우, 최고의 검은 m 이하의 공격력을 가집니다.
- 강한 검이 하나라도 존재하는 경우, 약한 검은 더 이상 후보로 고려할 필요가 없으며 최고의 검은 $m+1$ 이상의 공격력을 가집니다.

UCPC 2023

Naive DP

최고의 검 후보 집합이 S이며 공격력 범위가 $[l, r]$임을 알 때 필요한 테스트 횟수를 $D[S, l, r]$이라고 합시다.

m의 단단함을 가진 바위로 검을 테스트했을 때,

- 모든 검이 약한 검인 경우, $[l, m]$ 구간에서 검 집합 S로 테스트를 계속합니다.
- 강한 검이 하나라도 존재하는 경우, $[m+1, r]$ 구간에서 검 집합 $T \subseteq S$로 테스트를 계속합니다.
  - 최악의 경우를 따져야 하므로 테스트 횟수가 가장 많은 집합 T를 고려해야 합니다.

이를 정리하면 다음과 같은 DP 관계식을 세울 수 있습니다.

$$D[S,l,r] = \max_{l \le m \le r}\left(D[S,l,m],\ \max_{T \subseteq S} D[T,m+1,r]\right) + 1$$

물론 검을 테스트하지 않고도 최고의 검을 알 수 있는 경우가 있습니다.

- 어떤 검 $i \in S$의 최소 공격력이 S 안의 다른 모든 검의 최대 공격력보다 높다면, 검 i는 최고의 검입니다.
- 이러한 경우에는 DP 값을 0으로 설정해 줍니다.

참고로, $T \subseteq S \subseteq \{1, \ldots, n\}$ 을 만족하는 (S, T) 쌍을 순회하는 것은

- 평범하게 구현하면 $O(2^n \times 2^n) = O(4^n)$ 이지만
- 다음과 같이 구현하면 T 가 정확히 S 의 부분집합만 순회하므로 이항 정리에 따라 $O(3^n)$ 의 시간이 걸립니다.

```
for S in [0, 2**5):
    T = S
    while T > 0:
        visit(S, T)
        T = (T-1) & S
    visit(S, 0)
```

DP 최적화

앞에서 설명한 DP 식을 그대로 구현할 경우 시간복잡도는 $O(3^5 N^3)$ 로 매우 느립니다. 이를 최적화하기 위해 몇 가지 관찰이 필요합니다.

- **관찰 1.** DP 값은 최대 $\lceil \lg N \rceil$ 이다.
  - 간단한 이분 탐색을 통해 최고의 검을 찾을 수 있으므로, 필요한 테스트 횟수는 그 이하입니다.
- **관찰 2.** 두 구간이 포함관계에 있다면, 넓은 구간에 대한 답이 더 크거나 같다.
  - 넓은 구간에서 검을 테스트하는 최적의 전략을 좁은 구간에 적용해도 동일하게 최고의 검을 찾을 수 있기 때문입니다.

앞선 관찰들에 따르면 다음과 같은 성질을 알 수 있습니다.

- **성질 1.** 구간 $[l, r]$에 대한 DP 값은 r이 증가함에 따라 최대 $\lceil \lg N \rceil$까지 증가한다.
- **성질 2.** $D[S, l, r] = k$일 때, $[l, r]$ 구간에서 $D[S, l, m] = k - 1$을 만족하는 최대의 m을 선택해서 검을 테스트하는 것이 최적의 전략이다.
  - m을 가능한 한 오른쪽으로 보내더라도 오른쪽 구간에서 필요한 테스트 횟수는 증가하지 않기 때문입니다.

성질 1에 따르면, S와 l을 고정했을 때, 모든 r에 대한 DP 값을 찾는 대신 DP 값이 증가하는 위치를 계산하는 것이 효율적입니다.

k회 이하의 테스트로 최고의 검을 찾을 수 있는 최대의 r을 $E[k, S, l]$이라고 합시다. 그러면 **성질 2**에 따라 다음과 같은 DP 관계식이 나옵니다.

$$E[k, S, l] = \min_{T \subseteq S} E[k-1, T, m+1] \quad \text{(where } m = E[k-1, S, l]\text{)}$$

DP 배열을 채우는 데 드는 시간은 $O(3^5 N \log N)$으로, 제한 시간 안에 충분히 실행됩니다. DP 배열을 채운 뒤 $E[k, S, 1] = N$이 되는 최소의 k를 찾으면 답이 됩니다.

초깃값 계산

DP 관계식을 찾았으니 초깃값, 즉 $E[0, S, l]$ 을 계산하는 일이 남았습니다. 다음과 같은 성질을 관찰합시다.

- **성질 3.** $l_1 \le l_2$ 이면 $E[k, S, l_1] \le E[k, S, l_2]$ 이다.
  - $r = E[k, S, l_1]$ 이라고 하면, 구간 $[l_1, r]$ 안에 구간 $[l_2, r]$ 이 포함됩니다.
  - $[l_1, r]$ 에서 필요한 테스트 횟수가 k 이므로, 구간 $[l_2, r]$ 에서 필요한 테스트 횟수는 k 이하입니다.
  - 따라서 $E[k, S, l_2]$ 는 적어도 r 이상입니다.

단조성을 활용하기 위해 투 포인터 기법을 사용할 수 있습니다. l을 1부터 N까지 증가시켜 가면서, 구간을 테스트하지 않아도 되는 동안 r을 늘려 나가면 $E[0, S, l]$을 구할 수 있습니다.

구간을 테스트하지 않아도 되는 상황이 무엇이었는지 다시 떠올려 봅시다.

- 어떤 검 $i \in S$의 최소 공격력이 S 안의 다른 모든 검의 최대 공격력보다 높다면, 검 i는 최고의 검입니다.
- 추가적으로, 구간 내에 S에 속한 검이 존재하지 않는 경우에도, 구간을 테스트하지 않아도 되는 상황으로 가정합니다. 실제로는 DP 값을 정의할 수 없겠지만, 이렇게 가정해야 단조성이 생겨서 계산이 편리합니다.

각 검마다 구간 내에서의 공격력을 큐 등 적절한 자료 구조로 관리하면, 테스트가 필요 없는 상황인지 빠르게 알 수 있습니다.

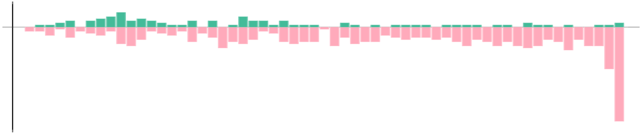
이 과정의 시간복잡도는 $O(2^5 \cdot 5N)$ 입니다. 따라서 전체 시간복잡도는 $O(3^5 N \log N)$ 으로 제한 시간 안에 문제를 해결할 수 있습니다.

K. 세미나 배정

`binary_search, greedy`

출제진 의도 – **Medium**

- 제출 495번, 정답 88팀 (정답률 17.778%)
- 처음 푼 팀: **16** (졸업, 하고, 싶어요), 6분
- 출제자: puppy



- 두 조건을 만족하는 세미나 배정 방안을 '유효한 세미나 배정'이라고 합시다.
- 답을 P라 합시다.
- P 미만의 임의의 수 c에 대해, 하루에 진행되는 세미나 수의 최댓값이 c 이하가 되는 유효한 세미나 배정은 없습니다.
- 임의의 c에 대해 모든 날에 c개 이하의 세미나를 진행하는 유효한 세미나 배정이 존재하는지 판별할 수 있다면, 이분 탐색을 통해 풀 수 있습니다.

- a_i 가 작은 세미나일수록 더 빠른 날에 진행된다고 가정해도 무관합니다.
- 즉, $a_i < a_j$ 이면 $m_i \le m_j$ 라고 가정해도 무관합니다. 증명은 다음과 같습니다.
  - 어떤 유효한 세미나 배정에서 $a_i < a_j$ 인데 $m_i > m_j$ 인 i, j 가 존재한다고 가정합시다.
  - i 번째 세미나와 j 번째 세미나의 일정을 서로 바꾸어도 세미나 배정은 여전히 유효합니다.
  - 이렇게 각 날에 진행되는 세미나의 수를 유지하며 $m_i < m_j$ 가 되게 할 수 있습니다. 따라서 $a_i < a_j$ 이면 $m_i \le m_j$ 라고 가정해도 됩니다.

- 모든 날에 c개 이하의 세미나를 진행하는 유효한 세미나 배정이 존재하는지 판별해봅시다.
- 먼저 a를 비내림차순으로 정렬합니다. 이건 처음 a를 입력받고 한 번만 하면 됩니다. 이제부터 i 번째 세미나는, 비내림차순으로 정렬된 a 상에서 i 번째인 세미나를 의미합니다.
- a가 작은 것부터 차례로 배정합니다. 이때, 모든 날에 c개 이하의 세미나를 진행해야 한다는 조건과 a_i 일차를 포함해야 한다는 조건을 만족하는 가장 이른 날짜에 배정합니다.

- 구체적으로, i번째 세미나를 배정할 날짜는 다음과 같이 정하면 됩니다.
  - $i \le c$일 경우, 지금까지 배정된 세미나가 c개 미만이므로 모든 날에 c개 이하의 세미나를 진행해야 한다는 조건을 무시해도 됩니다. 따라서 $max(a_i - T + 1, 1)$ 일차에 배정합니다.
  - $i > c$일 경우, $i - c$번째 세미나가 끝난 이후에 배정해야 합니다. 따라서 $max(a_i - T + 1, 1, (i - c)$번째 세미나 시작 날짜$+ T)$ 일차에 배정합니다.
- 이렇게 모든 세미나를 배정했을 때 모든 i에 대해 i번째 세미나의 진행 기간이 a_i 일차를 포함한다면 가능, 아니면 불가능입니다.

- 가능한 답의 범위는 1 이상 N 이하입니다. 이렇게 이분탐색을 하여 답을 구할 수 있습니다.
- 정렬은 처음에 1회만 하면 됩니다. 정렬을 제외한 판별은 $O(N)$에 할 수 있습니다.
- 최종 시간복잡도는 $O(N \log N)$입니다.