2026 숭고한 연합 알고리즘 경진대회

Official Solutions

주최 · 주관

숭실대학교 SCCC

고려대학교 ALPS · AIKor · MatKor

한양대학교 ALOHA

2026년 9월 19일

후원사



후원사



대회 운영진



숭실대학교 SCCC

- ✓ 조문성 `jms020820`
- ✓ 양성준 `egod1537`
- ✓ 최도현 `dohyonne`
- ✓ 노재원 `ian0704`
- ✓ 심기찬 `strain7626`

대회 운영진



고려대학교 ALPS · AlKor · MatKor

- ✓ 이승현 `kuming19`
- ✓ 임동현 `dong0625`
- ✓ 김기용 `kky085437`
- ✓ 하준서 `mathjuny`
- ✓ 이민재 `leeminjae`
- ✓ 송준민 `sjoonmin07`
- ✓ 이지훈 `ezh8n`
- ✓ 오서준 `osjoon`

대회 운영진



한양대학교 ALOHA

- ✓ 김태우 `mathtw1030`
- ✓ 노경민 `gmroh06`
- ✓ 김준호 `gubshig`
- ✓ 권용환 `joonyury`
- ✓ 이성현 `hibye1217`

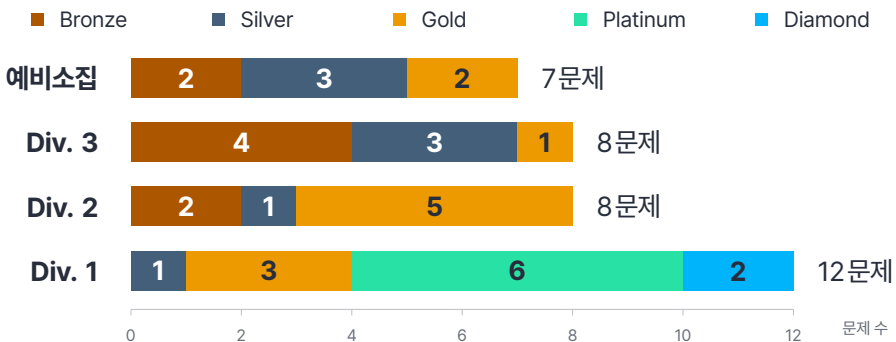
대회 검수진



- ✓ 0xchaser
- ✓ gs20036
- ✓ ibasic
- ✓ ishs311317
- ✓ jhnah917
- ✓ jthis
- ✓ mythofys
- ✓ petamingks
- ✓ realpsdoingdamyoo
- ✓ rlatjwls3333
- ✓ shinhuichan
- ✓ sorohue
- ✓ tlsdydaud1
- ✓ wyverns077
- ✓ yong0315

난이도 분포

solved.ac 기준 예상 난이도



예비소집 문제 목록



문제		의도한 난이도	출제자
A	15 + 15 = 50	**Easy**	이성현 hibye1217
B	반복올림	**Easy**	이성현 hibye1217
C	메모리 카드 게임	**Medium**	김태우 mathtw1030 조문성 jms020820
D	ORbit	**Medium**	이지훈 ezh8n
E	숭배하라, 고귀한 한 사람을.	**Hard**	이성현 hibye1217
F	선 연결하기	**Hard**	이승현 kuming19
G	GCD/LCM	**Challenging**	조문성 jms020820

Div. 3 문제 목록

문제		의도한 난이도	출제자
A	정12각형	Easy	이성현hibye1217
B	MOBIS 프로젝트 신청하기	Easy	최도현dohyonne
C	std.IO	Easy	이성현hibye1217
D	퍼리와 쿼리	Furry	이지훈ezh8n
E	돌려돌려돌림판	Medium	노경민gmroh06
F	색종이 자르기	Hard	조문성jms020820
G	마검의 조율	Hard	송준민sjoonmin07
H	제트팩	Challenging	김기용kky085437

Div. 2 문제 목록

문제		의도한 난이도	출제자
A	MOBIS 프로젝트 신청하기	Easy	최도현 dohyonne
B	돌려돌려돌림판	Easy	노경민 gmroh06
C	실버 문제 만들기	Medium	이승현 kuming19
D	대회 문제 구성하기	Hard	이승현 kuming19
E	자유시간권	Hard	이승현 kuming19
F	미로 탈출하기	Hard	김태우 mathtw1030 조문성 jms020820
G	별 배열	Challenging	양성준 egod1537
H	먼지 필터	Challenging	최도현 dohyonne

Div. 1 문제 목록 (A–F)

문제		의도한 난이도	출제자
A	윈드 쉴드 미러 디스플레이	**Hard**	이성현hibye1217
B	직사각형 고리 줍기	**Easy**	조문성jms020820
C	Sumptuous K-th Highway	**Hard**	최도현dohyonne
D	고양이 빚기	**Furry**	오서준osjoon
E	구 구조 구하기	**Hard**	이지훈ezh8n
F	숭고한 만들기	**Medium**	김기용kky085437

Div. 1 문제 목록 (G–L)

문제		의도한 난이도	출제자
G	마석 나누어가지기	Challenging	이성현 hibye1217
H	가장 긴 증가하는 모든 부분 수열	Medium	이성현 hibye1217
I	Simple Permutation Problem	Hard	조문성 jms020820
J	Do You Love Geometry?	No!	이성현 hibye1217
K	주식 거래	Challenging	이승현 kuming19
L	트리 만들기	Hard	하준서 mathjuny

0A. 15 + 15 = 50

implementation, arithmetic
출제진 의도 – **Easy**

✓ 출제자: 이성현hibye1217

0A. 15 + 15 = 50



✓ $L_i + R_i + 20$을 출력하면 됩니다.



0B. 반복올림

implementation, math
출제진 의도 – **Easy**

✓ 출제자: 이성현hibye1217

0B. 반복올림



- ✓ 반올림의 결과가 내림이면 반올림을 적용할 필요가 없습니다.
- ✓ 반올림의 결과가 올림이면 반올림을 적용하면 됩니다.
- ✓ 이를 뒤쪽 자리에서부터 차례로 적용하면 됩니다.

0B. 반복올림



✓ $0.444\ldots$ 초과인지 판별해도 됩니다.



0C. 메모리 카드 게임

`interactive, greedy`
출제진 의도 – **Medium**

✓ 출제자: 김태우`mathtw1030` 조문성`jms020820`

0C. 메모리 카드 게임



- ✓ 이미 확인했지만 아직 사라지지 않은 카드의 번호를 값별로 저장합니다.
- ✓ **값이 같은 카드 두 장을 알고 있다면** 그 두 장을 이번 턴의 첫 번째, 두 번째 카드로 골라 제거합니다.
- ✓ 그런 짝이 없다면 아직 확인하지 않은 카드 x를 `first`로 고릅니다.
  - x와 같은 값을 가진 기존 카드가 있으면 그 카드를 `second`로 골라 제거합니다.
  - 없으면 아직 확인하지 않은 다른 카드 y를 `second`로 고릅니다. 두 답이 같으면 제거하고, 다르면 둘 다 기억합니다.
- ✓ 모든 카드가 사라질 때까지 반복합니다. 매 질문 뒤에는 출력 버퍼를 비웁니다.

0C. 메모리 카드 게임



- ✓ 짝을 알고 있으면 제거하므로, 새 카드를 고를 때는 저장된 카드 중 같은 값의 두 장이 없습니다.
- ✓ 턴이 실패했다면 `first`의 값은 이전에 확인한 적 없는 값입니다. 이미 알던 값이었다면 저장된 짝을 `second`로 골라 성공했기 때문입니다.
- ✓ **첫 실패 턴**에는 값이 다른 새 카드 두 장을 확인하므로 서로 다른 값을 두 개 발견합니다. 이후 실패 턴마다 새로운 값을 적어도 하나 발견합니다.

0C. 메모리 카드 게임



- ✓ 값은 총 N 종류이므로 실패 턴은 최대 $N-1$ 번입니다. 성공 턴은 정확히 N 번이므로 확인 횟수는 최대 $2(2N-1)=4N-2$ 번입니다.
- ✓ 이 증명은 아직 보지 않은 카드의 배치를 가정하지 않아 적응적 채점기에도 성립합니다. 시간과 공간은 모두 $\mathcal{O}(N)$ 입니다.

0D. ORbit

`math, string, implementation`

출제진 의도 – **Medium**

✓ 출제자: 이지훈ezh8n

0D. ORbit



- ✓ t번 확장한 문자열에서 1의 개수를 C_t, 두 비트 중 적어도 하나가 1인 인접 쌍의 개수를 P_t라고 합시다.
- ✓ OR 결과가 1인 쌍마다 1을 하나씩 삽입하므로

$$C_{t+1} = C_t + P_t.$$

- ✓ 인접 쌍 (a, b)는 $(a, a \vee b), (a \vee b, b)$라는 두 쌍으로 나뉩니다. 이들이 확장 후의 모든 인접 쌍을 중복 없이 이룹니다.
- ✓ 00에서는 두 쌍 모두 00이고, $01, 10, 11$에서는 두 쌍 모두 적어도 하나의 1을 포함합니다. 따라서 $P_{t+1} = 2P_t$입니다.

0D. ORbit



- ✓ $P_{t+1} = 2P_t$ 이므로 귀납적으로 $P_t = 2^t P_0$ 입니다.
- ✓ 원래 있던 문자는 사라지지 않고, 각 단계에서 새로 생기는 1의 개수는 P_t 입니다. 따라서 정답은

$$C_K = C_0 + \sum_{t=0}^{K-1} P_t = \boxed{C_0 + (2^K - 1)P_0}.$$

- ✓ 원래 문자열에서 1과 00이 아닌 인접 쌍을 세기만 하면 됩니다. 실제로 확장한 문자열을 만들 필요가 없습니다.
- ✓ $K = 0$이면 추가되는 항이 0이며, $N = 1$이면 $P_0 = 0$입니다. 모든 문자가 0인 경우도 같은 식으로 처리됩니다.

0D. ORbit



- ✓ 문자열을 왼쪽부터 읽습니다. 현재 문자가 1 이면 C_0 를, 직전 문자와 현재 문자의 OR가 1 이면 P_0 를 증가시킵니다.
- ✓ 2^K 는 64비트 정수의 시프트 1LL « K로 계산하고, $C_0 + (2^K - 1)P_0$ 를 출력합니다.
- ✓ 최종 길이는 $(N-1)2^K + 1$ 입니다. 제한 $N \le 200\,000$, $K \le 40$ 에서 정답은 최대 $219\,901\,226\,043\,572\,225$ 이므로 부호 있는 64비트 정수에 들어갑니다. 곱셈도 64비트로 수행해야 합니다.
- ✓ 시간 $\mathcal{O}(N)$, 추가 공간 $\mathcal{O}(1)$. 입력 문자열 전체를 저장하는 구현의 공간은 $\mathcal{O}(N)$ 입니다.



0E. 숭배하라, 고귀한 한 사람을.

ad_hoc, math
출제진 의도 – **Hard**

✓ 출제자: 이성현hibye1217

0E. 숭배하라, 고귀한 한 사람을.



- ✓ 사용해야 하는 연산 횟수의 하한부터 구해봅시다.
- ✓ $d_i = x_i - x_{i-1}$ 이라고 합시다.
- ✓ 각 연산은 0 초과인 d_i 만 남긴 뒤, 인접한 두 d_i 의 값을 1 내리는 연산으로 볼 수 있습니다.

OE. 숭배하라, 고귀한 한 사람을.



- ✓ 우선 하나의 d_i 를 완전히 없애야 하므로, 최소 $\max d_i$ 회의 연산이 필요합니다.
- ✓ 또한, 하나의 연산은 d_i 의 합을 최대 2 감소시키므로, 최소 $\left\lceil \sum d_i/2 \right\rceil$ 회의 연산이 필요합니다.
- ✓ 따라서, 최소 $\max(\max d_i, \left\lceil \sum d_i/2 \right\rceil)$ 회의 연산을 진행해야 합니다,

0E. 숭배하라, 고귀한 한 사람을.



- ✓ 이제 연산 한 번으로 저 값을 1 줄일 수 있음을 보입시다.
- ✓ 즉, 저 값이 도달 가능한 최소임을 보입니다.

0E. 숭배하라, 고귀한 한 사람을.



- ✓ d의 길이가 1이라면 자명합니다.
- ✓ d의 길이가 2 이상이라면 $\left\lceil \sum d_i/2 \right\rceil$은 자명히 1만큼 줄어듭니다.
- ✓ $\max d_i$를 골라서 연산을 진행하면 $\max d_i$도 1만큼 줄일 수 있습니다.
- ✓ 또는, $\max d_i$가 여럿이라서 $\left\lceil \sum d_i/2 \right\rceil > \max d_i$가 되어 원하는 값이 1 줄어듭니다.



0F. 선 연결하기

combinatorics, prefix_sum
출제진 의도 – **Hard**

✓ 출제자: 이승현kuming19

0F. 선 연결하기



- ✓ 전체 빨간 점과 파란 점의 수를 각각 R, B 라고 합시다.
- ✓ 그을 수 있는 선분의 개수는 가능한 모든 R, B 를 매칭시켜주는 것이므로 항상

$$P = \min(R, B)$$

을 만족합니다.

- ✓ 구간 $(i, i+1)$ 의 왼쪽에 있는 빨간색과 파란색 점의 수를 R_L, B_L, 오른쪽의 수를 R_R, B_R 라고 합시다.

0F. 선 연결하기



- ✓ 해당 구간을 지나는 선분은 빨간 점(왼쪽)과 파란 점(오른쪽), 또는 빨간 점(오른쪽)과 파란 점(왼쪽)으로 구성할 수 있습니다.
- ✓ 이 값을 최대화해야 하므로 아래 식을 유도할 수 있습니다.

$$\min(R_L, B_R) + \min(B_L, R_R)$$

0F. 선 연결하기



- ✓ 반대로 빨간 점(왼쪽)과 파란 점(왼쪽), 또는 빨간 점(오른쪽)과 파란 점(오른쪽)으로 구성하면 해당 구간을 지나지 않는 선분을 만들 수 있습니다.
- ✓ 이 선분의 수를 최대화하는 것이 해당 구간을 지나는 선분의 수를 최소화하는 것이므로

$$P - \min(R_L, B_L) - \min(R_R, B_R)$$

의 식으로 구할 수 있습니다.

- ✓ 미리 각 구간에 대해 R_L, B_L, R_R, B_R를 누적합을 통해 전처리하면 $\mathcal{O}(N)$의 시간 복잡도로 처리할 수 있습니다.



0G. GCD/LCM

`math, sort`

출제진 의도 – **Challenging**

✓ 출제자: 조문성`jms020820`

0G. GCD/LCM



- ✓ GCD와 LCM 연산은 각 소인수 관점에서 지수와 관련이 있습니다
- ✓ GCD는 작은 지수를, LCM은 큰 지수를 택하는 연산입니다.
- ✓ 즉 주어진 연산은 각 소인수 지수를 증가하도록 교환하는 것과 같습니다.
- ✓ 각 지수를 정렬하여 복원하면 정답을 구할 수 있습니다.
- ✓ M의 서로 다른 소인수 개수가 $\omega(M)$이면 시간 복잡도는 $\mathcal{O}(\omega(M)N\log N)$이 됩니다.



3A. 정12각형

`implementation, arithmetic`

출제진 의도 – **Easy**

- ✓ 제출 13번, 정답 12번 (정답률 92.31%)
- ✓ 처음 푼 팀: **Hippopus**, 1분
- ✓ 출제자: 이성현`hibye1217`

3A. 정12각형



- ✓ 혹시 알고 계셨습니까?
- ✓ 반지름이 r 인 원에 내접한 정 12 각형의 넓이는 $3r^2$ 입니다!



3B2A. MOBIS 프로젝트 신청하기

`implementation`
출제진 의도 – **Easy**

- ✓ Div. 3: 제출 29번, 정답 12번 (정답률 41.38%)
- ✓ Div. 2: 제출 34번, 정답 14번 (정답률 41.18%)
- ✓ Div. 3: 처음 푼 팀: **Hippopus**, 6분
- ✓ Div. 2: 처음 푼 팀: **YuriLove_/**, 3분
- ✓ 출제자: 최도현dohyonne

3B2A. MOBIS 프로젝트 신청하기



- ✓ 지문에 있는 내용을 조건문으로 구현하면 됩니다.
- ✓ 간단하게는 O가 아닌 문자의 개수가 0 또는 4 라면 `hi`, 아니라면 `bye`를 출력하면 됩니다.



3C. std.IO

`implementation, string, math`
출제진 의도 – **Easy**

- ✓ 제출 18번, 정답 12번 (정답률 66.67%)
- ✓ 처음 푼 팀: **Hippopus**, 13분
- ✓ 출제자: 이성현`hibye1217`

3C. std.IO



- ✓ 주어진 과정을 순서대로 구현하면 됩니다.
- ✓ 구현을 편하게 하고 싶으시면 문자열을 한 글자씩 읽으면서 `O`, `I`가 나올 때마다 바로 이진수의 끝에 $0, 1$을 붙여줘도 됩니다.

3C. std.IO



✓ 여담으로, 하이비는 이 문제의 예제에 길이 $3 + 10$의 문자열을 숨겨놓았습니다.



3D. 퍼리와 쿼리

`ad_hoc`, `implementation`, `string`
출제진 의도 – **Furry**

- ✓ 제출 33번, 정답 9번 (정답률 27.27%)
- ✓ 처음 푼 팀: **민트못감청년**, 35분
- ✓ 출제자: 이지훈`ezh8n`

3D. 퍼리와 쿼리



- ✓ i번 퍼리가 내민 부위의 위치를 $f(i)$라고 합시다.
- ✓ S_i가 T, M, P일 때 $f(i)$는 각각 $i-1, i+1, i$입니다.
- ✓ 제한에서 **구간 $[1, N]$은 완벽하다고 보장**합니다. 따라서 서로 다른 퍼리가 내민 부위는 이미 서로 다른 위치에 있습니다.
- ✓ 어떤 부분 구간을 고르더라도 이 성질은 유지됩니다. 이제 **구간 밖으로 나가는 부위가 있는지**만 확인하면 됩니다.

3D. 퍼리와 쿼리



- ✓ 구간 $[l, r]$ 안의 퍼리 중 왼쪽으로 벗어날 수 있는 것은 $S_l = \texttt{T}$인 경우뿐입니다. 이때 $f(l) = l - 1$입니다.
- ✓ 오른쪽으로 벗어날 수 있는 것은 $S_r = \texttt{M}$인 경우뿐입니다. 이때 $f(r) = r + 1$입니다.
- ✓ 두 경우가 모두 아니면 모든 부위가 구간 안에 있습니다. 서로 겹치지 않고 부위의 수와 위치의 수가 같으므로, 모든 위치를 정확히 한 번 채웁니다.

$$[l, r]\text{이 완벽하다} \iff S_l \neq \texttt{T}\text{ 이고 } S_r \neq \texttt{M}.$$

3D. 퍼리와 쿼리



- ✓ 문자열 S를 저장하고, 각 쿼리마다 양 끝 문자만 확인합니다.
- ✓ $S_l \neq$ T이고 $S_r \neq$ M이면 YES, 그렇지 않으면 NO를 출력합니다.
- ✓ $l = r$일 때도 같은 판정을 씁니다. 이 경우 두 조건을 모두 만족하는 문자는 P뿐입니다.
- ✓ 쿼리당 시간은 $\mathcal{O}(1)$입니다. 전체 시간은 $\mathcal{O}(N+Q)$, 문자열을 저장하는 공간은 $\mathcal{O}(N)$입니다.



3E2B. 돌려돌려돌림판

`math, implementation`
출제진 의도 – **Medium**

- ✓ Div. 3: 제출 30번, 정답 10번 (정답률 33.33%)
- ✓ Div. 2: 제출 26번, 정답 13번 (정답률 50.00%)
- ✓ Div. 3: 처음 푼 팀: **Hippopus**, 26분
- ✓ Div. 2: 처음 푼 팀: **팀노트로 아내만 25장 뽑아오는 행복한 상상**, 9분
- ✓ 출제자: 노경민 `gmroh06`

3E2B. 돌려돌려돌림판



- ✓ 회전해도 번호의 시계 방향 순서는 변하지 않으므로, **맨 위 칸의 번호만 관리**하면 됩니다.
- ✓ 맨 위 칸의 번호를 0부터 세어 p에 저장합니다. 처음에는 $p = 0$입니다.
- ✓ 1 x: 시계 방향으로 돌리면 맨 위 번호는 작아집니다. $p \leftarrow (p - x + N) \bmod N$.
- ✓ 2 y: 반시계 방향으로 돌리면 맨 위 번호는 커집니다. $p \leftarrow (p + y) \bmod N$.
- ✓ 3 z: 맨 위에서 시계 방향으로 $z - 1$칸 이동하므로, $(p + z - 1) \bmod N + 1$을 출력합니다.
- ✓ 쿼리당 $\mathcal{O}(1)$, 전체 시간 $\mathcal{O}(Q)$, 추가 공간 $\mathcal{O}(1)$.



3F. 색종이 자르기

`math`
출제진 의도 – **Hard**

- ✓ 제출 4번, 정답 2번 (정답률 50.00%)
- ✓ 처음 푼 팀: **민트못감청년**, 52분
- ✓ 출제자: 조문성`jms020820`

3F. 색종이 자르기



- ✓ 임의의 쌍 $i\,j$ 에 대해 최종 정답에 더해지는 가중치는 서로 분리될 때 딱 한 번 $W_i \cdot W_j$ 가 됩니다.
- ✓ 따라서 답은 자르는 순서와 상관없이 $\sum_{i<j} W_i \cdot W_j$ 가 됩니다.
- ✓ 다시 쓰면 $\frac{(\sum_{i=1}^{N} A_i)^2 - \sum_{i=1}^{N} A_i^2}{2}$ 이 됩니다.
- ✓ 시간 복잡도는 $\mathcal{O}\,(N)$ 입니다.



3G. 마검의 조율

`math`, `prefix_sum`
출제진 의도 – **Hard**

- ✓ 제출 9번, 정답 3번 (정답률 33.33%)
- ✓ 처음 푼 팀: **IamTeapot**, 176분
- ✓ 출제자: 송준민 `sjoonmin07`

3G. 마검의 조율



- ✓ i번째 방에서의 마력은 $x_i = d_1 + d_2 + \cdots + d_i$ 이므로, 총점은 $\sum_{i=1}^{N} x_i A_i$ 입니다.
- ✓ d_i는 $x_i, x_{i+1}, \ldots, x_N$에 모두 더해지므로, 총점에서 $A_i, A_{i+1}, \ldots, A_N$에 곱해집니다.
  따라서 d_i 기준으로 묶으면 총점 $= \sum_{i=1}^{N} d_i \cdot S_i$ 이고, 여기서 $S_i = A_i + A_{i+1} + \cdots + A_N$ (**suffix sum**)입니다.

3G. 마검의 조율



✓ **각 d_i 는 서로 독립적으로** 총점에 $d_i \cdot S_i$ 만큼 기여합니다. 따라서 $-i \le d_i \le i$ 범위에서 항을 최대화하려면 극단값만 고르면 됩니다.

- $S_i > 0$ 이면 $d_i = i$
- $S_i < 0$ 이면 $d_i = -i$
- $S_i = 0$ 이면 d_i 는 무엇이든 상관없음

3G. 마검의 조율



- ✓ 따라서 답은 $\sum_{i=1}^{N} i \cdot |S_i|$ 입니다.
- ✓ $S_N = A_N$ 에서 시작해 N 번 방부터 1 번 방까지 **뒤에서부터** 누적합을 구하며 $i \cdot |S_i|$ 를 더해 나가면 됩니다.
- ✓ 복잡도: 시간 $\mathcal{O}(N)$, 공간 $\mathcal{O}(N)$.



3H. 제트팩

dp

출제진 의도 – **Challenging**

- ✓ 제출 10번, 정답 0번 (정답률 0.00%)
- ✓ 처음 푼 팀: 없음
- ✓ 출제자: 김기용kky085437

3H. 제트팩



- ✓ 고도를 0으로 낮춘 위치를 i라고 합시다.
- ✓ 그 이후 다시 고도를 낮추기 전까지, 위치 x에서의 고도는 $x - i$로 유일하게 결정됩니다.
- ✓ 따라서 지금까지의 모든 선택을 기억할 필요 없이, **마지막으로 고도를 낮춘 위치**만 기억하면 충분합니다.

3H. 제트팩



- ✓ 이를 이용하여 $dp[i]$ = (i에서 마지막으로 고도를 낮췄을 때의 방법의 수)로 정의하는 것을 생각해 봅시다.
- ✓ 두 번 연속으로 고도를 낮추는 위치가 $j < i$라면, 고도는 M을 넘을 수 없으므로 반드시 $i - j \leq M$이어야 합니다.
- ✓ 따라서 장애물에 대한 조건을 잠시 무시하면 $dp[i] = \sum_{j=i-M}^{i-1} dp[j]$와 같은 전이를 얻을 수 있습니다.
- ✓ 이제 어떤 위치 i에서 실제로 고도를 낮출 수 있는지만 판정하면 됩니다.

3H. 제트팩



- ✓ 위치 i에서 고도를 0으로 낮췄다고 합시다.
- ✓ i보다 뒤에 있는 j번째 장애물을 넘을 때, 다시 고도를 낮추지 않았다고 하더라도 위치 $j-1$에서의 고도는 $j-1-i$입니다.
- ✓ 따라서 $j-1-i<H_j$인 장애물이 하나라도 존재한다면 위치 i에서 고도를 낮춘 뒤에는 그 장애물을 넘을 수 없습니다.
- ✓ 즉, j번째 장애물은 $j-1-i<H_j$를 만족하는 위치 i, 다시 말해 $j-H_j\le i\le j-1$에서 고도를 낮추는 것을 모두 금지합니다.

3H. 제트팩



- ✓ 따라서 각 장애물 j 마다 구간 $[j - H_j, j - 1]$ 을 금지 구간으로 표시하면 됩니다.
- ✓ 여러 구간에 대한 표시이므로 차분 배열(imos)을 사용하면 모든 위치가 가능한지 여부를 $\mathcal{O}(N)$ 에 구할 수 있습니다.
- ✓ 배열 H 를 뒤부터 보면서 앞으로 몇 칸 동안 초기화가 불가능해지는지를 봐도 $\mathcal{O}(N)$ 에 처리가 가능합니다.

3H. 제트팩



- ✓ ok_i를 위치 i에서 고도를 낮출 수 있는지 여부라고 합시다.
- ✓ $ok_i = \text{false}$라면 해당 위치를 마지막 선택으로 하는 방법은 존재하지 않으므로 $dp[i] = 0$입니다.
- ✓ 반대로 $ok_i = \text{true}$라면, 직전에 고도를 낮춘 위치는 $i - M, i - M + 1, \ldots, i - 1$ 중 하나일 수 있습니다.

3H. 제트팩



✓ 따라서 다음과 같은 전이식을 얻습니다.

$$dp[i] = \begin{cases} 0 & (ok_i = \text{false}) \\ \displaystyle\sum_{j=\max(0,i-M)}^{i-1} dp[j] & (ok_i = \text{true}) \end{cases}$$

3H. 제트팩



- ✓ 위치 0을 시작점을 나타내는 가상의 초기화 위치로 두고 $dp[0] = 1$로 시작합니다.
- ✓ 또한 위치 N을 도착점을 나타내는 가상의 위치로 생각하면 $dp[N]$이 모든 장애물을 통과하는 방법의 수가 됩니다.
- ✓ 위 합은 길이가 M인 연속 구간의 합이므로, 누적 합이나 슬라이딩 윈도우를 이용하면 각 전이를 $\mathcal{O}(1)$에 계산할 수 있습니다.
- ✓ 시간, 공간 복잡도는 $\mathcal{O}(N)$입니다.



2C. 실버 문제 만들기

`bruteforcing`, `bitmask`
출제진 의도 – **Medium**

- ✓ 제출 14번, 정답 12번 (정답률 85.71%)
- ✓ 처음 푼 팀: **숭고한전부떨어진사람**, 16분
- ✓ 출제자: 이승현`kuming19`

2C. 실버 문제 만들기



- ✓ 서로 다른 아이디어를 고르는 경우의 수는 2^N 이므로 각각의 경우에 대해 계산을 해주어도 충분히 시간 내에 들어옵니다.
- ✓ 최선의 시간 복잡도는 $\mathcal{O}\left(2^N\right)$ 이지만 $\mathcal{O}\left(N2^N\right)$ 으로 구현해도 전혀 문제되지 않습니다.



2D. 대회 문제 구성하기

dp, knapsack
출제진 의도 – **Hard**

- ✓ 제출 85번, 정답 9번 (8팀, 정답률 10.59%)
- ✓ 처음 푼 팀: **YuriLove_/**, 29분
- ✓ 출제자: 이승현kuming19

2D. 대회 문제 구성하기



- ✓ DP를 전이시킬 때, 태그의 문제가 정확히 하나가 포함되어야 한다는 점만 고려한다면, 기본적인 배낭 문제 아이디어로 풀 수 있습니다.
- ✓ 같은 태그를 가진 문제끼리 각각의 그룹으로 묶었을 때, 다음과 같은 DP 식을 세울 수 있습니다.

$$dp[t][s] = t\text{번 태그까지 고려해 난이도 합이 } s\text{일 때의 최소 노력}$$

- ✓ 최종적으로 고려해야 하는 상황은 전체 태그를 고려했을 때, 난이도의 합이 $A \leq s \leq B$인 경우이므로 답은 $\min_{A \leq s \leq B} dp[T][s]$ 입니다.
- ✓ 문제들의 전체 난이도 합은 10^7 이지만, B를 넘어가는 난이도는 따로 확인할 필요가 없으므로 시간 복잡도는 $\mathcal{O}(NB)$ 입니다.



2E. 자유시간권

`sweep_line`, `greedy`, `intervals`

출제진 의도 – **Hard**

- ✓ 제출 24번, 정답 7번 (정답률 29.17%)
- ✓ 처음 푼 팀: **YuriLove_/**, 65분
- ✓ 출제자: 이승현`kuming19`

2E. 자유시간권



- ✓ 자유시간권의 시작 시각 순으로 정렬하고, 앞 시간부터 차례대로 탐색합시다.
- ✓ 이때, 확인 중인 자유시간권은 여러 출제자 중 가장 빨리 자유시간이 끝난 출제자가 사용하는 것이 항상 이득임을 관찰할 수 있고, 이 성질을 이용하면 그리디하게 문제를 풀 수 있습니다.

2E. 자유시간권



- ✓ 각 출제자마다 자유시간이 끝나는 시간을 우선순위 큐로 관리한다면 $\mathcal{O}(\log N)$ 에 자유시간권을 사용할 출제자를 선정할 수 있습니다.
- ✓ 선정된 출제자는 자유시간이 끝난 시각부터 새로 사용하는 자유시간권의 시작 시각이 겹치지 않는다면 그동안에는 일을 해야 하므로, 출제자가 새로 자유시간권을 사용할 때마다 이 구간의 길이를 카운트해줍니다.
- ✓ 모든 확인이 끝난 후에 마지막으로 종료 시각까지 출제자가 일하는 시간을 처리해주면 답을 구할 수 있습니다.
- ✓ 이 과정을 모든 자유시간권에 대해 진행하면 되므로 시간 복잡도는 $\mathcal{O}(N(\log N + \log M))$ 입니다.



2F. 미로 탈출하기

`bfs`, `periodicity`, `state_space_search`
출제진 의도 – **Hard**

- ✓ 제출 18번, 정답 2번 (정답률 11.11%)
- ✓ 처음 푼 팀: **YuriLove_/**, 115분
- ✓ 출제자: 김태우`mathtw1030` 조문성`jms020820`

2F. 미로 탈출하기



- ✓ 길이가 d인 구간을 왕복하는 벽의 위치는 $2d$초마다 반복됩니다.
- ✓ 모든 벽의 움직임은 $L = \mathrm{lcm}(2d_1, \ldots, 2d_K)$초마다 반복됩니다. $d_i \leq 10$이므로 $L \leq 5040$입니다.
- ✓ 따라서 상태를 **현재 칸** (r, c)**와 시각** $t \bmod L$로 정의할 수 있습니다. 같은 상태에 다시 도착하면 이후 가능한 이동도 같습니다.
- ✓ 각 시각 $q \in [0, L)$에 대해 벽이 차지한 칸 `occupied[q]`와 벽이 이동한 방향을 미리 계산합니다.

2F. 미로 탈출하기



- ✓ 상태 $(A, t \bmod L)$에서 대기하거나 인접한 칸 B로 이동합니다. 다음을 모두 확인합니다.
  1. B가 미로 안의 고정된 벽이 아닌 칸입니다.
  2. 시각 $t+1$에 움직이는 벽이 B를 차지하지 않습니다.
  3. 시각 t에 B에 있던 **같은 벽**이 시각 $t+1$에 A로 이동하지 않습니다.
- ✓ 마지막 조건은 각 벽의 이동 $B \to A$를 금지 이동 $A \to B$로 저장해 $\mathcal{O}(1)$에 검사할 수 있습니다.
- ✓ $(1, 1, 0)$에서 BFS를 시작합니다. (N, M)에 처음 도착한 시각이 답이고, 큐가 비면 -1입니다.

2F. 미로 탈출하기



- ✓ 전처리한 점유 칸과 금지 이동은 문제의 두 이동 조건을 정확히 나타냅니다. 따라서 BFS의 간선은 가능한 1초 이동과 일대일로 대응합니다.
- ✓ 벽의 배치가 L초마다 반복되므로 같은 칸, 같은 시각 나머지에 더 늦게 도착해도 이득이 없습니다. 처음 방문한 상태만 탐색해도 충분합니다.
- ✓ 모든 간선의 비용이 1이므로 BFS가 목표 칸을 처음 방문한 시각이 최소 도착 시각입니다.
- ✓ 상태 수는 NML, 상태마다 후보 이동은 최대 5개입니다. 시간 $\mathcal{O}\left((NM+K)L\right)$, 공간 $\mathcal{O}\left((NM+K)L\right)$입니다.



2G. 별 배열

`math`, `bruteforcing`
출제진 의도 – **Challenging**

- ✓ 제출 17번, 정답 2번 (정답률 11.76%)
- ✓ 처음 푼 팀: **YuriLove_/**, 193분
- ✓ 출제자: 양성준 `egod1537`

2G. 별 배열



- ✓ 별 (x, y)가 직선 $y = ax + b$에 속하려면 $|y - (ax + b)| \leq 1$이어야 합니다.
- ✓ b에 대해 정리하면 $y - ax - 1 \leq b \leq y - ax + 1$입니다.
- ✓ a, x, y, b가 모두 정수이므로 가능한 b는 $y - ax - 1$, $y - ax$, $y - ax + 1$의 세 가지입니다.
- ✓ 따라서 기울기 a를 고정하고 모든 별을 순회하면서
  절편 b의 등장 횟수를 카운팅하면 됩니다.
- ✓ 어떤 절편 b의 등장 횟수가 k라는 것은,
  $y = ax + b$로부터 **세로 오차가 1 이하인 별이 k개**라는 뜻입니다.

2G. 별 배열



- ✓ 탐색해야 하는 기울기 a의 범위는 어떻게 될까요?
- ✓ 서로 x 좌표가 다른 두 별 (x_1, y_1), (x_2, y_2)가 하나의 직선에 속한다고 합시다.
- ✓ 각 별의 세로 오차가 최대 1이므로 $|a(x_1 - x_2)| \leq |y_1 - y_2| + 2$를 만족합니다.
- ✓ $|y_1 - y_2| \leq N - 1$이고, 서로 다른 정수 x 좌표이므로 $|x_1 - x_2| \geq 1$입니다.
- ✓ 따라서 $|a| \leq |a(x_1 - x_2)| \leq |y_1 - y_2| + 2 \leq N + 1$입니다.
- ✓ 즉 $-(N + 1) \leq a \leq N + 1$만 순회하면 됩니다.
- ✓ 최종 시간 복잡도는 $\mathcal{O}(NS)$입니다.



2H. 먼지 필터

`bitmask`, `prefix_sum`
출제진 의도 – **Challenging**

- ✓ 제출 21번, 정답 3번 (정답률 14.29%)
- ✓ 처음 푼 팀: **숭고한전부떨어진사람**, 218분
- ✓ 출제자: 최도현dohyonne

2H. 먼지 필터



- ✓ 필터와 먼지 뭉치는 $H \times W$ 크기의 비트마스크로 표현할 수 있습니다.
- ✓ i번 필터의 비트마스크를 A_i, 먼지 뭉치의 비트마스크를 B라고 해봅시다.
- ✓ i번 필터가 우수한 필터가 되기 위해서는 (A_i & B) == A_i 를 만족해야 합니다.
- ✓ 각 필터에 대해서 우수한 필터가 될 수 있는 먼지 뭉치들의 비트마스크를 구해놓읍시다.
- ✓ 각 먼지 뭉치에 대해 우수한 필터의 개수를 누적합으로 만들어 놓는다면, 쿼리가 들어올 때마다 $\mathcal{O}(1)$에 우수한 필터의 개수를 계산할 수 있습니다.
- ✓ 시간 복잡도는 $\mathcal{O}\left(N \times 2^{H \times W} + Q\right)$ 입니다.



1A. 윈드 쉴드 미러 디스플레이

`string`, `suffix_array`, `combinatorics`
출제진 의도 – **Hard**

- ✓ 제출 6번, 정답 1번 (정답률 16.67%)
- ✓ 처음 푼 팀: **kokiri is cute**, 258분
- ✓ 출제자: 이성현`hibye1217`

1A. 윈드 쉴드 미러 디스플레이



- ✓ 글자 수에 비례하는 풀이부터 생각해봅시다.
- ✓ 문자열의 맨 바깥쪽부터 비교를 시작하면서, 새로 등장하는 문자열의 뒤집기 여부를 결정하면 됩니다.
- ✓ 쿼리로 주어진 문자열을 적절히 뒤집은 뒤 합한 문자열을 T 라고 합시다.
- ✓ $D_{i,j,l,r,x,y}$ 를 $T_i = T_j$ 를 만족해야 할 때, S_l 과 S_r 의 방향을 각각 x 와 y 로 고정한 경우의 수로 정의해서 해결할 수 있습니다.

1A. 윈드 쉴드 미러 디스플레이



- ✓ 하지만 글자 수가 너무 많습니다.
- ✓ (l, r) 이 같은 부분에 대해 $D_{i,j,l,r,x,y}$ 를 한 번에 묶어서 진행해야 합니다.
- ✓ 이를 위해서는 S 의 두 부분 문자열이 같은지 판별해야 합니다. 이는 Suffix Array + LCP Array + Range Minimum Query 로 구할 수 있습니다.
- ✓ 두 부분 문자열에서 겹치는 구간에 대해, 이를 적절히 뒤집어서 같은 문자열이 나오는지 판별하고, 만약 같은 문자열이 나온다면 경우의 수를 업데이트해주면 됩니다.



1B. 직사각형 고리 줍기

`implementation`
출제진 의도 – **Easy**

- ✓ 제출 43번, 정답 22번 (정답률 51.16%)
- ✓ 처음 푼 팀: **루루는 왔는데 테라가 안왔어요 ㅠㅠ**, 5분
- ✓ 출제자: 조문성`jms020820`

1B. 직사각형 고리 줍기



- ✓ $L = \max X_{i,1}\ R = \min X_{i,2}\ S = \max Y_{i,1}\ E = \min Y_{i,2}$ 일 때
- ✓ $L \leq R$ 이고 $S \leq E$ 면 (L, S) 는 모든 직사각형이 포함하는 점입니다.
- ✓ 위 조건이 아니면 $X_{i,1} > X_{j,2}$ 또는 $Y_{i,1} > Y_{j,2}$ 를 만족하는 $i\ j$ 를 찾으면 됩니다.
- ✓ 시간 복잡도는 $\mathcal{O}(N)$ 입니다.



1C. Sumptuous K-th Highway

`dag`, `priority_queue`
출제진 의도 – **Hard**

- ✓ 제출 46번, 정답 10번 (정답률 21.74%)
- ✓ 처음 푼 팀: **전국 짱크 연합**, 70분
- ✓ 출제자: 최도현dohyonne

1C. Sumptuous K-th Highway



- ✓ 주어진 그래프는 1 번 정점(요금소)에서 다른 모든 정점으로 이동이 가능한 DAG입니다.
- ✓ 조건을 만족하도록 간선(도로)을 제거하기 위해서는 2 번부터 N 번까지 각 정점이 자신에게 들어오는 간선을 정확히 하나씩만 가지면 됩니다.
- ✓ 이제 남은 $N-1$ 개의 간선의 이용료 합이 K 번째로 큰 경우를 찾아야 합니다.
- ✓ 2 번부터 N 번까지 각 정점마다, 자신에게 들어오는 간선들의 이용료를 원소로 가지는 리스트가 있다고 생각해 봅시다.
- ✓ 각 리스트에서 원소를 하나씩만 선택해서 그 합이 K 번째로 큰 경우를 찾는 문제가 됩니다.
- ✓ $N-1$ 개의 리스트를 한꺼번에 고려하면 어려우므로, 우선 2개의 리스트로만 생각해 봅시다.

1C. Sumptuous K-th Highway



- ✓ 두 리스트를 각각 내림차순으로 정렬하고, 우선순위 큐에 (첫 번째 리스트의 인덱스, 두 번째 리스트의 인덱스, 각 리스트의 인덱스에 해당하는 원소의 합)을 넣어봅시다.
- ✓ 첫 번째 혹은 두 번째 리스트의 인덱스를 1 씩 증가시키며 새로운 원소 쌍을 탐색합니다. 이 과정을 통해 최대 K 개의 원소 합을 구하고 나면, 그 이후의 원소 쌍은 고려하지 않아도 됩니다. 이때, 같은 원소 쌍이 큐에 중복으로 들어가지 않도록 별도의 처리가 필요합니다.
- ✓ 이 원소 합들을 새로운 리스트로 생각하여 다른 리스트와 병합하는 과정을 반복하면 정답을 구할 수 있습니다.

1C. Sumptuous K-th Highway



- ✓ 두 리스트를 한 번 병합할 때 $\mathcal{O}(K \log K)$, 이를 $N-2$번 반복하면 $\mathcal{O}(NK \log K)$가 됩니다.
- ✓ 리스트를 정렬하는 시간도 필요하므로 전체 시간 복잡도는 $\mathcal{O}(NK \log K + M \log M)$입니다.



1D. 고양이 빚기

segment_tree, lazy_propagation, sweeping
출제진 의도 – **Furry**

- ✓ 제출 34번, 정답 7번 (정답률 20.59%)
- ✓ 처음 푼 팀: **intiger**, 26분
- ✓ 출제자: 오서준osjoon

1D. 고양이 빚기



연산

- ✓ 찰흙덩이의 상태 s는 현재 붙어 있는 부위의 집합이므로, 3비트로 나타낼 수 있습니다.
- ✓ 세 부위가 모두 붙으면 즉시 비워지므로, 가능한 s는 0부터 6까지 7가지입니다.
- ✓ 여기에 각 찰흙덩이별로 현재까지 완성된 고양이의 수 c를 별도로 저장합시다.
- ✓ 각 쿼리는 다음과 같은 연산 한 번으로 생각할 수 있습니다.

$$(s,\ c) \leftarrow \begin{cases} (s \mid b,\ c) & (s \mid b \neq 7) \\ (0,\ c+1) & (s \mid b = 7) \end{cases} \qquad (b \in \{1, 2, 4\})$$

- ✓ 결합법칙은 성립하지만 교환법칙은 성립하지 않습니다. 따라서 순서가 중요합니다.

1D. 고양이 빛기



별해 – 느리게 갱신되는 세그먼트 트리

- ✓ 문제를 읽은 그대로 풀어 봅시다.
- ✓ 좌표 압축을 수행하면 좌표는 $\mathcal{O}(Q)$ 개가 되니, 이를 리프로 삼는 세그먼트 트리를 생각할 수 있습니다.
- ✓ 쿼리가 구간 상태를 변경(s)함과 동시에 조회(c 갱신)하므로, 느리게 갱신되는 세그먼트 트리가 필요합니다.
- ✓ 노드는 기본적으로 구간의 $(s,\ c)$를 저장합니다. 단 s는 7가지 상태 각각에 대해 그 상태인 좌표의 개수를 저장합니다.
- ✓ 하지만 레이지 상태는 어떻게 저장해야 할까요?

1D. 고양이 빛기



별해 – 느리게 갱신되는 세그먼트 트리

- ✓ 레이지 상태가 저장해야 하는 것은 **순서대로 쌓여 있는 쿼리**입니다.
- ✓ 이는 본질적으로 합성함수와 같으므로, 다음과 같이 나타낼 수 있습니다.
  - $\mathrm{dst}[s]$: 상태 s에서 시작해서 저장된 쿼리를 모두 처리했을 때 도달하는 마지막 상태
  - $\mathrm{cnt}[s]$: 상태 s에서 시작해서 저장된 쿼리를 모두 처리했을 때 완성되는 고양이의 수
- ✓ 합성 한 번은 $\mathcal{O}(7)$에 계산할 수 있습니다.

$$\mathrm{dst}_{g\cdot f}[s] = \mathrm{dst}_g[\mathrm{dst}_f[s]], \quad \mathrm{cnt}_{g\cdot f}[s] = \mathrm{cnt}_f[s] + \mathrm{cnt}_g[\mathrm{dst}_f[s]]$$

- ✓ 시간 복잡도는 $\mathcal{O}(7Q \log Q)$ 입니다.

1D. 고양이 빛기



정해: 세그먼트 트리 + 스위핑

- ✓ 여기서 축을 돌려 봅시다. 쿼리를 (좌표 x, 시간 i) 평면에 그리면, 쿼리 i는 높이 i에서 $[l_i, r_i]$를 덮는 가로 선분입니다.
- ✓ 좌표 x의 결과는 x를 지나는 세로선에 걸린 선분들을 **아래(이른 시간)에서 위로** 합성한 것입니다.
- ✓ 지금까지는 **시간 방향**으로 훑으며 **좌표축**에 트리를 세웠습니다.
- ✓ 반대로 **좌표 방향**으로 훑으며 **시간축**에 트리를 세우면 어떻게 될까요?

1D. 고양이 빛기



정해: 세그먼트 트리 + 스위핑

	Lazy (별해)	정해
훑는 축	시간 i	좌표 x
트리의 축	좌표 x	시간 i
쿼리 하나당	구간 갱신 및 조회	점 갱신 2번 및 구간 조회(전체) 1회
노드가 드는 것	$\mathrm{dst}[0\ldots6]$, $\mathrm{cnt}[0\ldots6]$, 레이지	$\mathrm{dst}[0\ldots6]$, $\mathrm{cnt}[0\ldots6]$ 만

✓ 합성 함수는 그대로이고, 구간 갱신이 점 갱신 두 번으로 바뀌면서 레이지가 사라집니다. 시간 복잡도는 같지만, 방문하는 노드와 전파가 줄어 상수가 작아집니다.

1D. 고양이 빚기



정해: 세그먼트 트리 + 스위핑

- ✓ 시간축 세그먼트 트리의 리프 i는, 켜져 있으면 쿼리 i의 함수를, 꺼져 있으면 아무것도 하지 않는 항등함수를 가집니다.
- ✓ 부모는 순서를 지켜 왼쪽(이른 쿼리)과 오른쪽(늦은 쿼리)을 합성합니다.
- ✓ 그러면 루트는 x를 덮는 모든 쿼리들을 시간순으로 합성한 함수가 됩니다.
- ✓ 리프를 켜고 끄는 이벤트($x = l_i$ 또는 $x = r_i + 1$)를 좌표순으로 처리하면서, 답에 $(x - x_{\text{prev}}) \times \text{cnt}[0]$을 더합니다.
- ✓ 이벤트가 $2Q$개이고 점 갱신 한 번이 $\mathcal{O}(7 \log Q)$이므로, 시간 복잡도는 $\mathcal{O}(7Q \log Q)$, 공간 복잡도는 $\mathcal{O}(Q)$입니다.



1E. 구 구조 구하기

`interactive`, `trees`, `stack`

출제진 의도 – **Hard**

- ✓ 제출 36번, 정답 11번 (정답률 30.56%)
- ✓ 처음 푼 팀: **Reboot Ssal Game**, 28분
- ✓ 출제자: 이지훈`ezh8n`

1E. 구 구조 구하기



- ✓ 구 구조는 잎이 왼쪽부터 $1, \ldots, N$ 이고, **모든 내부 정점의 자식이 둘인 순서 있는 이진 트리**입니다.
- ✓ $a < b < c$에 대해 $u = \mathrm{LCA}(a, b), v = \mathrm{LCA}(b, c)$ 라 합시다. 두 정점은 서로 다르며 조상 관계입니다.
- ✓ 질문 ? a b c의 답이 L이면 u가 v의 진자손이고, R이면 v가 u의 진자손입니다.
- ✓ 즉, **왼쪽 두 구를 먼저 묶을지, 오른쪽 두 구를 먼저 묶을지** 비교할 수 있습니다.

1E. 구 구조 구하기



- ✓ T_i를 원래 트리에서 i보다 큰 잎을 지우고, 자식이 하나인 정점을 없앤 트리로 정의합니다.
- ✓ 새 잎 $c = i + 1$을 붙일 때에는 T_i의 **오른쪽 끝 경로**에 있는 어떤 부분 트리 B를 $(B\ c)$로 바꾸기만 하면 됩니다.
- ✓ 그 경로의 왼쪽 형제 부분 트리들과 마지막 잎을, 위에서 아래 순서로 스택 S에 저장합니다. S를 오른쪽부터 묶으면 T_i가 됩니다.
- ✓ 새 잎이 들어갈 위치를 경로의 **아래에서 위로** 찾으면, 지나간 정점은 스택에서 합쳐 없앨 수 있습니다.

1E. 구 구조 구하기



- ✓ 빈 스택으로 시작하여 $c = 1, 2, \ldots, N$을 순서대로 처리합니다.
- ✓ 스택에 두 구 이상 있으면 맨 위의 두 구를 A, B라 합시다. 각 구의 오른쪽 끝 잎 a, b로 ? a b c를 질문합니다.
- ✓ L이면 A, B를 꺼내 (AB)를 넣고 반복합니다. R이면 반복을 멈춥니다. 스택에 한 구만 남아도 멈춥니다.
- ✓ 잎 c를 스택에 넣습니다. 모든 잎을 처리한 뒤에는 맨 위의 두 구를 (AB)로 합치는 작업을 하나가 남을 때까지 반복합니다.

1E. 구 구조 구하기



- ✓ 삽입 후보 B는 처음에는 잎 i입니다. 이미 지나온 경로의 정점들을 합친 뒤에는, 삽입 위치가 B 내부에 있지 않음을 알고 있습니다.
- ✓ A, B의 부모를 p라 합시다. L이면 p가 새 분기점보다 아래이므로 (AB) 전체를 보존하고 후보를 그 부모 쪽으로 올립니다.
- ✓ R이면 새 분기점이 p 아래에 있습니다. 후보 내부는 제외되었으므로 B 바로 위에 $(B\,c)$를 붙이는 것이 맞습니다.
- ✓ 부모가 없으면 $(B\,c)$가 새 루트입니다. 따라서 매 단계의 스택은 T_{i+1}을 나타내며, 마지막에 얻는 T_N은 원래 구 구조와 같습니다.

1E. 구 구조 구하기



- ✓ $N \geq 2$일 때 R 답은 $c = 3, \ldots, N$ 마다 최대 한 번으로, 총 $N - 2$번 이하입니다.
- ✓ L 답마다 스택 크기가 하나 줄어듭니다. 잎을 총 N번 넣고 마지막 처리 직후 크기가 2 이상이므로, L도 $N - 2$번 이하입니다.
- ✓ 따라서 질문 수는 $2N - 4 \leq 2N$입니다. $N = 1, 2$는 질문 없이 처리합니다.
- ✓ 구는 자식 포인터로 합칩니다. 마지막 순회에서 잎은 `.`, 양쪽 자식의 표현이 A, B인 정점은 `(AB)`로 출력합니다. 시간 $\mathcal{O}(N)$, 공간 $\mathcal{O}(N)$.
- ✓ 정답은 `!S` 형식입니다. 모든 질문과 정답을 출력한 뒤 버퍼를 비웁니다.



1F. 숭고한 만들기

ad_hoc, constructive

출제진 의도 – **Medium**

- ✓ 제출 45번, 정답 22번 (정답률 48.89%)
- ✓ 처음 푼 팀: **1 MiB**, 9분
- ✓ 출제자: 김기용kky085437

1F. 숭고한 만들기



- ✓ 격자가 주어졌을 때 숭고한 쌍의 개수를 세 봅시다.
- ✓ 모든 K를 기준으로 (인접한 S의 개수) $\times$ (인접한 H의 개수)를 세면 됩니다.
- ✓ $2 \times N$ 크기의 격자에서는 인접한 칸이 최대 3개이므로 위 값의 상한은 2입니다.
- ✓ 따라서 답의 상한이 $\frac{4}{3}N$임을 알 수 있습니다.
- ✓ 위 상한을 만족하는 격자를 구성하려면 다음 조건을 만족하면 됩니다.
  - K끼리 이웃하면 안 됩니다.
  - K는 가장자리 칸에 있으면 안 됩니다.
  - 각 K마다 인접한 S와 H가 최소 하나 있어야 합니다.

1F. 숭고한 만들기



- ✓ 다음은 조건을 만족하는 배치의 예시입니다.

SKSKSKS...SSS　　SKHSKH...KSH

HHKHKHK...HHH　　HSKHSK...SKH

- ✓ $N = 3$일 경우 예외적으로 상한을 만족할 수 없습니다.
- ✓ 이 경우 숭고한 쌍의 개수가 3개인 격자를 아무거나 출력하면 됩니다.



1G. 마석 나누어가지기

`min_cut`, `fractional_binary_search`
출제진 의도 – **Challenging**

- ✓ 제출 1번, 정답 0번 (정답률 0.00%)
- ✓ 처음 푼 팀: 없음
- ✓ 출제자: 이성현`hibye1217`

1G. 마석 나누어가지기



- ✓ 결론부터 말합니다.
- ✓ 마법사의 집합 S에 대해, S에 속한 마법사 중 1명 이상이 사용할 수 있는 마석의 집합을 $P(S)$라고 합시다.
- ✓ 문제의 답은 $P(S)$의 마력의 합을 $|S|$로 나눈 값의 최소입니다.

1G. 마석 나누어가지기



- ✓ 이 값이 상한임은 자명합니다. S에 속한 마법사들이 마력을 나누어 가져야 하기 때문이죠.
- ✓ 이 값이 도달 가능한 값임도 홀의 결혼 정리를 사용하여 간단히 증명할 수 있습니다.

1G. 마석 나누어가지기



- ✓ 그렇다면 이 값은 어떻게 찾을까요?
- ✓ 이진 탐색을 해봅시다. p/q를 고정하고, 답이 p/q 이상인지 확인을 해봅시다.
- ✓ 편의상 $X(S)$를 $P(S)$의 마력의 합이라고 합시다.
- ✓ 그러면 $p/q \leq X(S)/|S|$, 즉 $qX(S) - p|S| \geq 0$인 집합 S가 존재하는지 판별하면 됩니다.

1G. 마석 나누어가지기



- ✓ 이를 다음 문제로 환원해봅시다.
- ✓ 마법사를 집합 S에 넣을 때마다 p의 비용이 발생합니다.
- ✓ 마석이 집합 $P(S)$에 들어갈 때마다 마력 1당 q의 이익이 발생합니다.
- ✓ 마법사를 집합 S에 넣을 때마다, 마법사가 사용할 수 있는 마석들이 전부 $P(S)$에 들어가야 합니다.

1G. 마석 나누어가지기



- ✓ 이는 다음과 같은 플로우 모델링으로 해결할 수 있습니다.
- ✓ Source에서 마법사 i로 가중치 p의 간선을 연결합니다.
- ✓ 마석 j에서 Sink로 가중치 $q \times A_j$의 간선을 연결합니다.
- ✓ 마법사 i가 마석 j를 사용할 수 있다면 가중치 ∞의 간선을 연결합니다.
- ✓ 이 그래프의 최대 유량이 $p \times N$이면 됩니다.



1H. 가장 긴 증가하는 모든 부분 수열

`lis`, `combinatorics`

출제진 의도 – **Medium**

- ✓ 제출 62번, 정답 15번 (정답률 24.19%)
- ✓ 처음 푼 팀: **전국 짱크 연합**, 22분
- ✓ 출제자: 이성현`hibye1217`

1H. 가장 긴 증가하는 모든 부분 수열



- ✓ 다음 관찰을 해봅시다.
- ✓ 수열의 어떤 원소가 LIS에 등장한다면, 해당 원소가 등장하는 위치는 고정됩니다.
- ✓ 그렇지 않다면, 그 원소를 기준으로 앞/뒤를 잘라서 이어붙여 더 긴 LIS를 만들 수 있습니다.

1H. 가장 긴 증가하는 모든 부분 수열



- ✓ 따라서 각 원소에 대해 LIS에서 등장하는 위치와 횟수를 계산하면 됩니다.
- ✓ L_i를 i번째 원소로 끝나는 LIS의 길이라고 합시다.
- ✓ C_i를 i번째 원소로 끝나는 LIS의 개수라고 합시다.
- ✓ R_i를 i번째 원소에서 시작하는 LIS의 길이라고 합시다.
- ✓ D_i를 i번째 원소에서 시작하는 LIS의 개수라고 합시다.

1H. 가장 긴 증가하는 모든 부분 수열



- ✓ LIS에서 등장하는지 여부는 $L_i + R_i - 1$이 LIS의 길이인지 확인하면 됩니다.
- ✓ LIS에서 등장하는 위치는 L_i 입니다.
- ✓ LIS에서 등장하는 횟수는 $C_i \times D_i$ 입니다.
- ✓ 따라서 답의 L_i 번째 위치에 $C_i \times D_i$ 를 더해주면 됩니다.



1I. Simple Permutation Problem

`sqrt_decomposition, ad_hoc`
출제진 의도 – **Hard**

- ✓ 제출 29번, 정답 1번 (정답률 3.45%)
- ✓ 처음 푼 팀: **Reboot Ssal Game**, 270분
- ✓ 출제자: 조문성 `jms020820`

1I. Simple Permutation Problem



- ✓ 문제의 식은 적어도 3개의 점을 포함하는 직사각형 넓이의 최솟값으로 해석할 수 있습니다.
- ✓ 연속한 3개의 점을 선택하면 정답의 상한이 $3N$ 임을 알 수 있습니다. 비둘기집 원리를 이용하면 $2N + \alpha$ 까지도 보일 수 있습니다. :)
- ✓ 따라서 최적 직사각형의 가로 또는 세로 길이는 $\mathcal{O}\left(\sqrt{N}\right)$ 입니다.

1I. Simple Permutation Problem



- ✓ B를 $\mathcal{O}\left(\sqrt{N}\right)$으로 잡고, 가로 방향의 인덱스 차이가 B 이하인 경우만 보겠습니다. 세로 방향은 역순열에 같은 과정을 적용하면 됩니다.
- ✓ 왼쪽 끝점 i를 고정하고, 오른쪽 끝점 j를 $i+1$부터 $j-i \leq B$인 범위에서 증가시키며 스위핑합니다.
- ✓ 이 과정에서 현재 집합에 있는 값 중 주어진 값 x의 바로 다음 값을 찾아야 합니다. 기존 정해에서는 이를 연결 리스트로 처리했습니다.

1I. Simple Permutation Problem



- ✓ 그런데 검수진의 풀이 중 자료구조를 사용하지 않는 훨씬 좋은 풀이가 있어 소개합니다!
- ✓ i를 고정하고 j를 스위핑하면서, $P_{i+1}, \ldots, P_j$ 중 P_i보다 큰 값 중 가장 작은 두 값을 $up_1 < up_2$, 작은 값 중 가장 큰 두 값을 $down_1 < down_2$로 유지합니다.
- ✓ 각 j에서 $\min(up_2 - P_i + 1,\ P_i - down_1 + 1,\ up_1 - down_2 + 1) \cdot (j - i + 1)$을 정답 후보로 확인합니다.
- ✓ 선택한 $up, down$이 P_j가 아닐 수 있으므로, $j - i + 1$을 곱하면 실제 필요한 가로 길이보다 크게 계산될 수 있습니다. 하지만 선택한 두 점 중 오른쪽 점을 처리할 때 이미 더 짧은 가로 길이로 계산했으므로, 이후 크게 계산된 값은 최솟값에 영향을 주지 않습니다.
- ✓ 따라서 최종 시간 복잡도는 $\mathcal{O}\left(N\sqrt{N}\right)$입니다.



1J. Do You Love Geometry?

`geometry`, `case_work`
출제진 의도 – **No!**

- ✓ 제출 75번, 정답 11번 (정답률 14.67%)
- ✓ 처음 푼 팀: **Reboot Ssal Game**, 83분
- ✓ 출제자: 이성현`hibye1217`

1J. Do You Love Geometry?



- ✓ 특정 선분을 빗변으로 하는 직각삼각형을 만들기 위해서는 세 번째 점을 해당 선분을 지름으로 하는 원에 놓아야 합니다.
- ✓ 따라서, 두 직각삼각형을 동시에 만들기 위해서는 두 원의 교차점을 선택해야 합니다.
- ✓ 즉, 두 원의 교차점이 존재한다면 두 선분은 연결되어 있습니다.
- ✓ 교차점의 개수를 셀 때 선분의 끝점은 세지 않아야 함에 주의하세요.

1J. Do You Love Geometry?



- ✓ 모든 두 선분 쌍에 대한 연결 여부를 계산한 뒤에는 방향 비순환 그래프에서의 최장 경로를 찾는 문제가 됩니다.
- ✓ 이는 간단한 DP로 해결할 수 있습니다.



1K. 주식 거래

`li_chao_tree`, `convex_hull_trick`, `dp`
출제진 의도 – **Challenging**

- ✓ 제출 10번, 정답 6번 (정답률 60.00%)
- ✓ 처음 푼 팀: **ThatOnePythonUser**, 193분
- ✓ 출제자: 이승현`kuming19`

1K. 주식 거래



- ✓ 주식의 가격이 변동할 때마다, 해당 가격일 때, 최적의 거래소가 변동합니다.
- ✓ 가격이 x 일 때 최적의 거래소를 구해봅시다.
- ✓ 수수료 A_i 는 가격에 따라 변동하는 반면에 B_i 는 항상 고정되어 있습니다. 정수로 다루기 위해 10^5 를 곱해준다면,
- ✓ 거래소의 수수료는 $y = A_i x + 10^5 B_i$ 의 직선으로 나타낼 수 있습니다.
- ✓ 이 직선들을 Convex Hull Trick을 통해 각 주식의 가격마다 수수료가 가장 낮은 거래소를 찾을 수 있습니다.

1K. 주식 거래



- ✓ 이후는 그리디를 통해 처리할 수 있습니다.
- ✓ d 날짜에 주식을 가지고 있을 경우의 최대 주식 수와 현금, d 날짜에 현금만 가지고 있을 때의 최대 현금을 관리하고, 매 날짜마다 갱신해주면 답을 구할 수 있습니다.
- ✓ 시간 복잡도는 $\mathcal{O}\left((M+N)\log M\right)$ 입니다.

1K. 주식 거래



- ✓ 또는 리차오 트리를 이용해서도 구할 수 있습니다.
- ✓ CHT를 리차오 트리로 대체할 수 있다는 사실은 자명하고, 그리디의 경우에도 주식의 수를 기울기 값으로 하고 현금을 y 절편으로 갖는 직선을 통해 구할 수 있습니다.
- ✓ 이 경우 시간 복잡도는 $\mathcal{O}\left((N+M)\log N\right)$ 입니다.



1L. 트리 만들기

number_theory, constructive
출제진 의도 – **Hard**

- ✓ 제출 55번, 정답 21번 (정답률 38.18%)
- ✓ 처음 푼 팀: **루루는 왔는데 테라가 안왔어요 ㅠㅠ**, 28분
- ✓ 출제자: 하준서mathjuny

1L. 트리 만들기



- ✓ 가독성을 위해 $M = \frac{N}{2}, 2M = N$ 으로 서술합니다.
- ✓ 파란색 간선은 연결해서 손해볼 것이 없으니 M 개의 파란색 간선을 전부 연결합니다.
- ✓ 이제 트리 구조를 만들기 위해서는 $M - 1$ 개의 빨간색 간선이 필요합니다.
- ✓ 이제 빨간색 간선의 값이 될 수 있는 수들을 생각해 봅시다.
- ✓ M 보다 큰 수는 1 부터 $2M$ 까지의 수들 중 어떤 두 수의 공약수가 될 수 없습니다.
- ✓ M 는 M 와 $2M$ 의 공약수이지만, 이미 M 와 $2M$ 은 파란색 간선으로 이어져 있습니다.
- ✓ 따라서 $M - 1$ 개의 빨간색 간선이 1 부터 $M - 1$ 까지의 값을 하나씩 가지도록 해야 합니다.

1L. 트리 만들기



- ✓ 파란색 간선들을 정점 번호의 순서쌍으로 나열하면 $(1, M+1), (2, M+2), \cdots, (M, 2M)$ 와 같이 나타낼 수 있습니다.
- ✓ 위 순서쌍들을 아래처럼 왼쪽부터 재배치해 봅시다.
- ✓ $(M, 2M), (M+1, 1), (M+2, 2), \cdots, (2M-1, M-1)$
- ✓ 각 순서쌍의 첫 번째 값이 M 개의 연속된 수들임을 볼 수 있습니다.
- ✓ 이제 1 부터 $M-1$ 까지의 값을 가지는 빨간색 간선을 연결해 봅시다.

1L. 트리 만들기



- ✓ $(M, 2M), (M+1, 1), (M+2, 2), \cdots, (2M-1, M-1)$
- ✓ 값이 k인 빨간색 간선을 구성하는 하나의 방법은 k와 k의 배수를 연결하는 것입니다.
- ✓ 1부터 $M-1$까지의 k에 대해, k번 정점이 속해 있는 순서쌍 왼쪽에는 k개의 순서쌍이 있고, 이 k개의 순서쌍의 첫 번째 값은 k개의 연속된 정수입니다.
- ✓ 따라서 이 중 k의 배수가 정확히 하나 존재합니다. 해당 정점을 k번 정점과 연결해 줍니다.
- ✓ 위와 같이 간선을 연결하면 트리가 됨을 자명하게 보일 수 있습니다.
- ✓ 트리 만들기 성공!