2024 HCPC

by 2024 ALOHA

Official Solutions

출제

✓ 구건모	thak00	한양대학교 컴퓨터소프트웨어학부 / ALOHA
✓ 김동민	robert334	한양대학교 컴퓨터소프트웨어학부 / ALOHA
✓ 김현중	paull04	한양대학교 컴퓨터소프트웨어학부 / ALOHA
✓ 윤아람	ramzxc	한양대학교 컴퓨터소프트웨어학부 / ALOHA
✓ 이현빈	my3	한양대학교 컴퓨터소프트웨어학부 / ALOHA
✓ 조찬우	myyh1234	한양대학교 컴퓨터소프트웨어학부 / ALOHA

검수

- ✓ gunwookim
- ✓ heeda0528
- ✓ ingyu1008
- ✓ jk410
- ✓ juneharold
- ✓ junseo
- ✓ mjhmjh1104
- ✓ tlsdydaud1

Sponsors



Sponsors



Sponsors



Sponsors



Sponsors



SMT TOTAL SOLUTION

Beginner Division

문제		의도한 난이도	출제자
A	그런 사람은 없었습니다	**Easy**	윤아람, 조찬우
B	수수수수퍼노바	**Easy**	윤아람, 조찬우
C	순열 복원	**Medium**	김현중
D	아무나 풀어주세요	**Hard**	구건모
E	점과 원	**Hard**	이현빈
F	지나칠 수 없는 지하철 게임	**Medium**	윤아람, 조찬우
G	컴소인의 크리스마스	**Medium**	윤아람, 조찬우
H	코인의 신 건모	**Easy**	구건모
I	피아노	**Medium**	조찬우
J	HCPC 팀 짜기	**Hard**	김동민, 김현중

Advanced Division

문제		의도한 난이도	출제자
A	HCPC 팀 짜기	**Medium**	김동민, 김현중
B	흑백조경사	**Hard**	조찬우
C	아무나 풀어주세요	**Medium**	구건모
D	순열 복원	**Easy**	김현중
E	수열과 쿼리 HY	**Hard**	조찬우
F	수수수수퍼노바	**Easy**	윤아람, 조찬우
G	돌 게임 nm	**Easy**	김현중
H	덧셈 팰린드롬 수열과 쿼리	**Medium**	구건모
I	나연 정렬	**Medium**	조찬우
J	간선을 하나 그어서 루트까지 거리의 합을 최소로 만들기로 했습니다	**Challenging**	김현중

BA. 그런 사람은 없었습니다

`implementation`
출제진 의도 – **Easy**

- 처음 푼 팀: **역시내청춘러브캠퍼스는잘못됐다**, 40분
- 출제자: 윤아람, 조찬우

BA. 그런 사람은 없었습니다

- 폼 제출을 잊은 회원의 조건은 아래 두 조건으로 나타낼 수 있습니다.
  1. 해당 회원이 입금 전에 폼 제출을 하지 않음
  2. 입금 이후 S 명이 폼을 작성할 때까지 폼 작성을 하지 않음
- 입금한 사람만 폼 제출을 잊은 사람이 될 수 있으니 입금을 중심으로 생각합시다.
- 각 입금에 대해 위 두 조건을 검사하면 해당 입금자가 폼 제출을 잊었는지 확인할 수 있습니다.

- 사건을 일어난 시간 순서대로 확인합시다.
- 각 회원이 폼 제출을 했는지 체크하는 배열을 사용하면 1 번 조건을 $O(1)$ 에 확인할 수 있습니다.
- 각 입금 사건마다 이후 사건을 모두 확인하면 2 번 조건을 $O(M)$ 에 처리할 수 있습니다.
- 이렇게 두 조건을 확인하면 전체 시간복잡도는 $O(NM)$ 이며, 문제를 해결할 수 있습니다.

BB, AF. 수수수수퍼노바

`math`
출제진 의도 – **Easy**

- 처음 푼 팀(Beginner): **역시내청춘러브캠퍼스는잘못됐다**, 11분
- 처음 푼 팀(Advanced): **신_섭이HCPC를정상화하네**, 5분
- 출제자: 윤아람, 조찬우

- 일종의 진법 변환 문제입니다. $M \leq 26$인 경우, $M > 26$인 경우를 나누어 처리합니다.
- $M \leq 26$이라면 관측 순서는 M번째 알파벳 대문자입니다.
- $M > 26$이라면 관측 순서의 첫 번째 글자는 $\left\lfloor \frac{(M-1)}{26} \right\rfloor$번째, 두 번째 알파벳은 $((M-1) \bmod 26) + 1$번째 소문자입니다.
- C/C++에서 i번째 대문자는 `'A' + i - 1`, i번째 소문자는 `'a' + i - 1`로 구할 수 있습니다.
- Python에서는 `ord`, `chr` 함수를 이용하여 풀 수 있습니다.

BC, AD. 순열 복원

`constructive`
출제진 의도 – **Easy**

- 처음 푼 팀(Beginner): **2호선_잇빗역_기원**, 23분
- 처음 푼 팀(Advanced): **생일축하합니다**, 6분
- 출제자: 김현중

- 어떤 순열 P에 대해서 어떤 i에 대해 $P[i] < P[j]$를 만족하는 j의 개수는 $P[i] - 1$개 존재합니다.
- 따라서 주어진 행렬이 올바른 **비교행렬**이라면 각 행에 대해 -1의 개수를 세주면 됩니다.
- 그렇다면 주이진 행렬이 올바른 **비교 행렬**인지 판단해주면 됩니다.
- 어떤 순열 P에 대해 **비교행렬**을 만들면 다음 조건을 만족합니다.
  - $A[i][j] = -A[j][i]$
  - $A[i][i] = 0$
  - $A[i][j] \neq 0 (i \neq j)$

- 또한 $P[i] < P[j], P[j] < P[k]$ 면 $P[i] < P[k]$ 를 만족해야합니다.$P[i] < P[j]$ 를 만족하는 모든 (i, j) 쌍에 대해 i 에서 j 로 가는 간선으로 하는 방향 그래프를 만들었을 때 사이클이 발생하지 않는 것과 동치입니다.
- 따라서 $A[i][j] = 1$ 을 만족하는 모든 (i, j) 쌍에 대해 i 에서 j 로 가는 간선으로 하는 방향 그래프를 만들었을 때 사이클이 발생하지 않는 것과 동치입니다.
- 위의 4가지 조건을 만족하는지 검사해주면 올바른 **비교 행렬**인지 판단할 수 있습니다.
- 다른 풀이로는 순열을 만들고 순열의 **비교 행렬**과 일치하는지 체크 해주는 풀이가 있습니다.
- 주어진 행렬이 올바른 **비교 행렬**이라고 가정한 후 순열을 만듭니다. 해당 순열로 **비교 행렬**을 만들었을 때 주어진 행렬과 일치하는지 판단하면 됩니다.

BD, AC. 아무나 풀어주세요

`ad-hoc, dynamic-programming`
출제진 의도 – **Hard**

- 처음 푼 팀(Beginner): **N/A**, N/A분
- 처음 푼 팀(Advanced): **팀**, 90분
- 출제자: 구건모

BD, AC. 아무나 풀어주세요

- 문제를 풀기 위해 다음 두 가지 관찰이 필요합니다.
  1. 직전에 추가한 수는 항상 수열의 마지막에 위치한다.
  2. 수를 추가하는 행위로 기존에 존재하는 수열의 마지막 세 수만 영향을 준다.
- 해당 관찰을 통해 B를 시작 수열로 잡고, 맨 뒤 수를 하나씩 삭제하는 연산으로 A를 만드는 문제로 치환할 수 있습니다.
- 따라서, 기존에 존재하는 연산은 다음과 같이 두 연산으로 변경할 수 있습니다.
  1. 맨 뒤 수가 홀수일 경우 그냥 삭제한다.
  2. 맨 뒤 수가 짝수일 경우 삭제한 이후 수열 마지막에 존재하는 세 수는 원하는 대로 정렬할 수 있다. 단, 수를 삭제하기 전, 해당 세 수는 오름차순으로 정렬된 상태여야 한다.

BD, AC. 아무나 풀어주세요

- 예를 들어, 수열 $(\cdots, 4, 5, 6, 7, 8)$에서 8을 삭제하는 연산을 할 수 있으며, 8을 제외한 $(5, 6, 7)$은 정렬된 상태여야 한다. 이후, $(5, 6, 7)$은 $(\cdots, 4, \mathbf{6}, \mathbf{5}, \mathbf{7})$, $(\cdots, 4, \mathbf{7}, \mathbf{5}, \mathbf{6})$ 처럼 어떤 순서로도 재 정렬할 수 있다. 단, $(\cdots, \mathbf{5}, 4, \mathbf{6}, \mathbf{7})$ 처럼 정렬할 수는 없다.
- 수열 $(4, 6, 5, 8)$은 8을 삭제하기 전, $(4, 6, 5)$가 오름차순으로 정렬된 상태가 아니므로 8을 삭제할 수 없다. 따라서, 해당 수열은 더 이상 연산이 불가능한 수열이다.
- 따라서, 다음과 같은 풀이를 생각해 볼 수 있습니다.

BD, AC. 아무나 풀어주세요

1. $A_1, A_2, \cdots, A_{N-3}$ 은 어떤 순서로 연산하더라도 변경되지 않으므로, $A_1, A_2, \cdots, A_{N-3}$ 이 $B_1, B_2, \cdots, B_{N-3}$ 이 같은지 확인한다. 다르다면, 어떻게 연산하더라도 B 를 A 로 만들 수 없다. 이후 $A' = (A_{N-2}, A_{N-1}, A_N); B' = (B_{N-2}, B_{N-1}, \cdots, B_M)$ 으로 설정한다.
2. 수열 B' 부터 시작한다.
3. 현재 수열이 A' 과 같다면 YES 를 출력하고 종료한다. 만약 현재 수열 길이가 3 이지만, A' 과 다르다면 탐색을 종료하고 다른 탐색으로 돌아간다.
4. 현재 맨 마지막 수가 홀수라면, 마지막 수를 제거하고 재귀적으로 새로운 탐색을 시작한다.
5. 짝수라면, 나머지 세 수가 오름차순인지 확인하고, 오름차순이라면 세 수로 표현 가능한 모든 경우의 수로 정렬한 뒤 각각 재귀적으로 새로운 탐색을 시작한다.

- 하지만, 이렇게 탐색하면 최악의 경우 $O(3!^{M-N})$ 이 걸립니다. 탐색이 불가능한 경우를 제외해 커팅을 동반하더라도 충분히 오랜 시간이 걸립니다.
- 따라서, 시간 복잡도를 줄이기 위해 다음 사실에 집중해야 합니다.
- 수열 B' 부터 시작해 수를 지워 만든 길이가 T 인 수열의 앞 $T-3$ 개 수는 항상 고정이다.
- 존재할 수 있는 수의 범위는 1 부터 9 까지이므로, 수열 B' 부터 시작해 수를 지워 만든 길이가 T 인 수열은 최대 9^3 개만 존재할 수 있습니다.
- 따라서, 탐색 과정에서 존재할 수 있는 모든 수열의 가짓수는 9^3M 개라는 사실을 알 수 있습니다.

- 따라서 다음과 같은 점화식을 생각할 수 있습니다.
- $DP[T][a][b][c] = T$ 개 수가 고정이고 마지막 세 수가 각각 (a, b, c) 인 상태에서 수를 더 지워 A' 을 만들 수 있다면 1 아니라면 0
- $DP[0][A'_1][A'_2][A'_3] = 1$
- $DP[T][a][b][c] = DP[T-1][B'_T][b][c]$; if c is odd
- $DP[T][a][b][c] = DP[T-1][B'_T][b][c] \mid dp[T-1][B'_T][c][b] \mid \cdots \mid DP[T-1][c][b][B'_T]$; if c is even and $B'_T \leq a \leq b$

- 각 테스트 케이스별로 위 점화식을 풀고, $M' = len(B')$ 일 때, $DP[M'-3][B'_{M'-2}][B'_{M'-1}][B'_{M'}]$; 이 1 이라면 YES 아니라면 NO 를 출력하면, 문제를 해결할 수 있습니다.
- 총 시간 복잡도는 $O(9^3 3! M)$ 입니다.

BE. 점과 원

`math, geometry`
출제진 의도 – **Hard**

- 처음 푼 팀: **N/A**, N/A분
- 출제자: 이현빈

- 한 번에 네 개의 점에서 같은 거리만큼 떨어진 원을 구하는 것은 직관적이지 않습니다.
- 하지만, 세 개의 점에서 같은 거리만큼 떨어진 원을 구하는 것은 쉽습니다.
- 점 4개를 각각 A, B, C, D라 하고, D를 제외하고 외심을 구한다고 가정하겠습니다.
  - A, B, C를 꼭짓점으로 가지는 삼각형의 외심 O를 구할 수 있습니다.
- A, B, C에서 O를 중심으로 하는 원까지의 거리는 반지름에 상관 없이 같습니다.
- 즉, 원의 중심을 삼각형 ABC의 외심으로 설정하고, 반지름을 적절히 조절한다면 이 문제의 정답을 구할 수 있습니다.

- A, B, C 에서 O 까지의 거리는 같고, 이를 R 이라 하겠습니다.
- D 에서 O 까지의 거리를 r 이라 하겠습니다.
- 중심이 O 인 원이 R 과 r 의 가운데를 지난다면, 원의 반지름은 $\frac{R+r}{2}$ 이 됩니다.
- 이때, 네 점으로부터 원까지의 거리는 $\frac{R+r}{2} - min(R, r)$ 로 같습니다.

- 세 점이 일직선상에 위치하지 않는다면, 세 점을 연결한 삼각형으로부터 항상 외심을 구할 수 있습니다.
- 세 점의 외심은 세 변의 수직이등분선의 교점임을 이용하여 외심의 좌표를 구할 수 있습니다.
- 세 점 A, B, C 가 주어질 때, AB 의 중점 M_{AB} 의 좌표는 $(\frac{x_A + x_B}{2}, \frac{y_A + y_B}{2})$ 입니다.
- 마찬가지로 BC 의 중점 M_{BC} 의 좌표는 $(\frac{x_B + x_C}{2}, \frac{y_B + y_C}{2})$ 입니다.
- 두 점을 지나는 수직이등분선의 방정식은 각각 아래와 같이 표현됩니다.
  - $(y_B - y_A)y + (x_B - x_A)x = \frac{(y_B - y_A)(x_A + x_B) - (x_B - x_A)(y_A + y_B)}{2}$
  - $(y_C - y_B)y + (x_C - x_B)x = \frac{(y_C - y_B)(x_B + x_C) - (x_C - x_B)(y_B + y_C)}{2}$

- 두 식은 $ax + by = c$의 꼴을 가집니다. 아래와 같이 $a_1, b_1, c_1, a_2, b_2, c_2$을 정의하겠습니다.
  - $a_1 = x_B - x_A$
  - $b_1 = y_B - y_A$
  - $c_1 = \dfrac{(y_B - y_A)(x_A + x_B) - (x_B - x_A)(y_A + y_B)}{2}$
  - $a_2 = x_C - x_B$
  - $b_2 = y_C - y_B$
  - $c_2 = \dfrac{(y_C - y_B)(x_B + x_C) - (x_C - x_B)(y_B + y_C)}{2}$

- $\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$ 를 만족하는 x_c, y_c 가 원의 중심이 됩니다.
- x_c, y_c 를 계산하면 아래와 같습니다.
  - $x_c = \dfrac{c_1 b_2 - c_2 b_1}{a_1 b_2 - a_2 b_1}$
  - $y_c = \dfrac{a_1 c_2 - a_2 c_1}{a_1 b_2 - a_2 b_1}$
- (x_c, y_c) 은 구하려는 원의 중심이 됩니다.
- 따라서 중심이 (x_c, y_c) 이고 반지름이 $\dfrac{R+r}{2}$ 인 원이 정답이 됩니다.

BF. 지나칠 수 없는 지하철 게임

ad_hoc
출제진 의도 – **Medium**

- 처음 푼 팀: **2호선_잇빗역_기원**, 56분
- 출제자: 윤아람, 조찬우

BF. 지나칠 수 없는 지하철 게임

- 환승역이 없다면 자명히 선공이 승리합니다.
- 환승역이 하나 이상 있다면 종착역과 가장 가까운 환승역에서 출발하는 사람이 승리합니다.
- 모든 환승역이 붙어 있는 경우를 생각해 봅시다.
  - 환승역의 수가 짝수라면 첫 환승역에서 멈추고 차례가 끝난 사람이 마지막 환승역에서 출발하여 승리합니다.
  - 환승역의 수가 홀수라면 첫 환승역에서 출발한 사람이 마지막 환승역에서도 출발하여 승리합니다.
- 즉, 모든 환승역이 붙어 있고 선공이 첫 환승역 / 첫 환승역 이전 역 중 원하는 역으로 갈 수 있다면 선공이 항상 승리합니다.

BF. 지나칠 수 없는 지하철 게임

- 이는 환승역이 연속된 여러 덩어리로 나뉘어 있을 때도 성립합니다.
- 첫 환승역 덩어리 중 마지막 환승역에서 출발한 사람은 다음 환승역 덩어리의 첫 환승역 / 그 이전 역 중 원하는 역으로 갈 수 있습니다.
- 즉, **첫 환승역 덩어리의 마지막 환승역에서 출발한 사람**은 모든 환승역 덩어리의 마지막 환승역에서 출발할 수 있고, 승리합니다.
- 따라서 2번 역이 환승역이 아니거나, 2번 역부터 연속된 환승역이 짝수 개 있다면 선공인 세훈이가 승리합니다.
- 2번 역부터 연속된 환승역이 홀수 개 있다면 후공인 민아가 승리합니다.
- 연속된 환승역의 수를 $O(N)$에 세어 주면 문제를 풀 수 있습니다.

BG. 컴소인의 크리스마스

`constructive, ad_hoc`
출제진 의도 – **Medium**

- 처음 푼 팀: **N/A**, N/A분
- 출제자: 윤아람, 조찬우

- ✓ $\Sigma s_i \neq N-1$ 이라면 불가능합니다. 총 $N-1$ 번 틀리고 N 번 맞아야 하기 때문입니다.
- ✓ $\Sigma s_i = N-1$ 이라면 항상 가능합니다. 귀납법으로 증명할 수 있습니다.
  - $N=1$ 이라면 $s_1=0$ 이고, 1 번 문제를 바로 맞추면 되니 가능합니다.
  - $N=K-1$ 일 때 가능하다면 $N=K$ 일 때도 가능함을 보입시다.
  - $\Sigma s_i = N-1, s_i \geq 0$ 이니 $s_i=0$ 인 i 가 항상 존재합니다. 풀어 말하면, $\Sigma s_i = N-1$ 이면 바로 맞출 수 있는 문제가 항상 존재합니다.
  - 바로 맞출 수 있는 문제를 맞춰 없애고, 아직 틀리는 횟수가 남은 아무 문제나 틀리면 $N=K-1$ 인 상황이 됩니다. 따라서 $N=K-1$ 일 때 가능하다면 $N=K$ 일 때도 가능합니다.

- 귀납 증명 과정에서 사용한 전략을 그대로 구현하면 $O(N)$ 정도에 문제를 풀 수 있습니다.
- 바로 맞출 수 있는 문제와 한 번 이상 틀려야 맞출 수 있는 문제를 따로 관리하면 구현이 수월합니다.

BH. 코인의 신 건모

`math`, `binary-search`
출제진 의도 – **Easy**

- 처음 푼 팀: **Mongsennam**, 9분
- 출제자: 구건모

- $X\%$ 가 떨어진 금액에서 $Y\%$ 가 오르면 원금이라고 가정해 보겠습니다.
- 정답인 Y 를 찾기 위해 다음과 같은 식을 생각할 수 있습니다.
- 초기 금액 N 에서 $X\%$ 가 떨어진 금액 T 는 다음과 같습니다. $T = (N - N \times \frac{X}{100})$
- 따라서, 다음을 만족하는 Y 를 찾으면 됩니다. $N = T + T \times \frac{Y}{100}$
- 식 정리를 하면 다음과 같습니다. $Y = \frac{100(N-T)}{T} = \frac{100(N-(N-N\times\frac{X}{100}))}{(N-N\times\frac{X}{100})}$
- $= \frac{100(1-(1-\frac{X}{100}))}{1-\frac{X}{100}} = \frac{100-100+X}{1-\frac{X}{100}} = \frac{X}{\frac{100-X}{100}} = \frac{100X}{100-X}$

- 사실, 오차범위가 작아 모든 식 정리를 하지 않고도, N 과 T 에 관한 식으로 Y 를 구해도 문제를 해결할 수 있습니다.
- 또한, 위 풀이 말고도, $N \leq T + T \times \frac{Y}{100}$ 이 식을 만족하면서 가장 작은 Y 를 찾는 이분 탐색으로도 문제를 해결할 수 있습니다.

BI. 피아노

`greedy`
출제진 의도 – **Medium**

- 처음 푼 팀: **변수명지을땐애기이름짓듯이**, 22분
- 출제자: 조찬우

- 누를 건반을 순서대로 보면서, 건반 번호의 최댓값 M 과 최솟값 m 을 관리합시다.
- $M - m \geq K$ 일 때마다 손을 옮기는 것이 최적입니다.
  - $M - m < K$ 라면 해당 음들은 모두 손을 옮기지 않고 칠 수 있습니다. 아니라면, 반드시 손을 옮겨야 합니다.
  - 손을 안 옮겨도 될 때 미리 옮겨서 최소 횟수로 움직일 수 있다면, 미리 옮긴 행동을 옮겨야 할 때 옮기는 행동으로 바꿔도 최소 횟수를 만들 수 있습니다.

BJ, AA. HCPC 팀 짜기

`math, case_work, combinatorics`
출제진 의도 – **Hard, Medium**

- 처음 푼 팀(Beginner): **N/A**, N/A분
- 처음 푼 팀(Advanced): **다음엔 너가 신청해라**, 70분
- 출제자: 김동민, 김현중

- 편의를 위해 혼자 나갈 수 있거나, N명이 서로 꼬리를 물어 동시에 나갈 수 있는 상황을 '사이클'로 나타내겠습니다.
- 문제를 풀기 위해 크기가 1, 2, 3인 사이클을 미리 알아둘 필요가 있습니다.
- HCPC에 나갈 3명을 뽑을 수 있는 7가지 경우의 수가 존재합니다.
- 각각의 경우는 다음과 같으며, 이들을 다 합하여 주면 됩니다.

BJ, AA. HCPC 팀 짜기

1. 크기 1의 사이클 3개
2. 꼬리가 달린 크기 1의 사이클 1개와, 해당 사이클과 다른 크기 1의 사이클 1개
3. 꼬리 2개가 병렬적으로 달린 크기 1의 사이클 1개
4. 꼬리 2개가 수직적으로 달린 크기 1의 사이클 1개
5. 크기 2의 사이클 1개와, 크기 1의 사이클 1개
6. 꼬리가 1개 달린 크기 2의 사이클 1개
7. 크기 3의 사이클

AB. 흑백조경사

`trees`, `dp_tree`, `disjoint_set`
출제진 의도 – **Hard**

- 처음 푼 팀: **생일축하합니다**, 53분
- 출제자: 조찬우

- 흑백 중 더 많은 색깔만 뿌리가 될 수 있습니다. 흑백 개수가 같다면 어떤 정점도 뿌리가 될 수 없습니다.
  - 모든 정점은 뿌리 정점의 자손이기 때문입니다.
- 모든 간선 (u, v)에 대해 u가 v의 부모일 수 있는지, v가 u의 부모일 수 있는지를 검사합시다.
  - 이는 자손의 검은색/흰색 정점 수를 모두 세어 주면 구할 수 있습니다.
- u가 v의 부모일 수 **없다면** 간선에 $v \to u$ 방향을 줍시다. 반대도 마찬가지입니다. 이는 v가 u의 부모여야 한다는 의미입니다.

- 모든 간선에 방향을 부여할지 결정한 이후, 두 가지 경우로 나눌 수 있습니다.
  1. 마주보는 방향 간선이 있다면 어떤 정점도 뿌리가 될 수 없습니다.
  2. 마주보는 방향 간선이 없다면, 자신으로 향하는 방향 간선이 있는 정점은 뿌리가 될 수 없습니다.
- 뿌리가 될 수 있는 정점은 2번 경우에 해당하지 않는 정점 중 뿌리가 될 수 있는 색깔인 정점입니다.
- 간선에 방향이 부여되지 않았을 때 분리 집합으로 두 정점을 묶으면 구현이 수월합니다.

AE. 수열과 쿼리 HY

`math`
출제진 의도 – **Hard**

- 처음 푼 팀: **다음엔 너가 신청해라**, 40분
- 출제자: 조찬우

- A의 순서는 답과 상관없으니 A를 오름차순으로 정렬합시다.
- m으로 나눈 나머지가 가장 작은 수는 정렬된 A에서 m의 배수 이상이면서 가장 작은 수 중 하나입니다.
- 비슷하게, 나머지가 가장 큰 수도 m의 배수 미만이면서 가장 큰 수 중 하나입니다.
- A에서 m의 배수를 이분 탐색하면 m의 배수 앞뒤에 있는 수를 $O(\log N)$에 구할 수 있습니다.
- 따라서 수 범위 이내의 m의 배수를 이분 탐색하면 m에 대한 문제를 풀 수 있습니다.

- m의 배수는 대략 $\frac{300\,000}{m}$개 정도이므로, m에 대한 정답을 $O(\frac{300\,000 \log N}{m})$ 정도에 계산할 수 있습니다.
- $\frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{m} = O(\log m)$ 이므로,

  $\sum_{m=1}^{300\,000} \frac{300\,000 \log N}{m} = 300\,000 \log N \sum_{m=1}^{300\,000} \frac{1}{m} = O(300\,000 \log 300\,000 \log N)$ 입니다.
- 따라서 $1 \le m \le 300\,000$인 모든 m에 대한 정답을 충분히 빠르게 계산하여 문제를 풀 수 있습니다.

AG. 돌 게임 nm

$\mathrm{ad}_h oc$

출제진 의도 - **Easy**

- 처음 푼 팀: **생일축하합니다**, 9분
- 출제자: 김현중

- 주어진 게임은 (n, m)에서 하나의 값에 1을 빼서 두 값중 하나를 0으로 먼저 만드는 사람이 이기는 게임과 동치입니다.
- 일반성을 잃지 않고 $n \leq m$이라 가정합시다.
- $n = 1$인 경우에는 선공이 한행을 가져가면 게임이 끝나므로 항상 선공이 승리합니다.
- $n > 1, n + m = 2k$인 경우는 후공이 항상 승리합니다.
- $n > 1, n + m = 2k + 1$인 경우는 선공이 항상 승리합니다.

- 이는 수학적 귀납법으로 증명할 수 있습니다.
- $(2, 2)$ 인 경우는 선공의 차례에서 1 이 생기므로 후공이 항상 승리합니다.
- 합이 짝수인 임의의 (n, m) 에 대해, $\{(i, j) : i + j = 2k, (i, j) < (n, m), i \neq 1, j \neq 1\}$ 인 집합에 대해 후공이 항상 승리한다고 가정합시다.
- (n, m) 에서 턴이 두번 지나면 $(n - 1, m - 1)$, $(n - 2, m)$, $(n, m - 2)$ 중 하나의 상태가 됩니다. 후공은 집합에 포함되게 상태를 고를 수 있으므로 후공의 승리가 증명됩니다.
- $n + m = 2k + 1$ 인 경우에는 한턴이 지났을 때 선공이 후공인 상태가 되고 $n + m = 2k$ 인 상태가 되므로 선공이 항상 승리합니다.

AH. 덧셈 팰린드롬 수열과 트리

`ad-hoc`, `LCA`, `bruteforce`
출제진 의도 – **Medium**

- 처음 푼 팀: **N/A**, N/A분
- 출제자: 구건모

- 이 문제를 풀기 위해선 수열이 주어졌을 때, 덧셈 팰린드롬 수열인지 아닌지 판별할 수 있어야 합니다.
- 먼저, 다음과 같이 길이가 4 이상인 팰린드롬 수열을 생각해 보겠습니다. $(a, b, \cdots, b, a)$
- 해당 수열은 문제에서 정의한 덧셈 연산을 통해 항상 $(a, b + \cdots + b, a)$ 로 변경할 수 있습니다.
- 이 관찰을 통해, 어떤 초기 수열이 주어지더라도 덧셈 팰린드롬을 만족한다면, 덧셈 연산을 반복해서 길이가 3 이하인 덧셈 팰린드롬 수열로 바꿀 수 있다는 사실을 알 수 있습니다.
- 따라서, 덧셈 연산 이후 길이가 2 혹은 3인 팰린드롬 수열로 만들 수 있는지 검사하는 것으로 덧셈 팰린드롬인지 아닌지 알 수 있습니다.

- 모든 경우의 수를 살펴보면 다음과 같습니다.
- $(A_1, A_2, \cdots, A_N) \rightarrow (\sum_{i=1}^{k} A_i, \sum_{i=k+1}^{N} A_i)\ or\ (\sum_{i=1}^{l-1} A_i, \sum_{i=l}^{r} A_i, \sum_{i=r+1}^{N} A_i)$
- 최종적으로 합쳐진 수열에서 양 끝 두 수가 같다면 팰린드롬입니다.
- 따라서, 초기 수열 A에서 $l < r$을 만족하면서, $\sum_{i=1}^{l} A_i = \sum_{i=r}^{N} A_i$인 경우가 존재한다면, 해당 수열은 덧셈 팰린드롬 수열임을 알 수 있습니다.

- prefix sum 등을 활용해 모든 $l < r$에 대해 탐색하면 $O(N^2)$ 시간 복잡도로 해당 수열이 덧셈 팰린드롬 수열인지 판별할 수 있습니다. 단, 음수가 포함되어 있어 단조성이 보장되지 않으므로 투 포인터 같은 풀이로는 해결할 수 없습니다.
- 이 연산을 트리에서 모든 리프 쌍의 단순 경로에 대해 진행하면 됩니다.
- 두 리프 (u_1, u_2) 쌍의 LCA를 구해 $u_1, par[u_1], \cdots, LCA(u_1, u_2), \cdots .par[u_2], u_2$ 순서대로 수열을 구성하면, 단순 경로를 구할 수 있습니다.
- 또한, 두 리프의 LCA와 경로는 u_1, u_2가 같아질 때까지 두 수를 모두 2로 나누면 (= 부모로 이동하면) 쉽게 구할 수 있습니다.

- 포화 이진 트리의 리프는 2^{h-1} 개 존재합니다. 또한, 모든 경로의 길이가 최대 $2h-1$ 입니다.
- 따라서, 모든 리프의 쌍 탐색은 $O((2^{h-1})^2)$ 이고, 각 탐색별로 덧셈 팰린드롬 수열 판별은 $O((2h-1)^2)$ 이므로 $O((2^{h-1})^2 \times (2h-1)^2)$ 시간복잡도에 문제를 해결할 수 있습니다.
- 또한, BBST 같은 자료구조를 활용하면, $O((2^{h-1})^2 \times (2h-1)\log_2(2h-1))$ 로도 줄일 수 있지만, $2h-1$ 이 충분히 작아, 대개 상황에서 BBST를 적용하지 않은 풀이가 더 빠르게 동작합니다.

AI. 나연 정렬

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

- 처음 푼 팀: **뚜따뚜우따이**, 10분
- 출제자: 조찬우

- 스택이 필요할 때마다 생성한다고 생각하고 나연 정렬을 직접 수행해 봅시다.
- 오름차순으로 정렬하려면, 각 스택이 내림차순으로 정렬되어 있어야 합니다. 따라서 스택의 맨 위가 넣을 수 이상일 때만 넣을 수 있습니다.
- 현재 수를 넣을 수 있는 스택이 없다면, 스택을 새로 만들어야 합니다. 현재 수를 이미 존재하는 스택에 넣을 수 있다면, 무조건 넣는 것이 최적입니다.
  - 넣을 수 있는 스택이 여러 개라면, 스택 맨 위가 가장 작은 스택에 넣는 것이 최적입니다.
- 스택의 꼭대기를 `set` 등으로 관리하면 넣어야 할 스택을 찾을 수 있고, $O(N \log N)$ 정도에 문제를 풀 수 있습니다.

- ✓ 답을 구하는 과정을 잘 살펴보면, LIS를 구하는 과정과 같습니다.
- ✓ 실제로 필요한 스택의 최소 개수는 LIS의 길이와 같습니다. 증명은 여러분께 맡기겠습니다.

AJ. 간선을 하나 그어서 루트까지 거리의 합을 최소로 만들기로 했습니다

`math`, `dfs`, `prefix_sum`, `binary_search`
출제진 의도 – **Challenging**

- 처음 푼 팀: **N/A**, N/A분
- 출제자: 김현중

- $dist(u, v)$: u과 v의 최단거리
- $sz(v)$: 정점 v를 루트 정점으로 하는 서브트리 크기
- $sz(v, i)$: $sz(v)$ − 정점 i를 갖는 자식 트리의 크기
- 관찰 1
  - 루트로 연결하는 것이 최적임을 관찰할 수 있습니다.
  - 일반성을 잃지 않고 $dist(1, u) \leq dist(1, v)$인 정점 쌍 (u, v)을 골라 간선을 추가합시다. $dist(1, u) \leq dist(1, v)$이므로 $dist(x, u) + dist(u, v) + dist(v, 1) < dist(x, u) + dist(u, 1)$인 정점 x는 존재하지 않습니다다. 따라서 최적인 정점을 고르려면 $dist(u)$가 가장 작은 u를 고르면 되는데 이는 루트 정점인 1입니다.

AJ. 간선을 하나 그어서 루트까지 거리의 합을 최소로 만들기로 했습니다

- 그렇다면 각 정점마다 루트로 연결했을 때 거리가 얼마나 줄어드는지 계산하면 됩니다.
- 어떤 정점 v에서 1번 정점과 연결했을 때 u번 정점이 줄어드는 거리는 $max(0, dist(1,u) - dist(v,u))$ 입니다.
- p를 u, v의 최소공통조상이라고 했을 때, $max(0, dist(1,u) - dist(v,u)) = max(0, dist(1,p)+dist(p,u)-(dist(p,v)+dist(p,u))) = max(0, dist(p,1)-dist(p,v))$ 입니다.
- 따라서 v의 조상 정점들에 대해서만 거리를 관리해주면 됩니다.
- P_v를 정점 v의 조상 정점 집합이라고 할 때
- $\sum_{p \in P_v} max(0, dist(p,v) - dist(p,1)) sz(p,i)$ 로 줄어드는 거리를 계산할 수 있습니다.

AJ. 간선을 하나 그어서 루트까지 거리의 합을 최소로 만들기로 했습니다

- $dist(p,v)-dist(p,1) > 0$ 인 v의 조상 정점을 찾아준다면
- $\sum_{p \in P_v} (dist(p,v) - dist(p,1))sz(p,i)$ 로 식을 단순화 시킬 수 있습니다. 조건을 만족하는 조상 정점을 찾는 것은 이분탐색으로 $O(logN)$의 시간 복잡도로 찾을 수 있습니다.
- 더 식을 정리하면
- $\sum_{p \in P_v} (dist(p,v) - dist(p,1))sz(p,i) = \sum_{p \in P_v} (dist(1,v) - dist(1,p) - dist(p,1))sz(p,i) = dist(v,1) \sum_{p \in P_v} sz(p,i) - 2 \sum_{p \in P_v} dist(p,1)sz(p,i)$ 가 됩니다.
- 각 시그마 항을 dfs를 돌리면서 누적합으로 관리하면 $O(1)$의 시간에 계산할 수 있습니다.
- 각 정점마다 구하므로 전체 시간복잡도는 $O(NlogN)$ 입니다.