2026 UDPC

Official Solutions

UDPC 2026 - 출제진 및 검수진

- ✓ 출제진
  - 권찬우 kwoncycle
  - 곽민성 minsung05
  - 김도훈 99asdfg
  - 박서진 haruki201sa
  - 박신욱 bnb2011
  - 우은규 ekwoo
  - 이예승 ia0102
  - 이우진 0917ba
  - 장래오 leo020630
  - 최진우 andrew1010
  - 한동규 queued_q

- ✓ 검수진
  - jinhan814
  - jthis
  - pyb1031
  - utilforever

Senior Division	문제	의도한 난이도	출제자
A	대회 장소 정하기	**Easy**	박신욱 bnb2011
B	포닉스와 달구 2	**Easy**	이우진 0917ba
C	U-Deep-See	**Medium**	장래오 leo020630
D	카드 섞기 게임	**Medium**	이예승 ia0102
E	풍선줄 자르기	**Medium**	한동규 queued_q
F	문자열과 문자열과 쿼리	**Medium**	장래오 leo020630
G	송신탑 설치	**Hard**	한동규 queued_q
H	UDPC 온라인 게임	**Hard**	김도훈 99asdfg
I	포닉스의 트리 세기	**Hard**	곽민성 minsung05

Junior Division	문제	의도한 난이도	출제자
A	대회 장소 정하기	**Easy**	박신욱bnb2011
B	Swap-LIS	**Easy**	이우진0917ba
C	포닉스와 달구 2	**Easy**	이우진0917ba
D	U-Deep-See	**Medium**	장래오leo020630
E	카드 섞기 게임	**Medium**	이예승ia0102
F	바람에 몸을 맡기다	**Medium**	한동규queued_q
G	쉽고 간단한 문제	**Medium**	이우진0917ba
H	UDPC 온라인 게임	**Hard**	김도훈99asdfg
I	달구와 윤이의 카드 배틀	**Hard**	김도훈99asdfg

1A/2A. 대회 장소 정하기

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

- ✓ 통계:
  - Junior: 제출 52번, 정답 16번, 정답률 30.769%
  - Senior: 제출 32번, 정답 16번, 정답률 50.000%
- ✓ First Solve:
  - Junior: **BonGyejeong**[POSTECH], 11분
  - Senior: **——IQ100——**[POSTECH], 9분
- ✓ 출제자: 박신욱[bnb2011]

1A/2A. 대회 장소 정하기

- 원형 도로 위의 한 지점이 다른 어떤 지점으로부터 시계 방향으로 d만큼 떨어져 있다고 합시다.
- 두 지점 간의 **거리**는 $\min(d, N-d)$와 같이 계산할 수 있습니다.
- 원형 도로 위 적당한 기준점을 정했을 때, 모든 학교의 위치와 대회장의 위치는 해당 기준점으로부터 시계 방향으로 정수 길이만큼 떨어진 지점에 있다고 가정할 수 있습니다.
- 각 학교의 위치 및 대회장의 위치를 정하는 모든 경우의 수는 $\mathcal{O}\left(N^4\right)$ 이고, 각 경우가 올바른지는 $\mathcal{O}\left(1\right)$ 시간에 계산 가능합니다.
- 따라서 $\mathcal{O}\left(N^4\right)$ 시간에 문제를 해결할 수 있습니다.
- 각 위치를 어떻게 정하느냐에 따라 $\mathcal{O}\left(N^3\right), \mathcal{O}\left(N\right), \mathcal{O}\left(1\right)$ 시간에도 문제를 해결할 수 있습니다.

2B. Swap-LIS

Ad-Hoc

출제진 의도 – **Easy**

- 통계:
  - Junior: 제출 40번, 정답 10번, 정답률 25.000%
- First Solve:
  - Junior: **syeyoung**[POSTECH], 21분
- 출제자: 이우진[Ø917ba]

2B. Swap-LIS

- ✓ 주어진 순열이 증가 수열이라면 어떤 두 원소를 골라 교환하더라도 LIS의 길이가 감소하지 않도록 할 수 없습니다.
- ✓ 그렇지 않을 경우, $A_i > A_{i+1}$ 인 어떤 i 를 골라 A_i 와 A_{i+1} 을 교환하면 됩니다.

1B/2C. 포닉스와 달구 2

`game_theory`, `case_work`
출제진 의도 – **Easy**

- 통계:
  - Junior: 제출 41번, 정답 10번, 정답률 24.390%
  - Senior: 제출 53번, 정답 15번, 정답률 28.302%
- First Solve:
  - Junior: **syeyoung**[POSTECH], 42분
  - Senior: **36ismyfavnumber**[UNIST], 16분
- 출제자: 이우진[`0917ba`]

1B/2C. 포닉스와 달구 2

- 남아있는 칸이 두 칸 이하인 경우, 게임을 진행했을 때 달구의 턴이 오지 않거나 달구의 턴을 마쳤을 때 항상 게임판이 대칭을 이루므로 달구가 승리합니다. 따라서 남아있는 칸이 세 칸 이상인 경우만 생각합시다.
- N에 대한 홀짝성을 나누어 생각해볼 수 있습니다.

1B/2C. 포닉스와 달구 2

- ✓ N 이 짝수일 경우, 게임판이 초기에 대칭이면 달구가 승리하고, 그렇지 않으면 포닉스가 승리합니다.
  - 게임판이 초기에 대칭 상태면, 포닉스가 둔 위치의 반대편에 달구가 돌을 놓는 전략을 사용함으로써 게임판을 항상 대칭 상태로 유지할 수 있습니다.
  - 그렇지 않은 경우, 포닉스가 항상 첫 수만에 달구가 다음 턴에 대칭을 만들지 못하도록 할 수 있습니다.

1B/2C. 포닉스와 달구 2

- ✓ N 이 홀수일 경우, 게임판이 초기에 대칭이며 가운데 칸이 이미 차 있을 경우 달구가 승리하고, 그렇지 않으면 포닉스가 승리합니다.
  - 게임판이 초기에 대칭 상태이며 가운데 칸이 차 있을 경우, 짝수인 경우와 비슷하게 대칭 전략을 사용하여 달구가 게임판을 항상 대칭 상태로 유지할 수 있습니다.
  - 게임판이 초기에 대칭 상태이지만 가운데 칸이 비어있을 경우, 첫 턴에 포닉스가 가운데 칸에 착수하게 되면 달구는 다음 턴에 항상 게임판을 대칭 상태로 만들 수 없게 됩니다.
  - 게임판이 초기에 대칭이 아닌 경우, 포닉스가 항상 첫 수만에 달구가 다음 턴에 대칭을 만들지 못하도록 할 수 있습니다.

1B/2C. 포닉스와 달구 2

- 정리하면 달구의 승리 조건은 (초기에 빈 칸이 2개 이하) $\vee$ {(초기 게임판이 좌우 대칭) $\wedge$ (N이 짝수거나 N이 홀수인데 가운데 칸이 이미 차 있음)} 입니다.
- 이외의 경우에는 포닉스가 승리합니다.

1C/2D. U-DeeP-See

`constructive`
출제진 의도 – **Medium**

- 통계:
  - Junior: 제출 13번, 정답 7번, 정답률 53.846%
  - Senior: 제출 23번, 정답 12번, 정답률 52.174%
- First Solve:
  - Junior: **튀긴상추**[POSTECH], 40분
  - Senior: **36ismyfavnumber**[UNIST], 39분
- 출제자: 장래오[`leo020630`]

- $K = 1$ 이면 자명하게 불가능합니다.
- $K \geq 2$ 이면 항상 가능합니다.
- 각 부분 격자에 속한 칸의 합을 정확히 1씩만 증가시키는 방법을 생각해 봅시다.
- $(aK + 1, bK + 1)$ 꼴인 칸에 1을 더하면 각 부분 격자에 속한 칸의 합 역시 1씩 증가하게 됩니다.
- 같은 방법으로, $(cK + 2, dK + 1)$ 꼴인 칸에 1을 빼면 각 부분 격자에 속한 칸의 합 역시 1씩 감소하게 됩니다.
- 둘을 동시에 적용하면 각 부분 격자에 속한 칸의 합은 변화하지 않게 됩니다.
- 이외에도 여러 방법으로 문제를 해결할 수 있습니다.

1D/2E. 카드 섞기 게임

`permutation_cycle_decomposition`, `sorting`
출제진 의도 – **Medium**

- 통계:
  - Junior: 제출 9번, 정답 7번, 정답률 77.778%
  - Senior: 제출 22번, 정답 11번, 정답률 50.000%
- First Solve:
  - Junior: **syeyoung**[POSTECH], 66분
  - Senior: **36ismyfavnumber**[UNIST], 49분
- 출제자: 이예승[`ia0102`]

1D/2E. 카드 섞기 게임

- ✓ 순열 사이클 분할로 순열을 여러 사이클로 나눌 수 있습니다.
- ✓ 아직 방문하지 않은 각 원소에 대해서 순열을 따라가면서 리스트에 원소를 저장하고 이미 방문한 원소가 나오면 한 사이클을 찾은 것입니다. 이를 반복하면 됩니다.
- ✓ 사이클의 길이를 L이라 했을때, 해당 사이클의 p번째 원소에 해당하는 알파벳은 $(p+k) \bmod L$번째 사이클 원소에 해당하는 위치로 이동합니다.
- ✓ 이를 이용하여, 모든 초기위치 i에 대해서 k번 섞으면 어디로 가는지를 $O(N)$에 모두 구할 수 있습니다!

1D/2E. 카드 섞기 게임

- S를 정렬한 다음 미리 구해둔 섞은 뒤의 결과 위치를 이용해서 앞에서부터 하나씩 배치하면 됩니다.
- 시간 복잡도는 정렬을 counting sort로 구현시 $O(N)$, std::sort등으로 구현시 $O(N \log N)$ 입니다.

2F. 바람에 몸을 맡기다

ad_hoc
출제진 의도 – **Medium**

- 통계:
  - Junior: 제출 1번, 정답 1번, 정답률 100.000%
- First Solve:
  - Junior: **syeyoung**[POSTECH], 86분
- 출제자: 한동규[queued_q]

2F. 바람에 몸을 맡기다

- ✓ 문제에 주어진 이동 방법으로 BFS를 수행하면 $O(THW)$로 시간 초과가 납니다.
- ✓ 대신, 목적지부터 역순으로 이전 위치를 복원해 나갈 수 있습니다.
  - 바람의 반대 방향에 건물이 없는 경우, 이전에 반드시 그곳에서 날아왔어야 합니다.
  - 바람의 반대 방향에 건물이 있는 경우, 이전에 반드시 건물을 붙잡고 제자리에 있었어야 합니다.
- ✓ 따라서, 역순으로 이전 위치를 복원해 나갔을 때 가장 처음 위치가 시작 위치와 동일하면 답은 YES 입니다. 각 단계에서 바람을 타고 날아왔는지 건물을 붙잡았는지를 기록해 두면 윤이의 계획을 구할 수 있습니다.
- ✓ 만약 복원 도중에 격자를 벗어나거나 복원한 처음 위치가 시작 위치와 다르면 답은 NO 입니다.
- ✓ 시간복잡도는 $O(HW + T)$ 입니다.

2G. 쉽고 간단한 문제

`dp, greedy`
출제진 의도 – **Medium**

- 통계:
  - Junior: 제출 13번, 정답 5번, 정답률 38.462%
- First Solve:
  - Junior: **syeyoung**[POSTECH], 106분
- 출제자: 이우진[`0917ba`]

2G. 쉽고 간단한 문제

- 다음과 같은 LIS를 구하는 DP와 비슷한 형태의 점화식을 생각해봅시다.
  - $dp(i)$: A_i를 오른쪽 끝으로 하는 조건을 만족하는 부분 수열의 최대 길이.
  - $dp(i) = \max_{1 \le j < i,\, |A_i - A_j| \neq 1} dp(j) + 1$
- 시간복잡도 $O(N^2)$으로 모든 i에 대해 위 dp 값들을 모두 계산할 수 있고, 그중 최댓값이 답이 됩니다.
- 하지만 $N \le 500\,000$으로 최적화가 필요합니다. 어떻게 해야 할까요?

2G. 쉽고 간단한 문제

- 사실, $dp(i)$를 계산하는 데에 $1 \leq j < i$인 모든 j를 살펴보지 않아도 됩니다.
- $|A_i - A_j| = 1$인 A_j는 최대 2개 존재하므로, 매번마다 $dp(j)$가 큰 순서대로 최대 3개의 $(A_j, dp(j))$ 쌍만 봐도 됩니다.
- std::map과 같은 자료구조에 $(A_j, dp(j))$ 쌍을 저장해놓고, 매 iteration마다 map의 크기가 3 이하가 되도록 잘라주면 시간복잡도 $O(N)$에 문제를 해결할 수 있습니다.
- std::multiset, segment tree와 같은 자료구조를 사용하는 $O(N \log N)$ 풀이도 존재합니다.

1E. 풍선줄 자르기

`dijkstra`
출제진 의도 – **Medium**

- 통계:
  - Senior: 제출 17번, 정답 7번, 정답률 41.176%
- First Solve:
  - Senior: **36ismyfavnumber**[UNIST], 72분
- 출제자: 한동규[queued_q]

1E. 풍선줄 자르기

- 풍선의 높이는 실을 통해 고리까지 이동하는 최단 거리와 같습니다.
- 먼저 다익스트라 알고리즘을 한 번 수행해서 고리에서 각 풍선까지의 최단 거리를 계산합니다. 풍선 u까지의 거리를 d_u라고 표현합시다.
- 이제 각 풍선을 지탱하는 팽팽한 실을 찾아야 합니다. 길이 w의 실이 풍선 u와 풍선 또는 고리 v를 연결할 때, $d_u = d_v + w$를 만족하면 해당 실은 풍선 u를 팽팽하게 붙잡고 있습니다.
- 각 풍선 u의 아래 방향으로 연결된 실 중에서 팽팽하면서 가장 긴 것만 남기면 됩니다.
- 각 풍선을 아래 방향으로 잇는 실이 하나만 남으므로, $M - N$개의 실을 자르고 N개의 실이 남습니다. 참고로 이는 최단 경로 트리 구조를 이룹니다.
- 시간복잡도는 $O(M \log N)$입니다.

1F. 문자열과 문자열과 쿼리

dp
출제진 의도 – **Medium**

- ✓ 통계:
  - Senior: 제출 7번, 정답 1번, 정답률 14.286%
- ✓ First Solve:
  - Senior: **outfinity**[UNIST], 176분
- ✓ 출제자: 장래오[leo020630]

1F. 문자열과 문자열과 쿼리

- 두 문자열의 최장 공통 연속 부분문자열의 길이를 구하는 방법은 무엇일까요?
- Suffix Array와 LCP 배열을 이용해 $\mathcal{O}(N \log N)$에 해결하는 방법이 널리 알려져 있지만, 더 간단한 방법으로도 해결할 수 있습니다.
- $dp(i, j)$를 A의 i번째 문자를 끝으로, B의 j번째 문자를 끝으로 하는 공통 연속 부분문자열의 최대 길이로 정의합시다.
- 구체적으로는, $dp(i, j) = \begin{cases} dp(i-1, j-1) + 1 & \text{if } A_i = B_j \\ 0 & \text{otherwise} \end{cases}$ 와 같이 계산할 수 있습니다.
- 따라서 $\mathcal{O}(|A||B|)$에 문제를 해결할 수 있습니다.

- 이를 이용해 문제를 해결해 봅시다.
- 만약 모든 $x = 1$ 이라고 가정하면 앞에서 설명한 DP의 정의를 그대로 사용해 문제를 해결할 수 있습니다.
- 이를 조금 응용해 $x > 1$ 일 때의 문제도 해결해 봅시다.
- 우선, 각 문자열을 연속된 같은 문자들의 블럭으로 표현합시다.
- 예를 들어, 문자열 A 는 문자 C_1^A 이 X_1^A 개, 문자 C_2^A 가 X_2^A 개, $\cdots$, 문자 C_N^A 이 X_N^A 개 있는 형태의 문자열입니다.

- 이렇게 정의하고 나면 $dp(i, j)$ 를 A 의 i 번째 블럭의 마지막 문자와 B 의 j 번째 블럭의 마지막 문자를 끝으로 하는 공통 연속 부분문자열의 최대 길이로 정의할 수 있습니다.
- 구체적으로는, $dp(i, j) = \begin{cases} dp(i-1, j-1) + X_i^A & \text{if } C_i^A = C_j^B, X_i^A = X_j^B \\ 0 & \text{otherwise} \end{cases}$ 와 같이 계산할 수 있습니다.
- 단, 블럭의 마지막 문자를 끝으로 하는 경우가 답이 아닐 수 있으므로 모든 (i, j) 쌍에 대해 $C_i^A = C_j^B$ 라면 $dp(i-1, j-1) + min(X_i^A, X_j^B)$ 를 답의 후보로 고려해야 합니다.
- 연속된 문자가 추가되는 것에 유의하여 DP 배열을 갱신하면 $\mathcal{O}\left(Q^2\right)$ 에 문제를 해결할 수 있습니다.

1G. 송신탑 설치

`greedy`, `stack`
출제진 의도 – **Hard**

- 통계:
  - Senior: 제출 3번, 정답 0번, 정답률 0.000%
- First Solve:
  - Senior: -
- 출제자: 한동규[queued_q]

1G. 송신탑 설치

- 송신탑을 1 번부터 순서대로 최대한 왼쪽으로 밀착시켜 배치하는 그리디 전략으로 문제를 해결할 수 있습니다.

1G. 송신탑 설치

- ✓ 현재 $i-1$번 송신탑까지 설치했고, i번 송신탑을 설치할 차례라고 합시다.
- ✓ 이미 설치된 두 송신탑 $k<j$에 대해서, k번 송신탑의 우선순위가 더 낮다면 $(p_k<p_j)$ k번 송신탑을 고려할 필요가 없습니다.
  - $p_i>p_k$ 이면 $x_k+R_k\le x_j<x_i$ 이므로 k번 송신탑이 i번 송신탑에 간섭을 일으키지 않고,
  - $p_i<p_k$ 이면 $x_k+R_k\le x_j\le x_i-L_i$ 이므로 k번 송신탑과 i번 송신탑의 간섭 범위가 겹치지 않기 때문입니다.

1G. 송신탑 설치

- 이처럼 $k < j$ 이고 $p_k < p_j$ 일 때, k 번 송신탑이 j 번 송신탑에 의해 가려진다고 표현합시다.
- i 번 송신탑을 설치할 때, 이미 설치된 송신탑 중 "다른 송신탑에 가려지지 않는" 것만 고려해도 됩니다.
  - 이러한 송신탑 집합을 S_i 라고 합시다.
  - S_i 의 송신탑은 오른쪽으로 갈수록 우선순위가 낮아지게 됩니다.

1G. 송신탑 설치

- 한편, 어떤 두 송신탑 $k < i$에 대해 다음이 성립합니다. 이를 최소 거리 부등식이라고 합시다.

$$x_i \geq x_k + \min(R_k, L_i)$$

- 모든 $k \in S_i$에 대해 최소 거리 부등식을 만족하는 가장 작은 x_i를 찾으면, 그곳이 최적의 설치 위치가 될 것입니다.

1G. 송신탑 설치

- ✓ 송신탑 $k \in S_i$ 를 오른쪽부터 확인합시다.
- ✓ 만약 $R_k \leq L_i$ 라면,
  - $p_k < p_i$ 로 설정했을 때 간섭이 없을 조건 $x_k + R_k \leq x_i$ 가 최소 거리 부등식과 일치합니다.
  - 즉, x_i 를 최소화한다는 목표에 방해가 되지 않습니다.
  - 따라서 i 번 송신탑의 우선 순위를 높게 설정하고 넘어가도 괜찮습니다.
  - 이 과정을 반복하면서 x_i 의 하한인 $x_k + R_k$ 의 최댓값을 관리합니다.

- ✓ 그러다가 처음으로 $R_k > L_i$ 인 송신탑을 마주치면,
  - $p_k > p_i$ 로 설정했을 때 간섭이 없을 조건 $x_k + L_i \leq x_i$ 가 최소 거리 부등식과 일치합니다.
  - 즉, 우선순위를 더 낮게 설정해야 x_i 를 최소화한다는 목표에 방해가 되지 않습니다.
  - 더 왼쪽에 있는 송신탑 $k' < k$ 에 대해서도 자동적으로 $p_{k'} > p_k > p_i$ 와 $x_{k'} + L_i < x_k + L_i \leq x_i$ 를 만족하므로 간섭이 일어나지 않습니다.
- ✓ 따라서 x_i 를 앞서 관리한 하한 중 최댓값으로 설정하면 최적의 위치에 송신탑을 설치할 수 있습니다.

1G. 송신탑 설치

- S_i를 스택으로 관리하며 우선 순위가 p_i보다 낮은 송신탑을 pop하면 앞선 과정을 진행하는 동시에 S_{i+1}을 계산할 수 있습니다.
- 한편, 우선 순위 p_i를 설정할 때 이미 설치한 두 송신탑의 우선 순위 사이에 끼워넣어야 한다는 문제가 있습니다.
- 이를 단순히 우선 순위 값을 1씩 뒤로 미는 방식으로 구현하면 시간 초과가 납니다.
- 송신탑의 우선 순위를 설치 시 곧장 설정하는 것이 아니라, 스택에서 pop할 때만 설정한다면 이 문제를 해결할 수 있습니다. 스택에서 pop이 일어날 때마다 지금까지 pop을 수행한 횟수만큼 우선 순위를 설정해 주면 됩니다.
- 전체 시간복잡도는 $O(N)$입니다.

1H/2H. UDPC 온라인 게임

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

- ✓ 통계:
  - Junior: 제출 2번, 정답 0번, 정답률 0.000%
  - Senior: 제출 2번, 정답 0번, 정답률 0.000%
- ✓ First Solve:
  - Junior: -
  - Senior: -
- ✓ 출제자: 김도훈[99asdfg]

1H/2H. UDPC 온라인 게임

- ✓ 0번 정점을 루트로 하고, 각 던전을 정점, 선행 던전이 다음 던전의 부모 정점으로 하는 트리를 구성해 봅시다.
- ✓ 선행 던전이 없는 던전은 단순히 0번 정점을 부모로 가지도록 하면, 포레스트가 아닌 트리를 구성할 수 있습니다.
- ✓ 각 플레이어가 각자 단 한 번만 보스 던전을 클리어하는 것이 최적이므로, 각 플레이어가 클리어하고자 하는 보스 던전을 하나씩만 골라 곧장 해당 보스 던전을 클리어하러 가도록 하는 전략이 최적의 전략입니다.
- ✓ 이때, 깊이가 K보다 깊은 보스 던전은 아무도 K일 이내에 클리어하지 못하는 것이 자명하므로, 해당 보스 던전들은 전부 없다고 생각해도 좋습니다.

- 이때, 깊이가 x인 보스 던전의 수용량이 y라면, $K-x+1$일동안 최대 $y(K-x+1)$명의 플레이어가 칭호를 얻을 수 있습니다.
- 즉, 깊이가 서로 다른 두 보스 던전에 대해, 깊이가 더 얕은 보스 던전에 할당 가능한 인원이 더 많다면 단순히 모든 인원을 해당 보스 던전으로, 아니라면 남는 인원을 더 깊은 보스 던전으로 보내는 식으로 전략을 세우는 것이 최적입니다.
- tree dp를 사용하여, 가장 깊은 곳에 있는 보스 던전에 인원을 최대한 할당하다가 서로 다른 보스 던전으로 가는 경로의 일반 던전에 병목 현상이 생겨 모두 보내지 못하게 되는 경우, 가장 깊은 곳에 있던 보스 던전에 할당한 플레이어부터 제거하는 방식으로 구현할 수 있습니다.

- 이를 단순히 구현하면 시간 $\mathcal{O}\left(N^2\right)$ 이 되므로, 이를 최적화해야 합니다.
- 가장 깊은 곳부터 tree dp를 진행하며, smaller to larger 기법을 활용한 union find로 보스 던전의 깊이, 할당한 인원수 및 해당 노드까지 봤을 때 할당한 총 인원수를 저장해 둡시다.
- Set 등의 자료구조를 사용하면, 총 인원수가 해당 노드의 할당량보다 높을 때 깊은 노드에 할당한 인원부터 제거하면 약 $\mathcal{O}\left(N \log N\right)$ 시간복잡도로 tree dp를 구현할 수 있습니다.
- 최종적으로 0번 노드에 남은 결과를 토대로, (할당한 플레이어 수)$\times$($K-$(깊이)$+1$)을 각각 합하여 정답으로 출력하면 됩니다.

2I. 달구와 윤이의 카드 배틀

`greedy`
출제진 의도 – **Hard**

- 통계:
  - Junior: 제출 7번, 정답 0번, 정답률 0.000%
- First Solve:
  - Junior: -
- 출제자: 김도훈[99asdfg]

2I. 달구와 윤이의 카드 배틀

- 달구가 승리할 수 있는 라운드 수의 최댓값을 찾는 것이 목표이므로, 윤이가 각 라운드에 어떤 카드를 내는지를 미리 알고 있다고 가정하고 문제를 해결해도 좋습니다.
- 이제 윤이가 받은 카드를 짝수 카드와 홀수 카드로 나눈 뒤, 달구가 짝수 대항 카드와 홀수 대항 카드를 각각 배정한다고 해봅시다.
- 짝수 카드를 배정하는 경우, 반드시 큰 수를 홀수 대항 카드로, 작은 수를 짝수 대항 카드로 배정하는 것이 이득입니다.
- 반대로, 홀수 카드를 배정하는 경우, 반드시 큰 수를 짝수 대항 카드로, 작은 수를 홀수 대항 카드로 배정하는 것이 이득입니다.

- 즉, 만약 달구가 받은 짝수 카드의 장수가 E개일 때, 짝수 카드 중 홀수 대항 카드로 사용할 짝수 카드의 장수 e_1을 선택하면 짝수 대항 카드로 사용할 짝수 카드의 장수 $e_2 = E - e_1$로 자연스럽게 결정됩니다.
- 또한, 홀수 카드의 장수 $O = N - E$로, 배정할 수 있는 남은 홀수/짝수 대항 카드도 이미 결정되었으므로 짝수 카드 중 홀수 대항 카드로 사용할 카드의 장수 하나만 결정하면 상대 카드에 대해 어떤 카드를 배정할지 여부는 자동으로 결정됩니다.

2I. 달구와 윤이의 카드 배틀

- 홀수 카드 대항 집합에 홀수 카드가 x 장, 짝수 카드가 y 장 있을 때 승리하는 라운드 수를 계산해 봅시다.
- 우선 홀수 카드 대항 집합의 모든 카드가 홀수라고 가정하면, 단순히 오름차순 수열 a 가 주어졌을 때 $b_i < a_i$ 인 개수를 최대화하도록 b_i 를 배정하는 문제로 환원할 수 있습니다.
- 이는 b 수열에서 가장 큰 x 개의수를 맨 앞에 순서대로 배정하고, 나머지 카드를 b_{x+1} 부터 차례대로 배정했을때 $k > x$ 인 k 에 대해 $b_k > a_k$ 를 만족하는 x 를 찾는 것과 동일하고, 이는 $\mathcal{O}(N)$ 혹은 $\mathcal{O}(N \log N)$ 으로 해결할 수 있습니다.

- ✓ 이제 홀수 대항 집합에서 홀수를 하나 빼서 짝수를 하나씩 집어넣는다고 하면, 가장 큰 홀수를 빼서 가장 큰 짝수를 넣는 것이 최적입니다.
- ✓ 이때, 넣은 짝수는 항상 해당 짝수보다 작은 가장 큰 홀수 자리에 넣어주는 것이 최적이며, 이는 이분 탐색 등으로 원소 하나 당 $\mathcal{O}(\log N)$ 안에 위치를 찾아줄 수 있습니다.
- ✓ 위 과정을 짝수 대항 집합에도 동일하게 진행해주면, 시간복잡도 $\mathcal{O}(N \log N)$ 으로 각 대항 집합에 홀수 카드가 x 장, 짝수 카드가 y 장 있을 때 승리하는 라운드 수를 미리 계산해둘 수 있으며, 그 이후 각각의 e_1 에 대해 $\mathcal{O}(1)$ 로 계산하면 총 시간복잡도 $\mathcal{O}(N \log N)$ 으로 문제를 해결할 수 있습니다.
- ✓ 추가 관찰을 통해 투 포인터, 플로우 등 다른 풀이로도 해결할 수 있습니다.

11. 포닉스의 트리 세기

`combinatorics`
출제진 의도 – **Hard**

- 통계:
  - Senior: 제출 1번, 정답 0번, 정답률 0.000%
- First Solve:
  - Senior: -
- 출제자: 곽민성`minsung05`

- ✓ 구성된 트리에 차수가 2인 정점을 전부 축약하여 차수가 1이거나 3인 정점만 남겨 두는 관찰을 통해, $A_1 = A_3 + 2$가 트리가 존재할 필요충분조건이라는 것을 확인할 수 있습니다.
- ✓ 자식 노드가 0개, 1개, 2개인 노드의 수를 각각 n_0, n_1, n_2라고 합시다.
- ✓ 비어 있는 좌/우는 나중에 한번에 2^{n_1}으로 계산해줄 수 있으므로, 노드를 배치할 때 항상 왼쪽에 먼저 배치한다고 가정해도 좋습니다.

- 이제 자식이 $0, 1, 2$개인 노드를 차례대로 배치하는 경우의 수를 하나의 문자열로 나타내어 봅시다.
- preorder 순회를 기준으로 각 노드의 자식의 수를 나열하면, 이는 각 트리에 대한 unique한 문자열로 구성할 수 있습니다.
- 루트를 문자열의 첫 문자로 처리해야 하므로, 문자열의 첫 문자는 반드시 1이나 2여야 합니다.
- 또한, 마지막에는 리프로 끝나므로, 마지막 문자는 0이어야 합니다.
- 마지막으로, 모든 $i < N$에 대해 $[1, i]$ 구간에 있는 0의 개수는 항상 $[1, i]$ 구간에 있는 2의 개수보다 작거나 같아야 합니다. 해당 경우의 수는 카탈란 수로 계산할 수 있습니다.

- 루트가 n_1에 포함되는 경우와 n_2에 포함되는 경우를 따로 나눠서 계산하면, 경우의 수는 다음과 같습니다.

$$\frac{2^{A_2+1} \times (N-2)!}{(A_1-1)!A_2!A_3!} + \frac{2^{A_2} \times (N-2)!}{A_1!(A_2-1)!A_3!}$$

- 값이 크므로 오버플로우에 유의합시다.