슬기로운 마이폴 생활 풀이

Official Solutions

by

슬기로운 마이폴 생활 출제진

개인 후원 : toycartoon mj1000j lunarlity moner_ari cubic nflight11 nabina1395

요약

- 2시간 30분의 대회동안 총 **206**명이 참가했습니다.
- **1493**개의 제출이 있었고, **853**개의 코드가 AC를 받았습니다.
- All Solve가 나오기까지 **1시간 49분**이 소요되었습니다.

문제		의도한 난이도	출제자
A	덕후	**Beginner**	corkd
B	사각지대	**Beginner**	corkd
C	꼬마 미타	**Easy**	oh040411
D	PAUL 문자열 (Easy)	**Easy**	lunarlity
E	실크의 노래	**Easy**	oh040411
F	공룡 게임	**Medium**	kdy40929
G	박스 쌓기	**Medium**	amazingtt2
H	PAUL 문자열 (Hard)	**Medium**	lunarlity
I	중앙값의 함정	**Medium**	kdy40929
J	추진기	**Hard**	pyb1031
K	PLATiNA::LAB	**Hard**	kaorin
L1	사과 게임 (Easy)	**Challenging**	terrasphere
L2	사과 게임 (Hard)	**Challenging**	terrasphere, gs18115, qvixnh22

A. 덕후

- `implementation`, `arithmetic`
- 프리즈 전까지 제출 224번, 정답 177명 (정답률 79.018%)
- 처음 푼 사람: **79brue**, 0분
- 출제자: `corkd`

A. 덕후

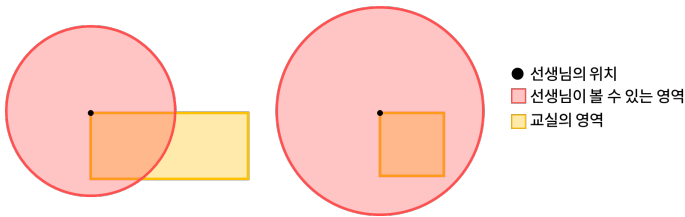
- 주어진 식을 그대로 구현하여 해결할 수 있습니다.
- 계산값이 0보다 작은 경우 0을 출력해야 함에 유의합니다.

B. 사각지대

- `geometry`
- 프리즈 전까지 제출 134번, 정답 117명 (정답률 87.313%)
- 처음 푼 사람: **pkearth**, 1분
- 출제자: `corkd`

B. 사각지대

- 선생님으로부터 거리가 r 이하인 영역은 반지름의 길이가 r 인 원이 되며, 이 중 교실에 대응하는 직사각형 영역과 겹치는 부분은 반지름의 길이가 r 인 사분원이 됩니다.
- 선생님께 들키지 않고 딴짓을 할 수 있는 영역의 넓이는 직사각형 영역의 넓이에서 사분원의 넓이를 뺀 값이 됩니다.
- $r \leq \min(w, h)$ 인 경우만 입력으로 주어지므로, 아래 그림과 같이 겹치는 부분의 영역이 사분원이 아닌 경우는 고려할 필요가 없습니다.



C. 꼬마 미타

- `string`, `arithmetic`
- 프리즈 전까지 제출 141번, 정답 86명 (정답률 60.993%)
- 처음 푼 사람: **79brue**, 3분
- 출제자: oh040411

C. 꼬마 미타

- 시침과 분침이 가리키는 수를 각각 H, M 이라고 합시다.
- 입력의 앞 2글자와 뒤 2글자를 정수로 변환하면 얻을 수 있습니다. 이것을 구현하는 방법은 여러 가지가 있습니다.
- Python
  - `H, M = map(int, input().split(":"))`
  - ```
    S = input()
    H, M = int(S[:2]), int(S[3:])
    ```
- C++
  - `int H, M; scanf("%d:%d", &H, &M);`
  - ```
    string S; cin >> S;
    int H = stoi(S.substr(0, 2)), M = stoi(S.substr(3, 2));
    ```

C. 꼬마 미타

- 분침은 한 시간에 $360\,^\circ$, 1 분에 $6\,^\circ$ 를 회전합니다.
- 따라서 정각을 기준으로 분침이 회전한 각도는 $\theta = (360\,^\circ)H + (6\,^\circ)M$ 입니다.
- 자명종 시계와 벽시계의 각도를 각각 $\theta_1 = (360\,^\circ)H_1 + (6\,^\circ)M_1$, $\theta_2 = (360\,^\circ)H_2 + (6\,^\circ)M_2$ 라고 하면, 한 방향으로 $\Delta\theta = |\theta_2 - \theta_1|$ 만큼 돌리면 시계를 맞출 수 있습니다.
- 한편, 분침을 $360\,^\circ \times 12 = 4320\,^\circ$ 만큼 돌리면 제자리로 돌아오게 됩니다. 따라서 반대 방향으로 분침을 $4320\,^\circ - \Delta\theta$ 만큼 돌려도 시계를 맞출 수 있습니다.
- 둘 중 더 작은 값인 $\min(\Delta\theta, 4320\,^\circ - \Delta\theta)$ 가 답이 됩니다.
- 별해로, 분침을 시계 방향과 시계 반대 방향으로 1 분씩 돌리는 시뮬레이션을 통해서도 해결할 수 있습니다.

D. PAUL 문자열 (Easy)

- ✓ `implementation`, `string`, `parity`
- ✓ 프리즈 전까지 제출 158번, 정답 80명 (정답률 50.633%)
- ✓ 처음 푼 사람: **makon**, 5분
- ✓ 출제자: `lunarlity`

D. PAUL 문자열 (Easy)

- 문자열에 가능한 연산이 이웃한 두 문자를 제거하는 것임에 주목합시다.
- 먼저, P, A, U, L을 제외한 문자를 무시하고 봤을 때, $PAUL$ 순서로 있는 것이 아니라면 항상 불가능합니다.
- P와 A, A와 U, U와 L 사이에 있는 문자의 수가 2의 배수가 아닐 경우, 이웃한 두 문자를 제거하는 것을 반복하여 사이에 있는 모든 문자를 없애는 것은 불가능합니다.
- 추가로, P 왼쪽에 있는 문자도 완전히 제거되어야 하므로, 거기에 있는 문자도 짝수 개여야 합니다.
- 마찬가지로 L 오른쪽에 있는 문자도 완전히 제거되어야 하므로, 거기에 있는 문자도 짝수개여야 합니다.

E. 실크의 노래

- `arithmetic, case work`
- 프리즈 전까지 제출 110번, 정답 56명 (정답률 50.909%)
- 처음 푼 사람: **makon**, 2분
- 출제자: `oh040411`

E. 실크의 노래

- 먼저 회복을 할 수 없는 경우, 즉 $S = 0$인 경우를 생각해 봅시다.
- 현재 체력이 X 라면, $\lceil X/2 \rceil$ 번의 공격을 맞을 수 있습니다. $\lceil x \rceil$ 은 x를 정수로 올림한 값을 의미합니다.
- 회복을 하면 항상 체력에 양수가 더해지고, 공격을 맞으면 항상 체력에 음수가 더해집니다.
- 덧셈은 교환 법칙이 성립하므로, (최대 체력을 무시하고) 처음에 모든 회복을 하고 그 다음부터 공격을 받는 것으로 생각해도 최종 결과는 같습니다.
  - 예를 들어, 예제 2번의 경우 $8 \to 11 \to 14 \to 12 \to 10 \to \cdots \to 0$으로 생각해도 공격을 받은 횟수는 7번으로 같습니다.
- 즉, 전투에서 회복한 총량을 Y 라고 하면, 체력이 $X = H + Y$에서 시작하여 회복을 하지 않고 맞기만 하는 것과 같은 결과가 됩니다.

- $H = 1$ 또는 $H = 2$인 경우, 한 번의 공격을 맞으면 죽기 때문에 회복할 기회가 없습니다. 따라서 $X = H$입니다.
- $H = 3$ 또는 $H = 4$인 경우, 한 번 맞을 때마다 죽지 않기 위해 회복을 해야 합니다. 이때 회복량은 항상 2입니다. 따라서 $X = H + 2S$입니다.
- $H \geq 5$인 경우, 체력이 1 또는 2 남았을 때마다 한 번씩 회복하는 전략을 사용하면 항상 3의 체력을 회복할 수 있습니다. 따라서 $X = H + 3S$입니다.
- 각 경우에 대해 정답은 항상 $\lceil X/2 \rceil$입니다.

F. 공룡 게임

- dp, bfs
- 프리즈 전까지 제출 70번, 정답 53명 (정답률 75.714%)
- 처음 푼 사람: **frtfrt1**, 6분
- 출제자: kdy40929

F. 공룡 게임

- i번 칸을 지날 수 있다면 $A[i] = 1$, 아니라면 $A[i] = 0$이 되는 배열 A를 떠올립시다.
- 1번 칸에서 출발하므로 $A[1] = 1$입니다.
- i번 칸에 장애물이 있다면 $A[i] = 0$입니다.
- i번 칸에 장애물이 없다면 $i - 1$번 칸과 $i - K$번 칸 중 적어도 하나를 통과해야 합니다.
- 또한, $i - 1$번 칸과 $i - K$번 칸 중 하나라도 지날 수 있다면 i번 칸도 지날 수 있습니다.
- 따라서 $A[i - 1] = 0$, $A[i - K] = 0$일 때만 $A[i] = 0$이고 다른 경우 $A[i] = 1$입니다.
- 이를 메모이제이션 dp를 사용하여 i를 2부터 N까지 반복하면 $O(N)$에 $A[N]$을 구할 수 있습니다.

F. 공룡 게임

- BFS를 활용하는 풀이도 가능합니다.
- N 개의 칸을 정점으로 하는 그래프를 구성합니다.
- i 번 칸과 $i+1$ 번 칸 모두 장애물이 없다면 i 번 칸에서 $i+1$ 번 칸으로 향하는 유향 간선을 형성합니다.
- 마찬가지로 i 번 칸과 $i+K$ 번 칸 모두 장애물이 없다면 i 번 칸에서 $i+K$ 번 칸으로 향하는 유향 간선을 형성합니다.
- N 번째 칸에 도착할 수 있는 것은 이 그래프에서 간선을 따라 이동해 1 번 정점에서 N 번 정점으로 갈 수 있는지와 동치입니다.
- 따라서 1 번 정점을 루트로 하고 BFS 탐색을 진행하여 N 번 정점에 도달하는 경우가 있다면 YES를, 아니면 NO를 출력하면 됩니다.
- 이때 그래프의 간선은 $2N$ 개를 넘지 않으므로 $O(N)$ 에 BFS 탐색이 완료됩니다.

G. 박스 쌓기

- ✓ `greedy`
- ✓ 프리즈 전까지 제출 62번, 정답 52명 (정답률 83.871%)
- ✓ 처음 푼 사람: **mythofys**, 2분
- ✓ 출제자: `amazingtt2`

G. 박스 쌓기

- N 개의 박스들을 무게 비내림차순으로 정렬합니다. 현재까지 쌓여진 박스 무게의 총합을 S 라고 합시다. S 의 초기 값은 0입니다.
- 무게가 작은 박스부터 탐색하면서 박스의 무게가 S 보다 크거나 같다면, 해당 박스를 쌓을 수 있습니다. 해당 박스의 무게를 S 에 추가합니다.
- 즉, 현재 탐색하지 않은 박스들 중 S 이상의 가장 작은 무게의 박스를 포함해줍니다. 이렇게 만든 해는 최적해의 집합에 포함되며, 답을 도출할 수 있습니다.

- 최대로 쌓을 수 있는 박스 개수가 t 이고 무게 조건을 만족하는 최적해 집합의 수열 $a[1], a[2], \cdots, a[t]$ 가 있다고 합시다.
- 또한 작은 무게의 박스부터 탐색하며, 현재 쌓여진 박스들의 무게 총합보다 작을 때마다 고른 길이 t 의 수열 $g[1], g[2], \cdots, g[t]$ 가 있다고 합시다.
- $g[1]$ 은 정의상 가장 작은 원소이기 때문에 $a[1] \geq g[1]$ 입니다. 즉, $a[1]$ 을 $g[1]$ 로 바꿔도 무게 조건을 만족합니다.
- 임의의 자연수 j 가 $j < t$ 를 만족할 때, $g[1], g[2], \cdots, g[j], a[j+1], \cdots, a[t]$ 가 있다고 가정합시다.

- 그러면 $g[1], \cdots, g[j+1], a[j+2], \cdots, a[t]$ 도 무게 조건을 만족합니다.
- 따라서 $a[j+1] \geq g[1] + g[2] + \cdots + g[j]$ 입니다.
- $g[j+1]$ 은 $g[1] + g[2] + \cdots + g[j]$ 이상인 가장 작은 원소이고, 따라서 $a[j+1] \geq g[j+1]$ 로 $a[j+1]$ 을 $g[j+1]$ 로 치환해도 무게 조건을 만족합니다.
- 따라서 무게 조건을 만족하는 $g[1], g[2], \cdots, g[t]$ 가 존재합니다.

H. PAUL 문자열 (Hard)

- ✓ `greedy`, `string`, `parity`
- ✓ 프리즈 전까지 제출 96번, 정답 46명 (정답률 47.917%)
- ✓ 처음 푼 사람: **79brue**, 6분
- ✓ 출제자: `lunarlity`

H. PAUL 문자열 (Hard)

- D의 풀이를 바탕으로 추가적으로 작성되었습니다.
- 두 문자씩 제거된다는 특징을 고려하였을 때, P, A, U, L의 위치의 홀짝성이 불변합니다.
- 1-based 기준으로 P, U는 홀수 위치에 있어야 하며, A, L는 짝수 위치에 있어야 합니다.
- 항상 두 개의 문자씩 제거되기 때문입니다. 따라서 나머지 문자들의 위치의 홀짝성이 변하지 않습니다.
- 이를 토대로, 그리디하게 앞부분부터 P, A, U, L을 순차적으로 살리면 됩니다.
- 모든 순회를 돌았을 때 $PAUL$이 남는 것이 가능하다면 YES를, 불가능하다면 NO를 출력해주면 됩니다.
- 전체 시간복잡도는 $O(N)$입니다.

I. 중앙값의 함정

- `greedy`, `ad_hoc`, `constructive`
- 프리즈 전까지 제출 76번, 정답 55명 (정답률 72.368%)
- 처음 푼 사람: **jthis**, 4분
- 출제자: kdy40929

- $N = 1$인 경우는 시행을 하지 않으므로 자명합니다.
- 세 수 a, b, c를 이용해서 시행하는 경우 세 수의 중앙값은 a, b, c 중 하나이므로 매 시행은 새로운 수를 만들어낼 수 없습니다.
- 정확히 세 수가 남았을 때 반드시 세 수를 시행에 모두 사용하므로, 모든 수는 적어도 한 번 시행에 사용됩니다.
- 따라서 N개 수 중 최댓값은 시행 시 중앙값이 될 수 없으므로 마지막에 남길 수 없습니다.

- 한편, N 개 수 중 두 번째로 큰 값을 남길 수 있습니다. 아래에서 이를 보이겠습니다.
- $N > 3$ 일 때 항상 가장 작은 세 수를 이용해서 시행한다면 가장 큰 수 2 개는 그대로 유지한 채 수의 개수를 반복해서 2 씩 줄일 수 있습니다.
- 따라서 칠판에 남은 수가 정확히 3개가 되면 전체 배열에서 최댓값, 두 번째로 큰 값, 그리고 이보다 작은 수 하나가 남게 됩니다.
- 마지막으로 이 세 수를 이용해서 시행을 진행하면 배열에서 두 번째로 큰 값을 남길 수 있습니다.

J. 추진기

- `constructive`
- 프리즈 전까지 제출 75번, 정답 20명 (정답률 26.667%)
- 처음 푼 사람: **jthis**, 12분
- 출제자: pyb1031

- x_i를 추진기를 i번 사용한 후의 위치라 할 때 $x_0 = 0, x_{i+1} = x_i + a_i$인 수열 x에 대해 $\max(|x_i|)$의 값을 최소화 하는 문제입니다.
- $|x_1| \geq l$입니다. 또한 $|a_p| = r$인 p에 대해 $|x_p - x_{p-1}| = r$, 즉 $\max(|x_p|, |x_{p-1}|) \geq \lceil r/2 \rceil$이므로 답은 $\max(l, \lceil r/2 \rceil)$보다 작아질 수 없습니다.
- 따라서 $\max(|x_i|)$의 값이 $\max(l, \lceil r/2 \rceil)$가 되게한다면 그것이 답이 됩니다.
- $a_1 = \max(l, \lceil r/2 \rceil)$으로 한 후 사용하지 않은 큰 수부터 부호를 번갈아가며 배치하면 그런 수열을 구성할 수 있습니다.

K. PLATiNA::LAB

- ad_hoc
- 프리즈 전까지 제출 76번, 정답 33명 (정답률 43.421%)
- 처음 푼 사람: **jthis**, 21분
- 출제자: kaorin

- 문제의 조건에 따라 양끝의 미생물과 AA, BB와 같이 자기 자신과 같은 종류의 미생물이 인접한 미생물에서는 변화가 일어날 수 없습니다.
- 그렇지 않은 미생물들에게서는 적어도 변화가 한번은 일어나게 됩니다. 이러한 미생물들을 인접한 애들끼리 묶어본다면 ABAB... 또는 BABA... 꼴의 문자열들이 나오게 됩니다.
- 이 중에서 하나만 부분 문자열로 가져와서 좀 더 관찰해봅시다. 편의상 이 문자열을 S라고 부르겠습니다.

- S 길이가 k 라고 한다면 다음과 같은 관찰을 할 수 있습니다.
- 처음에 1 시간이 지난다면 S 에 속하는 모든 미생물들이 변화합니다.
- 여기서 1 시간이 더 지난다면 (즉, 2 시간이 지났다면) S 에서 1 번 미생물과 k 번 미생물에는 변화가 없습니다.
- 계속해서 확인해본다면 t 시간이 지났을때는 S 에서 $1, 2, \cdots, t-1$ 번 미생물과 $k-t+1, \cdots, k-1, k$ 번 미생물에 변화가 일어나지 않음을 알 수 있습니다.

- 결과적으로 t 시간이 지났을 때, S에서 i번 미생물과 $k-i$번 미생물은 $\min(i, t)$ 번만 변화하고 그 이후로는 변화하지 않습니다.
- 따라서 1 시간이 지났을 때 변화하는 미생물들을 인접한 애들끼리 묶어 부분문자열을 만들어준 후에 각각의 부분문자열에서 변화를 홀수번만 한 미생물들만 변화시키고 결과를 출력해주면 답이 됩니다.
- 부분 문자열을 만드는 과정은 $O(N)$ 시간이 걸리고, 만들어진 부분 문자열을 통해 모든 미생물의 변화를 판별하는 것도 $O(N)$ 이 걸리므로 총 시간 복잡도 역시 $O(N)$ 입니다.

L1. 사과 게임

- ✓ `game_theory`, `ad_hoc`, `parity`
- ✓ 프리즈 전까지 제출 23번, 정답 7명 (정답률 30.435%)
- ✓ 처음 푼 사람: **fermion5**, 37분
- ✓ 출제자: `terrasphere`

- 남아 있는 사과의 개수를 L 이라고 합시다.
- 한 턴에서 할 수 있는 교환은 모두 “시작 위치를 기준으로 거리가 같은 두 사과를 짝지어 바꾸는 것” 입니다.
- 이 과정에서 L 이 홀수인 경우에만 플레이어가 앞으로 가져갈 사과의 집합을 바꿀 수 있습니다.

- 현재 남아 있는 사과 개수를 L 이라고 하고, 시작 위치부터 반시계 방향으로

  검은색, 흰색, 검은색, 흰색, ...

  순서로 번갈아 색을 칠한다고 합시다.
- 이렇게 색을 칠하면, 현재 턴의 플레이어가 앞으로 가져가게 될 사과들은 항상 한 색의 칸들에만 위치하게 됩니다.
- 이제 시작 위치에서 반시계 방향으로 k 번째 사과와 시계 방향으로 k 번째 사과를 생각합니다.
  - 인덱스를 $0, 1, \ldots, L-1$ 로 두면 각각 k 와 $L-k$ 에 해당합니다.

- ✓ L이 짝수인 경우, k와 $L-k$는 항상 같은 홀짝성을 가지므로 항상 같은 색이 됩니다.
- ✓ L이 홀수인 경우, k와 $L-k$는 항상 서로 다른 홀짝성을 가지므로 항상 다른 색이 됩니다.
- ✓ 따라서
  - L이 짝수이면 같은 색끼리만 교환할 수 있으므로, 현재 플레이어가 가져가게 될 색 칸들의 집합은 바뀌지 않습니다.
  - L이 홀수이면 서로 다른 색끼리만 교환할 수 있으므로, 좋은 사과를 자신의 색으로 끌어올 수 있고, 미래에 가져갈 사과 집합을 실제로 바꿀 수 있습니다.
- ✓ 따라서 N이 홀수인 경우와 짝수인 경우를 나눠서 생각합니다.

- N 이 홀수이면, 처음 $L = N$ 에서 게임을 시작하므로 모든 테라의 턴에서 L 이 홀수입니다.
- 따라서 미래에 어떤 사과들을 가져갈지 집합을 바꿀 수 있는 플레이어는 테라뿐입니다.
- 루루는 어떤 전략을 사용하더라도, 결국 자신이 가져갈 사과들의 집합 자체는 바꾸지 못합니다.

- 먼저 사과 중에 맛이 0인 사과가 하나도 없다고 가정합니다.
- 이 경우 테라는 첫 턴에 반드시 양수 점수의 사과를 가져가게 됩니다.
- 이후 남은 사과의 개수는 짝수가 되고, 루루의 턴에서는 항상 L이 짝수입니다.
- 앞에서 본 관찰에 따라 루루는 자신이 가져갈 사과 집합을 유리하게 바꾸지 못합니다.
- 반면 테라는 첫 번째 턴에 모든 사과 쌍에서 자신의 사과의 맛을 루루의 사과의 맛 이상으로 변경할 수 있습니다.
- 따라서 N이 홀수이고 맛 0이 하나도 없으면 테라가 승리합니다.

- 이제 사과 중에 맛이 0인 사과가 적어도 하나 존재한다고 가정합니다.
- 앞선 관찰에 의해 루루는 반드시 맛이 0인 사과를 시작 위치로 선택해야 합니다.
- 어떤 0인 사과를 시작 위치에 두고 그 사과를 x_0이라고 합시다. 남은 사과는 x_0을 기준으로 반시계 방향과 시계 방향에 각각 $(N-1)/2$개씩 존재합니다.
- 만약 남은 사과의 맛이 x_0를 기준으로 대칭이 아니라면, 첫 턴에 테라는 자신의 사과의 맛의 합을 루루의 사과의 맛의 합보다 크도록 할 수 있습니다.
- 따라서 x_0을 기준으로 맛이 대칭이 아닌 경우 테라가 승리합니다.

- 이제 배열이 x_0을 기준으로 완전히 대칭이라고 가정합니다.
- 이때 루루는 자신의 턴이 올 때마다 가능한 모든 사과 쌍을 교환하면, 다음 차례에 테라가 받을 배열은 대칭인 상태가 유지됩니다.
- 결과적으로 두 사람이 얻는 총 점수는 항상 같아지고, 게임은 동점으로 끝납니다.
- 동점인 경우에는 루루가 승리하므로, 이러한 대칭 구조를 만드는 0이 하나라도 존재하면 루루가 승리합니다.

- 이제 N이 짝수인 경우를 생각합니다. 이때 처음 $L = N$은 짝수입니다.
- 게임이 진행되는 동안
  - 테라의 턴에는 항상 L이 짝수이고,
  - 루루의 턴에는 항상 L이 홀수입니다.
- 따라서 미래에 가져갈 사과 집합을 실제로 바꿀 수 있는 쪽은 항상 루루이고, 테라는 어떤 교환을 하더라도 자신이 가져갈 사과들의 집합을 바꾸지 못합니다.

- 루루가 아무런 교환을 하지 않는 경우, 루루와 테라는 인덱스의 홀짝성이 같은 사과들만 가져가게 됩니다.
- 루루는 게임 시작 전에 시작 위치를 마음대로 고를 수 있기 때문에 1번 사과를 시작 위치로 하는 경우와 2번 사과를 시작 위치로 하는 경우 중 더 유리한 쪽을 선택할 수 있습니다.
- 이 두 경우 중 적어도 한 경우에서는 루루가 테라 이상으로 점수를 가져갈 수 있음이 보장됩니다.
- 동점인 경우에는 루루가 승리하므로, N 이 짝수인 경우에는 입력과 무관하게 항상 루루가 승리합니다.

- 최종적으로 N 이 홀수인 경우에는 어떤 사과가 존재하여 맛이 0이고, 그 사과를 기준으로 사과의 맛이 대칭이라면 루루가 승리하고, 그렇지 않으면 테라가 승리합니다.
- N 이 짝수인 경우에는 사과의 맛과 관계없이 루루가 승리합니다.
- 시간복잡도는 N 이 홀수라면 $O(N^2)$, N 이 짝수라면 $O(1)$ 입니다.
- L1의 경우 Manacher 알고리즘을 사용하면, $O(N)$ 에 풀 수 있습니다.

L2. 사과 게임 (Hard)

- ✓ `game_theory`, `ad_hoc`, `parity`, `dp`
- ✓ 프리즈 전까지 제출 14번, 정답 2명 (정답률 14.286%)
- ✓ 처음 푼 사람: **sadtreap**, 109분
- ✓ 출제자: `terrasphere`, `gs18115`, `qvixnh22`

- L1의 풀이 방식과 동일하게, N 이 홀수인 경우와 짝수인 경우를 나눠서 생각합시다.
- 마찬가지로, 시작 위치를 정했다고 가정하고, 시작 위치부터 반시계 방향으로 검은색과 흰색을 번갈아 칠합니다.
- 시작 위치부터의 V_i 값을 U_i 로 정의합시다. 시작점을 index 0으로 해서 생각합니다.

- N 이 홀수인 경우, 테라가 첫 턴에 하는 행동은 다음 두 가지입니다.
  - L1과 마찬가지로 사과의 인덱스를 매겼을 때, k와 $N-k$를 각 k에 대해 바꿀지 말지 결정합니다.
  - 시작 위치에 놓인 사과를 자신의 점수로 합산합니다.
- N 이 홀수이므로, 테라의 첫 번째 행동의 각 k에 대해, 하나는 검은색, 하나는 흰색입니다.
- 테라가 가져가게 될 사과는 검은색에 해당하므로, 두 사과 중 높은 맛 점수의 사과를 검은색에 배치하는 것으로 다음과 같은 최소 점수 차이를 보장할 수 있습니다.
  - $U_0 + \sum_{i=k}^{(N-1)/2} |U_k - U_{N-k}|$

L2. 사과 게임 (Hard)

- 이제 루루의 관점에서 바꿔서 바라봅시다.
- 테라의 전략 상으로, 루루가 어떻게 하더라도 테라가 아무것도 하지 않는 것만으로 첫 턴에 배치된 검은색 사과와 흰색 사과의 점수 차이를 보장할 수 있습니다.
- 따라서 루루는 만약 가능하다면, 정확히 이 값만큼 점수 차이가 나도록 조절하여야 합니다.
- 실제로 이는 다음 전략을 통해서 가능합니다.
  - 모든 턴에, 가능한 모든 스왑 쌍을 스왑합니다.
- 이렇게 하면 테라의 가능한 교환 쌍이 바뀌지 않으므로, 테라의 첫 턴을 기준으로 (U_k, U_{N-k})의 교환 쌍에서 테라가 최대로 낼 수 있는 점수 차는 $|U_k - U_{N-k}|$입니다.
- 따라서 테라와 루루의 최대 점수 차는 최소 점수 차와 동일한 $U_0 + \sum_{k=1}^{(N-1)/2} |U_k - U_{N-k}|$ 입니다.

- 최소 점수 차와 최대 점수 차가 동일하므로, 각자가 최선을 다할 때 점수 차이는 $U_0 + \sum_{k=1}^{(N-1)/2} |U_k - U_{N-k}|$ 입니다.
- 루루가 정할 수 있는 한 위치에 대해 이를 구하는 것은 $O(N)$ 에 수행할 수 있습니다.
- 위치가 N 개이므로, 총 시간복잡도 $O(N^2)$ 에 모든 위치의 점수를 구할 수 있고, 이 중 최솟값이 답이 됩니다.

- 이제 N 이 짝수인 상황을 고려합시다.
- 첫 턴의 테라의 스왑을 고려하여, 스왑이 끝난 후의 맛 점수 상황을 U'_i 라고 합시다.
- 이후 루루가 받게 되는 $N-1$ 개의 사과는 홀수 개이므로, 앞서 설명했던 것과 동일한 논리를 통해 2번째 턴 이후부터의 루루와 테라의 점수 차이는 다음과 같습니다.
  - $U'_1 + \sum_{k=1}^{(N-2)/2} \left|U'_{k+1} - U'_{N-k}\right|$
- 첫 턴까지 고려하면, 테라와 루루의 점수 차이는 다음과 같습니다.
  - $P_{total} = U'_0 - U'_1 - \sum_{k=1}^{(N-2)/2} \left|U'_{k+1} - U'_{N-k}\right|$

- 테라는 P_{total} 을 최대화하여야 합니다. 이는 DP를 통해 $O(N)$ 에 구할 수 있습니다.
- DP의 핵심 아이디어는, 작은 i 부터 (U_i, U_{N-i}) 쌍의 스왑 여부를 정해 나가며 값을 구해 나갈 수 있다는 것입니다.
- $F(i)$ 라는 함수를 정의합시다. $F(i)$ 는 i 번째까지 결정했을 때, 결정된 점수 차이의 합입니다. 다음과 같이 구할 수 있습니다.
  - $F(i) = U'_0 - U'_1$ if $i = 1$
  - $F(i) = U'_0 - U'_1 - \sum_{k=1}^{i-1} \left| U'_{k+1} - U'_{N-k} \right|$ if $2 \le i \le \frac{N}{2} - 2$
  - $F(i) = U'_0 - U'_1 - \sum_{k=1}^{i} \left| U'_{k+1} - U'_{N-k} \right|$ if $i = \frac{N}{2} - 1$

- 다음과 같은 DP를 1 부터 $\frac{N}{2}-1$ 까지의 i에 대해 정의합시다.
  - $DP[i][0]$: U_i와 U_{N-i}를 바꾸지 않았으며, $1 \le j \le i-1$을 만족하는 모든 j에 대한 (U_j, U_{N-j}) 쌍의 스왑 여부가 결정되었을 때의 최대 $F(i)$
  - $DP[i][1]$: U_i와 U_{N-i}를 바꿨으며, $1 \le j \le i-1$을 만족하는 모든 j에 대한 (U_j, U_{N-j}) 쌍의 스왑 여부가 결정되었을 때의 최대 $F(i)$
- $DP[1][0]$과 $DP[1][1]$은 각각 U_0-U_1, U_0-U_{N-1}로 구할 수 있습니다.
- 2 이상의 i에 대해 $DP[i][0]$과 $DP[i][1]$의 값은 $DP[i-1][0]$과 $DP[i-1][1]$의 상태를 고려하여 $O(1)$에 구할 수 있습니다.
- $DP[\frac{N}{2}-1][0]$과 $DP[\frac{N}{2}-1][1]$ 중 큰 값이 테라와 루루의 최적 점수 차가 됩니다.

- ✓ 각 위치의 최적 점수 차이를 $O(N)$에 DP로 구할 수 있습니다.
- ✓ 모든 N개의 위치를 고려하여, 이 중 최솟값을 $O(N^2)$에 구할 수 있고, 이것이 답이 됩니다.