# 2023 SW-IT Contest 해설

by

충남대학교 알고리즘 동아리 ANA

| 문제 | | 의도한 난이도 | 출제자 |
|---|---|---|---|
| **A** | 스위트콘 가격 구하기 | **Easy** | ygonepiece |
| **B** | 치즈버거 만들기 | **Easy** | ygonepiece |
| **C** | 지폐 세기 | **Easy** | kangwlgns |
| **D** | 알파벳 뒤집기 | **Easy** | ygonepiece |
| **E** | 타슈 | **Easy** | kaorin |
| **F** | 강의실 예약 시스템 | **Easy** | kaorin |
| **G** | 치즈버거 만들기 2 | **Medium** | kaorin |
| **H** | 순열 선물하기 | **Medium** | kaorin |
| **I** | 행사 준비 | **Medium** | kaorin |

| 문제 | | 의도한 난이도 | 출제자 |
|---|---|---|---|
| **J** | 전구 상태 바꾸기 | **Hard** | ygonepiece |
| **K** | 옥수수밭 | **Hard** | kaorin |
| **L** | K의 배수 | **Hard** | kangwlgns |
| **M** | 차원문 | **Challenging** | kaorin |
| **N** | 비밀의 화원 | **Challenging** | kaorin |
| **O** | 잃어버린 순수 | **Challenging** | hhs2003 |
| **P** | 도미노 수열 | **Challenging** | ygonepiece |

# A. 스위트콘 가격 구하기

`math`

출제진 의도 – **Easy**

- 제출 62번, 정답자 25명 (정답률 40.323%)
- 처음 푼 사람: **타올**, 1분
- 출제자: `ygonepiece`

## A. 스위트콘 가격 구하기

- $A \times \frac{11}{10} = B$ 이므로 식을 정리하면 $A = B \times \frac{10}{11}$ 입니다.
- $B$ 는 $11$ 의 배수이므로 $11$ 로 나눌 수 있습니다.

# B. 치즈버거 만들기

`math`

출제진 의도 – **Easy**

- 제출 139번, 정답자 61명 (정답률 43.884%)
- 처음 푼 사람: **미미미누**, 1분
- 출제자: `ygonepiece`

## B. 치즈버거 만들기

- 만약 패티가 치즈보다 많다면 만들 수 있는 버거의 최대 크기는 $2 \times B + 1$ 입니다.
- 그렇지 않을 경우에는 $2 \times A - 1$ 입니다.
- 위와 같이 경우의 수를 나누지 않고 $2 \times \min(A - 1, B) + 1$ 으로 정답을 구할 수도 있습니다.
- 수식으로 계산하지 않고, 패티와 치즈가 남아있을 때까지 치즈버거에 계속 쌓는 과정을 시뮬레이션해서 구할 수도 있습니다.

# C. 지폐 세기

`implementation`
출제진 의도 – **Easy**

- 제출 43번, 정답자 23명 (정답률 53.488%)
- 처음 푼 사람: **1938929**, 7분
- 출제자: `kangwlgns`

## C. 지폐 세기

- 모든 지폐의 가로 길이가 다르기 때문에, 가로 길이를 통해서 지폐의 종류를 구분할 수 있습니다.
- $N$ 개의 지폐를 가로 길이로 구분해서 지폐의 금액을 알아낸 뒤 합을 구하면 됩니다.

# D. 알파벳 뒤집기

`implementation`
출제진 의도 – **Easy**

- 제출 31번, 정답자 16명 (정답률 51.613%)
- 처음 푼 사람: **doju00**, 8분
- 출제자: ygonepiece

## D. 알파벳 뒤집기

- 격자의 각 칸을 확인하면서 각각의 알파벳을 주어진 방향에 맞게 뒤집어줍니다.
- 격자 자체를 뒤집는 것이 아닌, 각 칸의 알파벳을 뒤집는 것에 유의합니다.

# E. 타슈

ad_hoc
출제진 의도 – **Easy**

- 제출 80번, 정답자 46명 (정답률 57.500%)
- 처음 푼 사람: **미미미누**, 3분
- 출제자: `kaorin`

## E. 타슈

- $c_i = a_i - b_i$ 로 정의합니다. 유실된 자전거는 없기 때문에, $c_1 + c_2 + \cdots + c_N = 0$ 입니다.
- $c_i < 0$ 인 대여소의 자전거 하나를 $c_j > 0$ 인 자전거 대여소로 옮기는 과정을 반복하면 $c_1, c_2, \cdots, c_N$ 이 모두 $0$ 이 되게 만들 수 있습니다.
- $c_i < 0$ 인 $c_i$ 의 합은 $c_i > 0$ 인 $c_i$ 의 합과 같습니다. 따라서 위 과정은 $c_i < 0$ 번 반복하면 되고, 이 횟수가 $b_1, b_2, \cdots, b_N$ 을 $a_1, a_2, \cdots, a_N$ 으로 만드는데 필요한 자전거를 옮기는 횟수의 하한입니다.
- 요약하면, 정답은 $\frac{\sum_{i=1}^{N} |a_i - b_i|}{2}$ 입니다.

# F. 강의실 예약 시스템

`implementation`
출제진 의도 – **Easy**

- 제출 184번, 정답자 16명 (정답률 17.934%)
- 처음 푼 사람: **김승현**, 6분
- 출제자: `kaorin`

## F. 강의실 예약 시스템

- $r[i]$ 를 $i$ 번째 강의실에 대해서 수락한 마지막 예약이 끝나는 시각으로 정의합니다.
- $j$ 번째 예약에 관한 정보가 $k_j, s_j, e_j$ 라고 할 때, $r[k_j] \leq s_j$ 라면 예약이 가능하고, 그렇지 않다면 예약이 불가능합니다.
- 예약이 가능하다면 $r[k_j] \leftarrow e_j$ 로 업데이트 해줍니다.
- 입출력을 최대 $200\,000$ 줄 해야하므로 버퍼 입출력 등을 사용해야합니다.

# G. 치즈버거 만들기 2

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

- 제출 105번, 정답자 21명 (정답률 20.000%)
- 처음 푼 사람: **bongbong**, 9분
- 출제자: `kaorin`

## G. 치즈버거 만들기 2

패티 $A$개와 치즈 $B$개를 모두 사용하는 것이 가능한지 먼저 판단해야합니다.

- ✓ $A$는 $2 \times B$ 이하여야 합니다.
- ✓ $B$는 $A - 1$ 이상이여야 합니다.

따라서 $B + 1 \leq A \leq 2B$를 만족하는 경우에만 YES, 그렇지 않은 경우에는 NO가 됩니다.

햄버거를 구성하는 방법은 다양하지만 출제자의 풀이는 다음과 같습니다.

- ✓ $A = B + 1$을 만족할 때까지 aba를 출력합니다.
- ✓ $A = B + 1$이 되었다면 크기가 $A + B + 1$인 abab$\cdots$baba를 출력합니다.

# H. 순열 선물하기

`math`, `number_theory`, `ad_hoc`, `constructive`, `backtracking`
출제진 의도 – **Medium**

- 제출 28번, 정답자 9명 (정답률 32.143%)
- 처음 푼 사람: **박현민**, 20분
- 출제자: `kaorin`

## H. 순열 선물하기

- $N = 1$일 경우 $1$은 소수가 아니기 때문에 $[1]$ 순서로 선물합니다.
- $N = 2$일 경우 $[1, 2], [2, 1]$ 어느 방법으로도 선물하는 것이 불가능합니다.
- $N \geq 3$일 경우 $\frac{N \times (N+1)}{2}$은 항상 합성수임을 이용합니다. $[1, 3, 2]$ 뒤에 $4, 5, \cdots, N$을 이어붙이는 방법으로 선물합니다.

# I. 행사 준비

`greedy`
출제진 의도 – **Medium**

- 제출 127번, 정답자 16명 (정답률 12.598%)
- 처음 푼 사람: **22학번최약체**, 31분
- 출제자: `kaorin`

## I. 행사 준비

- 물건의 가격 $(p_1, q_1), (p_2, q_2), \cdots, (p_N, q_N)$ 을 $p_i - q_i$ 를 기준으로 오름차순으로 정렬합니다. 앞에서부터 동하가 $A$ 개의 물건을, 지원이가 $B$ 개의 물건을 구입하는 것이 최적해입니다.
- 이 방법이 최적해임을 증명하기 위해서, 이 방법을 따르지 않는 임의의 더 나은 최적해가 있다고 가정하겠습니다.
- 이 방법을 따르지 않는 임의의 더 나은 최적해에는 $p_i - q_i < p_j - q_j$ 를 만족하는 $i, j$ 번째 물건 중에서 $i$ 번째 물건은 지원이가, $j$ 번째 물건은 동하가 구입한 경우가 적어도 하나 존재합니다. 각각의 물건을 구입하는 비용을 합치면 $q_i + p_j$ 입니다.
- 그런데, $p_i - q_i < p_j - q_j$ 의 항을 적절히 이항시키면 $p_i + q_j < q_i + p_j$ 로 만들 수 있습니다. 즉, 반대로 $i$ 번째 물건을 동하가, $j$ 번째 물건을 지원이가 구입하면 더 나은 해를 만들 수 있다는 것이고, 이는 우리가 처음에 가정했던 임의의 더 나은 최적해라는 가정에 모순됩니다. 따라서 이 방법이 최적해입니다.

# J. 전구 상태 바꾸기

greedy
출제진 의도 – **Hard**

- 제출 27번, 정답자 4명 (정답률 14.814%)
- 처음 푼 사람: **배민수**, 94분
- 출제자: ygonepiece

- 연속한 세 전구의 상태를 바꾸는 연산은 연산의 순서가 중요하지 않습니다. 어떠한 순서로 연산을 적용하건 최종적인 전구의 상태는 똑같기 때문입니다.
- 연속한 세 전구의 상태는 세 번 이상 바꿀 필요가 없습니다. 세 번 바꾸면 원래 상태로 돌아오기 때문입니다.
- 그렇다면 $1, 2, 3$ 번째 전구의 상태를 몇 번 바꿀지 고민하는 것을 시작으로 $2, 3, 4$ 번째 전구, $3, 4, 5$ 번째 전구$\cdots$ 를 계속해서 고려해줄 수 있습니다. 그러나, 각 경우마다 가능한 선택지는 $3$ 가지이므로 단순하게 생각하면 $\mathcal{O}\left(3^N\right)$ 가지 경우를 모두 시도해봐야 합니다.

## J. 전구 상태 바꾸기

- 전구의 색 모두 빨간색으로 만들어야 한다고 가정하고 이전 과정을 진행해봅시다. $1$ 번째 전구를 빨간색으로 만들려면 $1, 2, 3$ 번째 전구의 상태를 바꿔야 하는지는 바로 알 수 있습니다.
- 그렇게 $1$ 번째 전구를 빨간색으로 만들고 나면, $2$ 번째 전구도 빨간색으로 만들려면 $2, 3, 4$ 번째 전구의 상태를 몇 번 바꿔야 하는지도 바로 알 수 있습니다.
- 그렇게 모든 전구를 빨간색으로 바꾸려면 몇 번 상태를 바꿔야 하는지 알 수 있습니다. 물론 모든 전구가 빨간색으로 빛나게 할 수 없을 수도 있습니다.
- 이 과정을 빨간색, 초록색, 파란색에 대해서 모두 시도한 후에 그 중 최솟값을 구하면 문제의 정답입니다.

# K. 옥수수밭

`graphs`, `graph_traversal`, `priority_queue`
출제진 의도 – **Hard**

- 제출 50번, 정답자 7명 (정답률 14.000%)
- 처음 푼 사람: **박현민**, 37분
- 출제자: `kaorin`

## K. 옥수수밭

- 현재 수확할 수 있는 옥수수의 가치를 유지하는 우선순위 큐를 정의합니다. 우선순위는 옥수수의 가치로 정의합니다.
- 격자의 $i$ 행 $j$ 열 $(i, j)$ 에 위치한 옥수수를 수확하면, 현재 수확할 수 있는 옥수수는 상하좌우로 인접한 격자 $(i \pm 1, j \pm 1)$ 에 위치한 옥수수가 추가됩니다. 이미 수확한 옥수수는 제외합니다.
- 힙을 이용하여 우선순위 큐를 구현하면 $\mathcal{O}\left(K \times \log K\right)$ 시간에 해결할 수 있습니다.

# L. K의 배수

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

- 제출 11번, 정답자 2명 (정답률 18.182%)
- 처음 푼 사람: **박현민**, 58분
- 출제자: `kangwlgns`

- $j-1$ 자리 수인 어떤 수 $p$ 의 맨 뒤에 숫자 하나를 더 붙여 $j$ 자리 수를 만드는 과정을 $p \times 10 + q (0 \leq q \leq 9)$ 로 표현할 수 있습니다.
- 예를 들어 $13$ 을 세 자리로 확장하면 앞의 두 자리가 $13$ 인 세 자릿수가 나오게 됩니다. 그런 수는 $130, 131, \cdots, 139$ 가 있고, 이는 $13 \times 10 + q$ 로 표현할 수 있습니다.
- 자릿수를 확장하는 과정에서 곱셈과 덧셈만 사용됐으므로 $\bmod$ 연산의 성질에 의해서 다음이 성립합니다. $(p \times 10 + q) \mod K = (((p \mod K) \times 10) \mod K + q \mod K) \mod K$.

- $j$ 자릿수 중 $K$로 나눈 나머지가 $i$인 수의 개수를 점화식 $R(i, j)$라고 할 때, 위처럼 모듈러 연산을 이용하면 $R(i, j-1)$로부터 $R(((i \mod K) \times 10 \mod K + a_n \mod K) \mod K, j)$를 도출할 수 있습니다.
- $10 \mod K$는 간단하게 구할 수 있고, $q \mod K$는 사용 가능한 숫자 $a_n (1 \le n \le N; 0 \le a_n \le 9)$들을 $K$로 나눈 나머지를 의미하므로 간단하게 구할 수 있습니다.
- $p \mod K$를 매번 구한다면 오래 걸리므로 다이나믹 프로그래밍을 이용하여 시간복잡도를 줄일 수 있습니다.

## L. K의 배수

- 점화식을 이용해 DP 테이블을 채워 나가면 $R(0, M)$ 를 얻을 수 있고, $K$ 의 배수는 $K$ 로 나눈 나머지가 $0$ 이라는 것과 같으므로 $R(0, M)$ 를 출력하면 정답입니다.
- $M$ 자릿수이고 $K$ 로 나눈 나머지는 $K$ 개 존재하므로 DP 테이블을 생성하기 위해 $O(NM)$ 의 공간이 필요하고, 매 테이블을 채우는 과정에서 $N$ 개의 숫자에 대해 모듈러 연산을 하였으므로 시간복잡도는 $O(NMK)$ 입니다.
- 첫 자리가 $0$ 이 아니어야 했으므로 $j = 0$ 일 때 $0$ 을 세지 않음에 주의합니다.

# M. 차원문

`permutation_cycle_decomposition`
출제진 의도 – **Challenging**

- 제출 6번, 정답자 0명 (정답률 0.000%)
- 처음 푼 사람: **없음**
- 출제자: `kaorin`

## M. 차원문

- $i$에서 차원문을 사용해 $a_i$로 이동하는 과정을 $f(i)$라고 합시다. 이걸 $k$번 반복해서 얻은 값을 $f^k(i)$라고 놓는다면, $f^k(i) = i$가 되는 어떤 값 $k$가 존재합니다. 이 과정에서 나온 값의 집합 $f(i), f^2(i), f^3(i), \cdots, f^k(i)$은 사이클을 이루게 되므로 이 문제가 순열 사이클 분할 문제임을 알 수 있습니다.
- 결국 이 문제는 순열의 원소들을 적절하게 뒤바꾸면서 $f^k(i) = i$를 만족시키는 최소의 양의 정수 $k$가 $N$이 되도록 만드는 문제입니다.

이 문제를 해결하기 위해서 아래와 같은 관찰이 필요합니다.

- 서로 다른 두 사이클의 원소를 교환하면 두 개의 사이클이 하나로 합쳐지면서 사이클의 개수가 $1$ 줄어듭니다.
- 어떤 사이클의 크기가 $N$ 이 아니라면, 사이클 안에 있는 어떤 원소에 대해서 크기 차이가 $1$ 인 원소가 다른 사이클에 반드시 존재합니다.

- 초기 순열 사이클의 개수가 $C$ 라고 할 때, 위 관찰을 이용해서 답을 구해보면 사이클들을 하나로 합치는 과정은 $C-1$ 번 필요하며, 이러한 과정을 거칠때 필요한 마나의 비용은 $1^2=1$ 이 되어 $C-1$ 이 됩니다.
- 위 내용을 이해했다면 해를 구성하는 것 역시 가능합니다. 순열 사이클 분할을 한 후에 아무 원소나 하나 골라서 차이가 $1$ 인 원소가 다른 사이클에 존재한다면 그 두개의 사이클을 하나로 합쳐주는 과정을 반복하면 됩니다. 이 과정의 시간복잡도는 $O(N)$ 입니다.

# N. 비밀의 화원

`parametric_search`, `sweeping`
출제진 의도 – **Challenging**

- 제출 12번, 정답자 0명 (정답률 0.000%)
- 처음 푼 사람: **없음**
- 출제자: `kaorin`

## N. 비밀의 화원

- 장애물이 없고 $N$ 과 $M$ 이 커진 토마토 문제입니다. 토마토와 같은 풀이를 이용하면 메모리 초과가 발생합니다.
- 풀이를 바꿔봅시다. 화원의 모든 칸에 꽃이 피워진다는 것은 각각의 행들 모두 모든 칸에 꽃이 피워져 있다는 것과 같습니다.
- 따라서 $N$ 과 $K$ 가 상대적으로 작다는 것을 이용해 시간을 변수로 잡고 처음에 심었던 꽃이 각 행마다 얼마나 피웠는지를 저장한 후에, 각 행마다 꽃이 피워진 구간들을 전부 합쳐봤을 때 모든 칸이 채워져있는지를 확인하면 됩니다.

- 특정 시간 $t$에서 화원의 모든 칸에 꽃을 피웠다면 $1$ 이상의 값 $k$에 대해서 $t+k$ 시간에도 화원의 모든 칸에는 꽃이 피워져 있을 것이고, $t$ 시간 미만이라면 화원의 모든 칸에 꽃이 피워져 있지 않을 것입니다.
- 즉 매개 변수 탐색을 이용하여 $t$를 빠르게 탐색할 수 있습니다.

- 구간을 합치는 과정은 스위핑으로 해결이 가능합니다.
- 각 행에 대해서 $K$ 개의 꽃들이 피워진 구간들을 저장한 후에 정렬 과정을 거쳐서 붙이면 $O(K \log K)$ 시간에 그 행의 모든 칸에 꽃이 피워져 있는지 판별이 가능합니다.
- $N$ 개의 행에 걸쳐서 $K$ 개의 꽃들이 피워진 구간을 정렬하므로 시간복잡도는 $O(NK \log K)$ 입니다.
- 최종 시간복잡도는 $O(NK \log K \log(N+M))$ 이 됩니다.
- 하지만 시간초과가 날 확률이 높습니다.

매개 변수 탐색 이전에 x축을 기준으로 정렬해두는 것도 가능합니다. 구간의 가운데를 $X$, 구간의 왼쪽 끝을 $L$, 구간의 오른쪽 끝을 $R$이라고 놓는다면 정렬했을 경우 다음 조건들이 성립합니다.

- ✓ $X_i < X_j$ 면서 $L_i > L_j$ 라면 구간 $i$는 구간 $j$에 포함되어 있다.
- ✓ $X_i < X_j$ 면서 $R_i > R_j$ 라면 구간 $j$는 구간 $i$에 포함되어 있다.

- 이를 이용해서 다른 원소에 포함되는 경우들을 제외하고 남은 구간들은 $L$ 값이 작은 순으로 정렬되어 있게 됩니다.
- 이제 이 구간들을 전부 합쳤을 때, 모든 칸이 채워지는지 확인해주면 됩니다.
- 이때 시간복잡도는 $O(NK\log(N+M)+K\log K)$ 가 되며, 위의 방식보다 약 10배 정도 빠르게 해결할 수 있습니다.
- 여담으로, $O(NK)$ 에 문제를 해결하는 방법도 있습니다.

# O. 잃어버린 순수

`trees, constructive`
출제진 의도 – **Challenging**

- 제출 1번, 정답자 0명 (정답률 0.000%)
- 처음 푼 사람: **없음**
- 출제자: hhs2003

- 차수가 1인 정점에 간선을 추가하지 않는다면, 그 정점은 자기 자신에게 돌아올 수 있는 경로가 존재 하지 않습니다.
- 따라서, 차수가 1인 정점들은 간선을 추가하여, 사이클에 속하게 해주어야 합니다.
- 차수가 1인 정점들의 개수를 $K$ 라고 합시다.
- 그럼 적어도 간선을 $\lceil \frac{K}{2} \rceil$ 개 추가하여야 합니다. $\lceil \frac{K}{2} \rceil$ 개의 간선을 추가하여, 모든 정점이 사이클에 속하게 할 수 있을 까요?

## O. 잃어버린 순수

- 결론부터 말하자면, 가능합니다.
- 문제를 차수가 1인 정점 쌍의 경로로, 모든 트리의 정점을 덮는 문제로 치환하여 봅시다.
- 차수가 2이상 인 점을 루트로 하여, 트리를 만들어 봅시다.
- 리프노드를 거슬러 올라가게 한다음, 특정 노드에서 리프노드를 서로 매칭하게 하면, 거슬러 올라온 모든 점이 경로에 포함 되게 됩니다.
- 이런식으로, 트리 전체를 리프노드가 거슬러 올라가게 한다음 모두 매칭시키면, 모든 정점이 사이클을 갖는 경로들을 찾을 수 있을 것 입니다.
- 깊이가 깊은 리프노드에서부터 부모로 거슬러 올라가게 해봅시다.

## O. 잃어버린 순수

- 차수가 3이상인 점에서는 리프노드 끼리 만나게 됩니다. 이때, 리프노드 여러개를 거슬러 올라가게 할 필요가 없습니다.
- 적어도 1개만 거슬러 올라가게 해도, 루트까지 이어진 경로를 다 포함 시킬 수 있습니다.
- 그러면, 특정노드에서 거슬러 올라가게 할 1, 2개의 노드를 제외하고 전부 매칭합니다.
- 특정 노드의 자식으로 부터 1, 2개의 노드가 거슬러 올라 올 때, 서로 다른 자식으로 부터 올라온 리프노드 끼리 매칭해야 합니다.
- 그렇지 않고, 같은 자식에서 올라온 2개의 리프노드를 매칭하면, 리프노드에서 특정 노드까지의 경로가 포함 되지 않을 수 있습니다.
- 한 자식으로 부터, 1개 혹은 2개의 리프노드가 올라온다면, 어떤 식으로 매칭 해야 할 까요?

## O. 잃어버린 순수

- 이는 간단한 그리디 문제로 해결 할 수 있습니다.
- 특정 $A$ 번의 노드에서 자식의 개수를 $K$ 라고 할 때, $A_i$ 의 값을 자식에서 올라온 리프노드의 갯수라고 합시다.
- 그럼 $A_i$ 의 값은 1아니면 2입니다.
- $A_i$ 가 2일 때, 한 자식에서 올라온 두 정점을 $A$ 번 노드에서 매칭시키면 거슬러 온 경로가 포함되지 않는 다는 것을 알 수 있습니다.
- 그럼 노드 $A$ 에 대하여, 순열 $A_1, A_2, A_3, \cdots, A_K$ 에 다른 원소 $A_i, A_j (i < j)$ 를 골라 두 원소에 -1연산을 하여, 전체 순열에 있는 원소를 1하나만 남기거나 1 두개, 혹은 2하나 남기고 모두 0으로 만들 수 있는지에 대한 문제로 치환 할 수 있습니다.

## O. 잃어버린 순수

- 값이 2인 원소들 끼리 미리 매칭시켜 모두 1로 만들고 1인 원소들을 모아 아무거나 매칭시키면 됩니다.
- 이를 dfs를 통해 구현할 수 있으며, 각 노드에서 올라온 정점들을 계산하고 최적의 정점들만 올려보낸 다는 점에서 트리 dp와 유사하게 풀 수 있습니다.

# P. 도미노 수열

`segment_tree, lazy_propagation`
출제진 의도 – **Challenging**

- 제출 60번, 정답자 0명 (정답률 0.000%)
- 처음 푼 사람: **없음**,
- 출제자: ygonepiece

- 수열의 첫 번째 수부터 넣으면서 도미노 수열을 만들어 봅시다.
- 수를 넣을 때, 새로운 도미노 수열을 만들 수도 있고 기존의 것에 수를 이어 붙일 수도 있기 때문에 경우의 수가 많이 생기게 됩니다.
- 도미노 수열의 합과 길이를 중심으로 생각해 봅시다.
- 만들어진 도미노 수열에 특정 수를 뒤에 붙일 수 있으면 무조건 붙이는 게 이득이라는 것을 알 수 있습니다.

- 이를 토대로 Naive한 DP 식을 떠올려 보겠습니다.
- $a_i$ 이상의 합을 이루는 도미노 수열 뒤에 $a_i$ 를 붙이고, 도미노 수열의 합이 $w$ 을 이루는 수열 중 최대 길이를 $dp[i][w]$ 라고 합시다.
- 초기 점화식은 $dp[i][a_i] = 1$ 이고, 점화식은
  $dp[i][w] = dp[i-1][w-a_i] + 1 \ (a_i \leq w - a_i) \ (1 \leq dp[i-1][w-a_i])$,
  $dp[i][w] = dp[i-1][w] \ (w < a_i)$ 입니다.
- 만약 $dp[i-1][w-a_i]$ 의 값이 0이라면 합이 $w - a_i$ 인 도미노 수열이 존재 하지 않으므로 $a_i$ 를 덧붙일 수 없습니다.

- 위 점화식은 $a_i$ 이상의 무게를 갖는 모든 도미노 수열에서 $a_i$ 가 덧붙여졌고 길이도 1 증가 한다는 특징을 가집니다.
- $w$ 가 굉장히 클 수 있기 때문에 모든 $w$ 를 Naive하게 탐색하며 dp식을 적용 할 수 없습니다.
- 존재하는 $w$ 만 관리하여 $a_i$ 를 덧붙이면 한번의 작업에 $O(N)$ 의 시간이 걸리게 됩니다.

- 레이지 세그먼트 트리를 이용하면 빠른 시간안에 도미노 수열의 합이 $a_i$ 이상인 모든 도미노 수열의 길이을 $1$ 증가 시키고 무게도 $a_i$ 증가 시킬 수 있습니다.
- $dp[i][w]$ 가 $1$ 이상인 모든 $w$ 는 $i+1$ 번째 연산 후, $w + a_{i+1} (a_{i+1} \leq w)$ 으로 값이 $1$ 더해져 전이하게 됩니다.
- 즉, 서로 다른 도미노 수열의 합의 크기 순서는 변하지 않습니다.

- $a_i$ 보다 합이 큰 도미노 수열이 존재하지 않는 경우에는 새로운 도미노 수열을 만들면 됩니다.
- $a_i$ 보다 합이 큰 도미노 수열이 존재한다면, $a_i$ 로 시작하는 도미노 수열을 만드는 것보다 그 수열에 덧붙이는 게 이득이기 때문입니다.
- 따라서 이 같은 경우는 가장 긴 도미노 수열이 될 수 없기 때문에, $a_i$ 로 시작하는 도미노 수열은 만들지 않습니다.

- 필요 없는 도미노 수열은 만들지 않으면서 도미노 수열들을 관리하게 되면 무게를 기준으로 정렬된 도미노 수열들을 관리 할 수 있습니다.
- 그러면 각 도미노 수열을 특정 인덱스로 잡고, 무게를 관리하는 레이지 세그먼트 트리와 길이를 관리하는 레이지 세그먼트 트리로 관리 할 수 있습니다.
- 무게를 관리하는 레이지 세그먼트 트리를 이용해서 덧붙일 수 있는 도미노 수열들이 있는 구간을 찾고, 레이지하게 무게와 길이를 업데이트 하면 됩니다.
- 이때, 길이는 실시간으로 파악하지 않아도 되므로 업데이트에 $O(1)$ 마지막에 $O(N)$ 으로 관리해줘도 됩니다.

- 도미노 수열들의 무게를 관리하고 무게로 이분 탐색을 하기 위해서는 구간 최댓값 세그먼트 트리를 이용하면 됩니다.
- 세그먼트 트리 왼쪽으로 우선 내려가며, 인자로 받은 무게 값보다 구간 최댓값이 더 작다면 리턴하여 올라오면 됩니다.
- 따라서 이 문제는 $O(N \log N)$ 으로 해결 할 수 있습니다.