SNUPC 2019 풀이

2019년 9월 7일

수고하셨습니다

- 총 참가자: 57명 (Div 2: 36명, Div 1: 21명)
- 총 제출 횟수: 854회 (Div 2: 586회, Div 1: 268회)
- 총 정답 횟수: 221회 (Div 2: 146회, Div 1: 75회)
- 각 문제별 슬라이드의 모든 통계는 프리즈된 스코어보드 기준입니다.

Div2A. 과자 사기

- 제출 87회, 정답 36명 (36명 시도, 정답률 41.38%)
- 처음 푼 참가자: 김현우 (3분)
- 출제자: Namnamseo

Div2A. 과자 사기

- 여섯 개의 값 p_S, w_S, p_N, w_N, p_U, w_U를 입력받습니다.
- 각각의 가격을 10배 하고, $5,000$ 이상이면 500을 뺍니다.
- 이제 (무게) $\div$ (가격)의 값을 서로 비교합니다.

Div2A. 과자 사기

실수형을 사용해도 되고, 다음과 같이 처리해도 됩니다.

양의 정수 a, b, c, d에 대해,

$$\frac{b}{a} < \frac{d}{c} \quad \Leftrightarrow \quad b \cdot c < d \cdot a$$

이므로 정수의 곱셈으로 분수를 비교할 수 있습니다.

Div2B. 평행 우주

- 제출 127회, 정답 31명 (36명 시도, 정답률 24.41%)
- 처음 푼 참가자: 한진제 (10분)
- 출제자: kipa00

Div2B. 평행 우주

- 문제 아이디어는 0.5x A Presses에서 왔습니다.
  - 오로지 A를 0.5번 더 적게 누르기 위해 평행 우주를 누비는 마리오를 구경하세요

Div2B. 평행 우주

- 행성 $(i-1)$에서 행성 i로 가는 데 드는 속도 v_i가 주어짐
- 이 속도의 양의 정수 배의 속도로 각 행성을 주어진 순서대로 방문
- 각 행성에서 원하는 만큼 속도를 떨어뜨릴 수 있음
- 초기 속력을 최소화

Div2B. 평행 우주

- 행성 $(i-1)$에서 행성 i로 가는 속력을 f_i로 놓으면 f_1을 최소화하면 됨
- 만족해야 하는 조건: $f_{i-1} \geq f_i$
- f_i이 작으면 f_{i-1}의 범위가 더 넓어짐
- 따라서 f_n부터 거꾸로 속력을 결정하는 그리디 알고리즘이 통합니다!

Div2C. Christmalo.win

- 제출 57회, 정답 22명 (29명 시도, 정답률 38.60%)
- 처음 푼 참가자: 최윤영 (27분)
- 출제자: kazel

Div2C. Christmalo.win

- 앞에 올 길이 S인 단어를 하나 선택했다고 가정해 봅시다.

Div2C. Christmalo.win

- 앞에 올 길이 S인 단어를 하나 선택했다고 가정해 봅시다.
- i번째 위치의 알파벳을 선택한다고 했을 때 지워지는 문자의 개수는 $(S - i - 1)$개

Div2C. Christmalo.win

- 앞에 올 길이 S인 단어를 하나 선택했다고 가정해 봅시다.
- i번째 위치의 알파벳을 선택한다고 했을 때 지워지는 문자의 개수는 $(S - i - 1)$개
- 각 알파벳별로 이 값들 중 제일 작은 값을 저장합시다.

Div2C. Christmalo.win

- 앞에 올 길이 S인 단어를 하나 선택했다고 가정해 봅시다.
- i번째 위치의 알파벳을 선택한다고 했을 때 지워지는 문자의 개수는 $(S - i - 1)$개
- 각 알파벳별로 이 값들 중 제일 작은 값을 저장합시다.
- 비슷한 방법으로 단어가 뒤에 오는 경우도 계산할 수 있습니다.

Div2C. Christmalo.win

- 지금까지 봤던 단어 중 하나와 현재 보고 있는 단어를 선택했다고 가정해 봅시다.

Div2C. Christmalo.win

- 지금까지 봤던 단어 중 하나와 현재 보고 있는 단어를 선택했다고 가정해 봅시다.
- 지금까지 봤던 단어 중, 앞 또는 뒤에 오면서 특정 알파벳을 선택했을 때 가장 좋은 결과가 나오는 값을 저장합니다.

Div2C. Christmalo.win

- 지금까지 봤던 단어 중 하나와 현재 보고 있는 단어를 선택했다고 가정해 봅시다.
- 지금까지 봤던 단어 중, 앞 또는 뒤에 오면서 특정 알파벳을 선택했을 때 가장 좋은 결과가 나오는 값을 저장합니다.
- 저장된 값과 현재 보고 있는 단어의 값을 합해서 답을 찾을 수 있습니다.

Div2C. Christmalo.win

- 지금까지 봤던 단어 중 하나와 현재 보고 있는 단어를 선택했다고 가정해 봅시다.
- 지금까지 봤던 단어 중, 앞 또는 뒤에 오면서 특정 알파벳을 선택했을 때 가장 좋은 결과가 나오는 값을 저장합니다.
- 저장된 값과 현재 보고 있는 단어의 값을 합해서 답을 찾을 수 있습니다.
- 시간 복잡도는 $26N + \sum |S_i|$

Div2D. 두 개의 문

- 제출 43회, 정답 9명 (12명 시도, 정답률 20.93%)
- 처음 푼 참가자: 조승한 (99분)
- 출제자: kipa00

Div2D. 두 개의 문

- $+1$ 혹은 -1로 구성된 길이 n의 수열이 주어짐
- 가능한 연산 m개는 정해진 위치의 수에 (-1)을 곱하는 것
- 합이 $(-n)$부터 n까지가 되는 경우를 각각 찾아 출력

Div2D. 두 개의 문

- 기본적인 관찰
  - 각 연산을 적용하는 순서는 크게 상관없다
  - 각 연산을 두 번 적용하면 적용하지 않은 것과 같다
- 따라서 전부 다 해 보면 됩니다!
- 각 경우 2^m, 계산하는 데 걸리는 시간 nm이므로 $\mathcal{O}(nm2^m)$
- 통과하기에는 역부족

Div2D. 두 개의 문

- 시간을 더 줄이자
  - 목표: 이전 상태에서 다음 상태를 얻는 데 최대 한 번의 연산만 하자
- 각 경우를 방문하는 순서를 재귀 함수나 grey code 등으로 방문
- 연산 1회에는 최악의 경우 시간 n이 걸리므로 $\mathcal{O}(n2^m)$

Div2D. 두 개의 문

- **시간을 더 줄이자**
  - 목표: 연산 1회에 걸리는 시간을 n보다 더 줄이기?

Div2D. 두 개의 문

- **시간을 더 줄이자**
  - 목표: 연산 1회에 걸리는 시간을 n보다 더 줄이기?
- 수열의 상태는 1bit로 표현 가능
- 다른 상태가 되는 것은 정해진 bit를 negate하는 것
- 선택적 bit-wise negation?

Div2D. 두 개의 문

- **시간을 더 줄이자**
  - 목표: 연산 1회에 걸리는 시간을 n보다 더 줄이기?
- 수열의 상태는 1bit로 표현 가능
- 다른 상태가 되는 것은 정해진 bit를 negate하는 것
- 선택적 bit-wise negation?
- xor!

Div2D. 두 개의 문

- SIMD(Single Instruction, Multiple Data)라고도 볼 수 있음
  - xor 연산 한 번에 데이터 32개 혹은 64개를 다루게 됩니다

Div2D. 두 개의 문

- 이외에 bit를 세는 것, grey code의 bit를 찾는 것 등도 16개 단위에 대해 한꺼번에 다룰 수 있음
  - 혹은 `__builtin_ctz`, `__builtin_popcount` 등의 gcc 비표준 함수를 이용할 수도 있습니다
- 시간 복잡도: $\mathcal{O}(n2^m/c)$ where $c = 32$ or 64

Div2E/Div1E. 순위 계산

- 제출 99회, 정답 20명 (30명 시도, 정답률 20.20%)
- 처음 푼 참가자: 서영호 (37분)
- 출제자: bryan

Div2E/Div1E. 순위 계산

- 가장 낮은 순위를 먼저 생각해봅시다.
- 현재 탐레프의 순위 이상인 사람이 전부 탐레프보다 점수가 높으면 됩니다.

Div2E/Div1E. 순위 계산

- 가장 높은 순위는, 현재 탐레프의 순위와 같은 사람이 전부 탐레프와 점수가 같으면 될 것 같습니다.
- (예제 2가 없었다면 많은 분들이 여기서 못 넘어갔을지도 모릅니다.)

Div2E/Div1E. 순위 계산

- 탐레프가 심사 직후 4위이고, 다음 참가자 세 명이 연속으로 같은 4위를 받았고, 그 다음 참가자가 6위가 되는 상황을 생각해봅시다.
- 이때 탐레프가 위치할 수 있는 가장 높은 순위는 6위입니다.
  - 공동 4위가 될 수 있는 사람은 최대 2명입니다.
  - 탐레프 이후에 심사를 받은 사람은 탐레프보다 점수가 높거나 같으므로, 탐레프는 2명 안에 들 수 없습니다.
  - 즉, 탐레프보다 두 명이 점수를 높게 받은 상황이 최선이 됩니다.

Div2E/Div1E. 순위 계산

- 즉, 탐레프의 순위와 동점자의 수를 기록하면서, 최선의 순위를 상황에 맞게 낮추면(=값을 높이면) 됩니다.
- 가장 낮은 순위와 가장 높은 순위 모두 $\mathcal{O}(N)$에 구할 수 있습니다.

Div2F. 갓

- 제출 85회, 정답 19명 (28명 시도, 정답률 22.35%)
- 처음 푼 참가자: 김현우 (43분)
- 출제자: doju

Div2F. 갓

- 동현이가 이기는 투표가 있다면 거기에 엄청나게 많은 표를 몰아 주는 것이 이득입니다.
- 그러나 예제에서는 동현이가 두 투표에서 모두 졌음에도 불구하고 *진정한 갓*이 되었습니다.
- 어떻게 이런 일이 발생했을까요?

Div2F. 갓

- 동현이가 진행하는 투표는 현수와 승원이의 표만 늘려 주므로, 참가자가 적을수록 좋습니다.
- 나머지 두 투표의 결과는 미지수로 나타낼 수 있습니다.
  - 현수가 진행한 투표에서 동현이는 HDx표, 승원이는 HSx표를 받았습니다.
  - 승원이가 진행한 투표에서 동현이는 SDy표, 현수는 SHy표를 받았습니다.

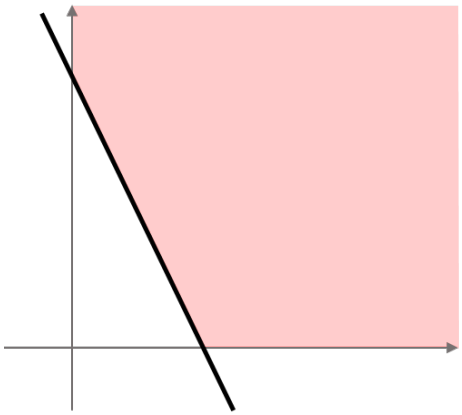
Div2F. 갓

- 먼저 현수를 이겨 봅시다.
- 현수를 이기기 위해서는 다음의 조건문이 성립해야 합니다.

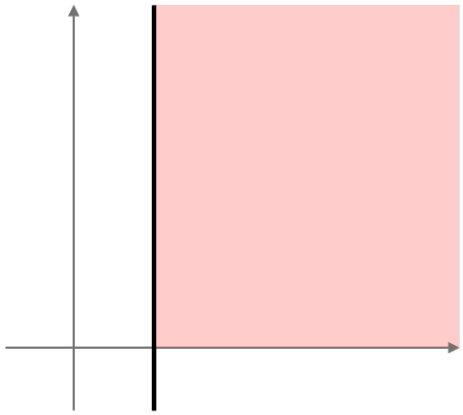
$$HDx + SDy > SHy + DH$$
$$\Rightarrow x > \frac{SH - SD}{HD}y + \frac{DH}{HD}$$

- 이를 좌표평면에 나타내 보면 $(\frac{DH}{HD}, 0)$을 지나는 직선임을 쉽게 발견할 수 있습니다.

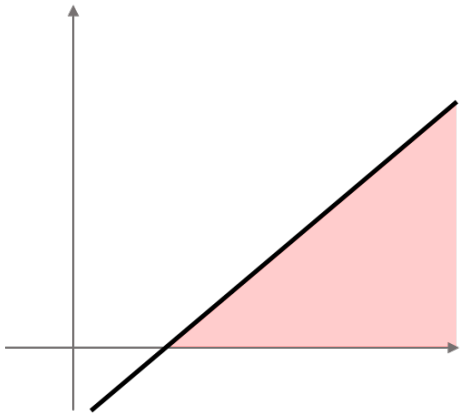
Div2F. 갓



현수에게 이겼을 때
(SD > SH)



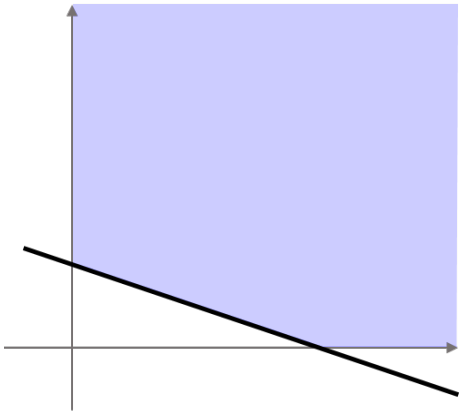
현수와 비겼을 때
(SD = SH)



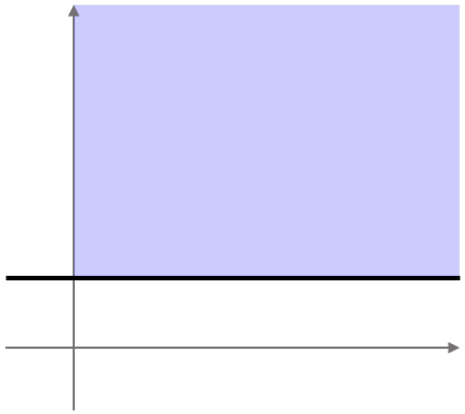
현수에게 졌을 때
(SD < SH)

- 승원이가 진행한 투표의 결과에 따라 다양한 그래프가 나옵니다.

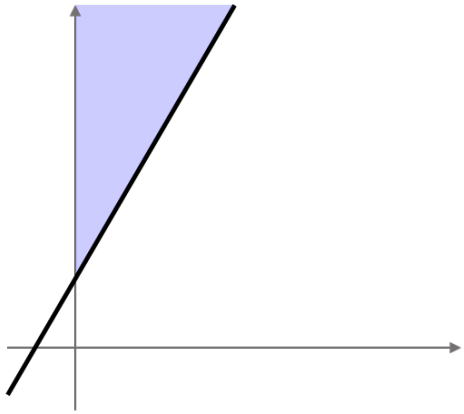
Div2F. 갓



승원이에게 이겼을 때
(HD > HS)



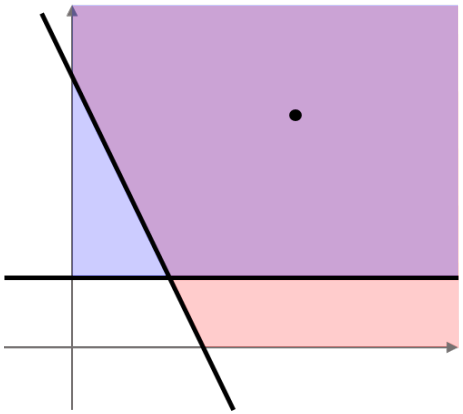
승원이와 비겼을 때
(HD = HS)



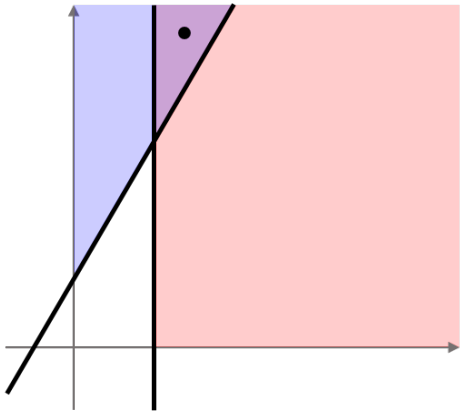
승원이에게 졌을 때
(HD < HS)

- 마찬가지로 승원이를 이기기 위한 조건은 현수가 진행한 투표의 결과에 따라 다른 그림이 나옵니다.

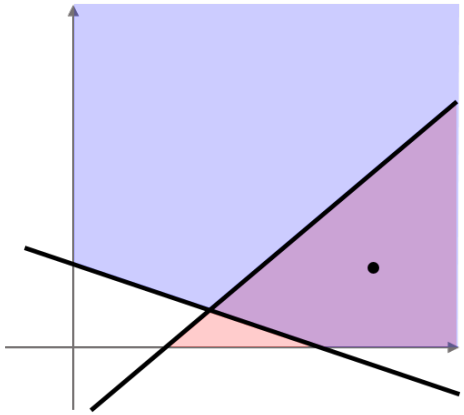
Div2F. 갓



SD > SH, HD = HS



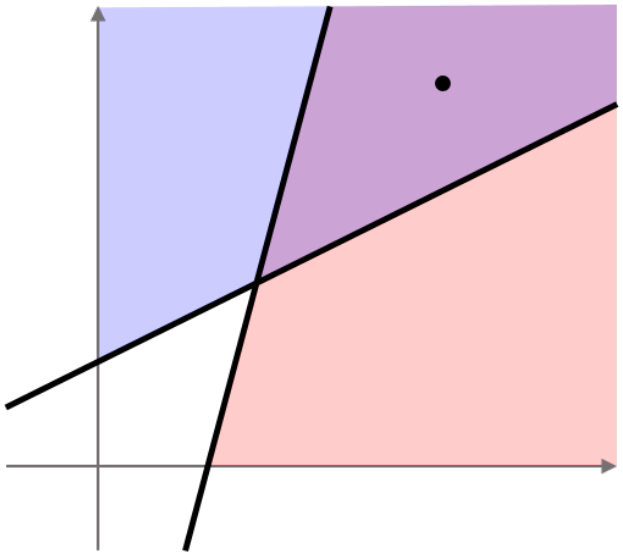
SD = SH, HD < HS

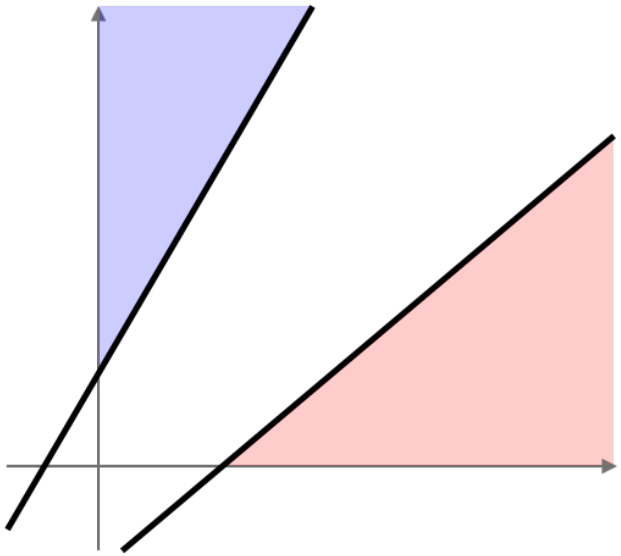


SD < SH, HD > HS

- 두 영역이 겹쳐지는 영역에서 (x, y)를 아무렇게나 뽑으면 동현이가 현수와 승원이를 모두 이길 수 있습니다.
- 동현이가 적어도 하나의 투표에서 이기거나 비긴다면 *진정한 갓*이 될 수 있습니다.

Div2F. 갓





- 동현이가 두 투표에서 모두 진다면 위와 같은 두 가지 경우가 있을 수 있습니다.
- 후자의 경우는 두 영역이 겹치지 않으므로 동현이는 *진정한 갓*이 될 수 없습니다.

Div2F. 갓

- 어떤 경우에 두 영역이 겹치지 않는지 식을 구하면 간단한 조건문으로 문제를 풀 수 있습니다.
- 동현이가 *진정한 갓*이 되기 위해 생각보다 훨씬 많은 표를 받아야 할 수도 있습니다.
  - `9 10 9 10 1 9`와 같은 입력에서 동현이가 이기기 위해서는 최소 991표를 받아야 합니다.

Div2G. kdh9949

- 제출 32회, 정답 6명 (10명 시도, 정답률 18.75%)
- 처음 푼 참가자: 김상민 (157분)
- 출제자: kdh9949

Div2G. kdh9949

- 입력으로 주어지는 간선에는 방향이 없기 때문에, 어떻게 시작해야 할지 잘 모르겠습니다.
- 간선에 방향을 매겨 줄 수 있을까요?

Div2G. kdh9949

- 우선, 편의상 그래프에서 알파벳 `X`가 적힌 정점을 그냥 `X`라 부르겠습니다.
- 우리가 찾는 경로에서 가능한 이동은 $\texttt{K} \to \texttt{D}$, $\texttt{D} \to \texttt{H}$, $\texttt{H} \to \texttt{K}$ 뿐입니다.

Div2G. kdh9949

- 우선, 편의상 그래프에서 알파벳 `X`가 적힌 정점을 그냥 `X`라 부르겠습니다.
- 우리가 찾는 경로에서 가능한 이동은 $\texttt{K} \to \texttt{D}$, $\texttt{D} \to \texttt{H}$, $\texttt{H} \to \texttt{K}$ 뿐입니다.
- 즉, 모든 간선에 대해 이 간선이 연결하는 두 정점의 알파벳에 따라 간선의 방향이 결정됩니다!
- (연결하는 두 정점의 알파벳이 같다면, 그 간선은 그냥 무시해도 됩니다.)

Div2G. kdh9949

- 이제, 입력으로 주어지는 그래프를 유향 그래프로 바꿔 줄 수 있습니다.
- 이 그래프에는 사이클이 있거나, 사이클이 없습니다.

Div2G. kdh9949

Observation 1.

그래프에 사이클이 있으면 무한히 긴 KDHKDH... 경로가 존재한다.

Div2G. kdh9949

Observation 1.

그래프에 사이클이 있으면 무한히 긴 KDHKDH... 경로가 존재한다.

- 사이클에는 K가 반드시 있습니다.
  - 간선 방향의 제약 조건 때문에, D와 H만 가지고는 죽어도 사이클을 못 만듭니다.
- 모든 정점에 대해, 그 정점의 알파벳이 결정되면 거기서 간선을 하나 타고 나갔을 때 만나는 알파벳도 결정됩니다.
- 따라서, K에서 시작해서 사이클을 계속 빙글빙글 돌면 그 경로는 KDHKDH...일 수 밖에 없습니다!
- 경로를 무한히 길게 만들 수 있으므로 답은 -1.

Div2G. kdh9949

- 사이클이 없는 경우는 어떻게 될까요?
- 사이클 없는 유향 그래프를 DAG (Directed Acyclic Graph) 라고 합니다.
- PS 경험이 어느 정도 있는 분들이 DAG를 보면 무슨 생각이 들까요?

Div2G. kdh9949

- 사이클이 없는 경우는 어떻게 될까요?
- 사이클 없는 유향 그래프를 DAG (Directed Acyclic Graph) 라고 합니다.
- PS 경험이 어느 정도 있는 분들이 DAG를 보면 무슨 생각이 들까요?
- **위상 정렬**을 한 뒤 **DP**를 하고 싶어질 겁니다.

Div2G. kdh9949

- 위상 정렬이란, 어떤 유향 그래프에 대해 다음 성질을 만족하는 정점들의 순서를 구하는 것입니다.
  - 그래프의 임의의 간선 $u \to v$에 대해, u가 v보다 순서상으로 앞에 있다.
- 간선의 형태로 주어지는 선후 관계가 있을 때, 위상 정렬을 한다면 이 관계가 꼬이지 않도록 올바른 순서로 모든 정점에 대해 처리를 해 줄 수 있습니다.
- 구현은 각 정점의 indegree(그 정점으로 들어가는 간선 개수)를 관리하면서 queue를 활용해서 할 수 있습니다. 자세한 건 검색 부탁..

Div2G. kdh9949

- 그래프에 사이클이 있는지 없는지는 어떻게 알까요?

Div2G. kdh9949

- 그래프에 사이클이 있는지 없는지는 어떻게 알까요?
- 위상 정렬을 시도해 봤을 때, 실패했다면 사이클이 있는 것입니다!
- 일단 주어진 그래프에 대해 위상 정렬을 해 보고, 안 된다면 -1을 찍고 나가면 됩니다.

Div2G. kdh9949

- 그래프를 위상 정렬하는 데 성공했다면, 순서대로 다음과 같이 정의된 DP 값을 구해 줍시다.

- $D_i = i$번 정점에서 끝나는 경로 중 가장 긴 것의 길이.

- 각 간선 $u \to v$에 대해 $D_v = \max(D_v, D_u + 1)$ 과 같은 식으로 갱신을 해 주면 $O(N + M)$ 시간복잡도로 계산할 수 있습니다.
- 모든 정점에 대해 이 값을 다 구했다면, $\max_{C_i = \mathtt{H}}(3 * [\frac{D_i}{3}])$ 이 답이 됩니다.
- 위에서 구한 답이 0일 경우에는 -1을 출력해야 함에 유의하세요.

Div2H/Div1C. 성냥팔이 소년

- 제출 34회, 정답 3명 (12명 시도, 정답률 8.82%)
- 처음 푼 참가자: 최민기 (178분)
- 출제자: kdh9949

Div2H/Div1C. 성냥팔이 소년

- 일단, 0은 **무조건** 안 고르는 게 좋습니다.
- 고로, 0은 일단 버리고 시작해도 좋겠습니다.
- 모든 정수의 절댓값이 1 이상이라고 가정합시다.

Div2H/Div1C. 성냥팔이 소년

- 우선, 우리가 '고를' (성냥개비를 1개 이상 사용할) 수들의 집합 S를 고정했다고 해 봅시다.
- 다음의 3가지 경우가 있습니다.
  - $|S| = 0$
  - $1 \leq |S| < K$
  - $|S| = K$

Div2H/Div1C. 성냥팔이 소년

1. $|S| = 0$

- $T = 1$입니다.
- 1이 답의 후보에 추가됩니다.
- 입력의 모든 수가 양수인데 $K = 1$인 경우와 같이 이 케이스가 답이 되는 경우가 존재합니다!

Div2H/Div1C. 성냥팔이 소년

2. $1 \leq |S| < K$

- 일단 모든 수에 성냥개비를 하나씩 써 봅시다.
- $|S| < K$이므로, 남는 성냥개비가 있습니다.
- 그 성냥개비를 아무데나 붙이거나 말거나 하는 것으로 T의 값을 항상 양수로 만들 수 있습니다.
- T의 절댓값이 클수록 좋으므로, 입력된 수들 중에서 절댓값이 가장 큰 $\min(N, K-1)$개를 고르면 됩니다.

Div2H/Div1C. 성냥팔이 소년

3. $|S| = K$

- $K > N$이면 이 케이스는 고려할 필요가 없습니다. $K \leq N$이라 합시다.
- 일단 T의 절댓값이 크면 좋을 것 같으므로, 앞에서 했던 것과 비슷하게 절댓값이 가장 큰 K개의 수를 골라 봅시다.
- 만약 이 때 T가 양수라면, 그 값이 정답입니다.
- T가 음수라면, 고른 수들 중 몇 개를 대체해서 T의 부호를 바꿔야 합니다.

Div2H/Div1C. 성냥팔이 소년

3.1. $|S| = K$, 절댓값이 가장 큰 K개를 골랐을 때 T가 음수

- 생각을 좀 해 보면, 다음의 2가지 경우만 고려하면 됨을 알 수 있습니다.
  - 고른 수들 중 절댓값이 가장 작은 양수를 빼고, 고르지 않은 수들 중 절댓값이 가장 큰 음수를 고른다.
  - 고른 수들 중 절댓값이 가장 작은 음수를 빼고, 고르지 않은 수들 중 절댓값이 가장 큰 양수를 고른다.
- 상황에 따라 두 경우가 모두 가능할 수도 있고, 둘 중 하나만 가능할 수도 있고, 둘 다 불가능할 수도 있습니다.

Div2H/Div1C. 성냥팔이 소년

- **3.1.**의 경우, 앞에서 이야기한 2가지 경우와 더불어 **2.**, 즉 $|S| < K$인 경우도 답의 후보가 됩니다.
- 총 3가지 케이스에 대해서 뭐가 가장 좋은지를 결정해 줘야 하는데, T의 값은 매우 커질 수 있으므로 각각에 대해 T를 직접 구할 수는 없습니다.
- 3가지 케이스에 대해 공통으로 곱해지는 수가 많으므로, 이들을 미리 약분해주면 케이스 간 비교를 `long long` 범위에서 해줄 수 있습니다.

Div2H/Div1C. 성냥팔이 소년

- 참고로 구현이 생각보다 짜증납니다. 케이스 분류를 신중하게 하지 않으면 짜기 매우 힘들 것입니다.
- 양수/음수 각각에서 절댓값이 큰 것부터 고르면 이득이라는 것까지 관찰하고, $\log(T)$의 값을 구해서 가장 크게 되는 시점을 구하는 식으로 풀 수도 있습니다.
- (시도하는 사람이 있으려나..)

Div2I. 필살! 60단 컴보

- 제출 5회, 정답 0명 (2명 시도, 정답률 0.00%)
- 처음 푼 참가자: 없음
- 출제자: 16silver

Div2I. 필살! 60단 컴보

- 일단 문제를 수학적으로 나타내 봅시다.
- 플레이 기록을 나타낸 2진법의 수에서, 길이가 1인 연속된 구간 각각마다 길이의 제곱을 더한 값이 점수입니다.
- 이 함수를 f라 하면, 구하려는 값은 $f(n) > f(c)$이고 $a \leq n \leq b$인 n의 개수에 1을 더한 값입니다.

Div2I. 필살! 60단 컴보

- 개수를 셀 때, ($0 \le n \le b$일 때의 개수) - ($0 \le n \le a-1$일 때의 개수)를 구하면 됩니다.
- 따라서 우리는 $f(n) \ge C$이고 $0 \le n \le b$인 n의 개수를 빠르게 세는 방법을 찾으면 됩니다.
- 플레이 기록은 최대 $M = 2^{60} - 1$까지 가능하기 때문에 일일히 세는 것은 불가능합니다.
- 하지만 적당한 전처리만 해둔다면 각 쿼리를 $\mathcal{O}(\log M)$에 처리할 수 있습니다!

Div2I. 필살! 60단 컴보

- 필요한 전처리는 다음과 같습니다.
- $dp(i, j) = (0 \leq n \leq 2^i - 1$이고 $f(n) \geq j$ 인 n의 개수)

$$dp(i, j) = \sum_{k=0}^{i} dp(i - k - 1, max(j - k \cdot k, 0)) \quad (0 \leq j \leq i^2)$$

- $dp(-1, 0) = 1$로 둡니다.
- 참고: $dp(i, j) = (0 \leq n \leq 2^i - 1$이고 $f(n) = j$인 n의 개수)로 정의해도 점화식을 세울 수 있습니다.
- 이 때의 점화식을 이용하면 커팅으로 계산량을 1/5로 줄일 수 있습니다.

Div2I. 필살! 60단 컴보

- 예시를 먼저 봅시다! ($b = 26 = 11010_{(2)}, C = 3$)
- 0 이상 $b + 1 = 27 = 11011_{(2)}$ 미만의 모든 수들은 2진법으로 나타냈을 때 다음 중 정확히 하나에 속합니다.
- 0XXXX, 10XXX, 1100X, 11010 (X는 1 또는 0)

Div2I. 필살! 60단 컴보

- 그러므로 다음과 같이 구간을 나누어 셉니다.
- 0XXXX: $00000_{(2)}$ ~ $01111_{(2)}$ 에서 $dp(4, 3) = 8$개
- 10XXX: $10000_{(2)}$ ~ $10111_{(2)}$ 에서 $dp(3, 2) = 4$개
- 1100X: $11000_{(2)}$ ~ $11001_{(2)}$ 에서 $dp(1, \mathbf{0}) = 2$개
- 11010: $11010_{(2)}$ ~ $11010_{(2)}$ 에서 $dp(0, \mathbf{0}) = 1$개

Div2I. 필살! 60단 컴보

- 일반적으로 $f(n) \geq C$이고 $0 \leq n \leq b$인 n의 개수를 세는 방법
- $b+1$의 비트들 중 1인 비트마다, 그 비트을 0으로 바꾸고 뒤의 비트가 0...0 ~ 1...1일 때의 개수를 셉니다.
- 식으로 나타내면 다음과 같습니다. ($bit(x, y)$는 x의 2^y자리 값, $cut(x, y)$는 x의 뒤 y비트를 자른 값)

$$ans = \sum_{0 \leq k \leq 60, bit(b+1,k)=1} dp(i, max(c - f(cut(b+1, k+1)), 0))$$

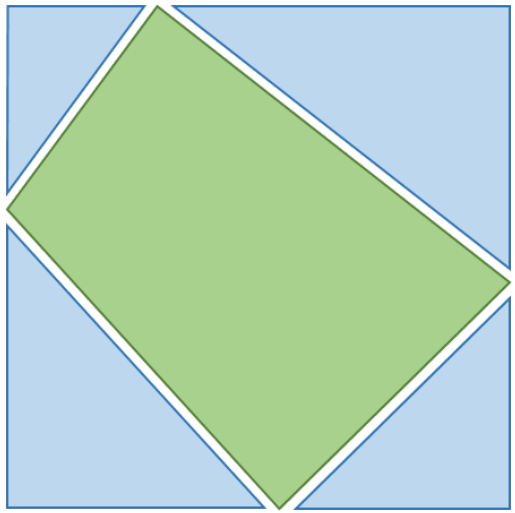
- $f(cut(b+1, k+1))$은 k를 $60 \to 0$ 순으로 잘 관리합니다.
- 시간복잡도는 전처리 $\mathcal{O}((\log M)^4)$, 쿼리 $\mathcal{O}(T \log M)$

Div2J/Div1B. 라이언 동상 구하기

- 제출 1회, 정답 0명 (1명 시도, 정답률 0.00%)
- 처음 푼 참가자: 없음
- 출제자: kazel

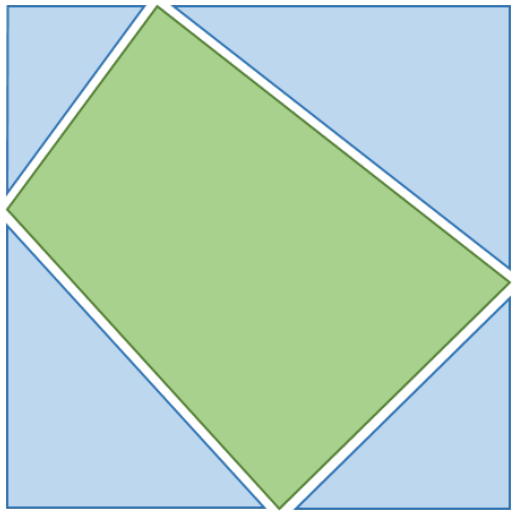
Div2J/Div1B. 라이언 동상 구하기

- 새로운 땅은 원래의 땅에서 네 귀퉁이를 자른 것으로 볼 수 있습니다.



Div2J/Div1B. 라이언 동상 구하기

- 새로운 땅은 원래의 땅에서 네 귀퉁이를 자른 것으로 볼 수 있습니다.



- 전체 동상의 가치의 합에서 네 귀퉁이에 있는 동상의 가치의 합을 빼면 새로운 땅에 있는 동상의 가치의 합이 됩니다.

Div2J/Div1B. 라이언 동상 구하기

- 각 귀퉁이를 자를 수 있는 방법별로 값을 계산해야 합니다.
  왼쪽 위 귀퉁이의 값을 구하는 경우를 예로 들어 봅시다.

Div2J/Div1B. 라이언 동상 구하기

- 각 귀퉁이를 자를 수 있는 방법별로 값을 계산해야 합니다. 왼쪽 위 귀퉁이의 값을 구하는 경우를 예로 들어 봅시다.
- 가능한 각 선분별로, 각각의 동상이 이 선분의 왼쪽에 있는지 오른쪽에 있는지 판별하면 됩니다.

Div2J/Div1B. 라이언 동상 구하기

- 신발끈 공식을 응용하여 이 판정을 쉽게 할 수 있습니다.

$$A = \frac{1}{2} \left| \begin{matrix} x_1 & x_2 & x_3 & x_1 \\ y_1 & y_2 & y_3 & y_1 \end{matrix} \right|$$

Div2J/Div1B. 라이언 동상 구하기

- 신발끈 공식을 응용하여 이 판정을 쉽게 할 수 있습니다.

$$A = \frac{1}{2} \left| \begin{matrix} x_1 & x_2 & x_3 & x_1 \\ y_1 & y_2 & y_3 & y_1 \end{matrix} \right|$$

- 결과가 양수가 나오는 경우 세 점이 반시계 방향으로, 결과가 음수가 나오는 경우는 세 점이 시계 방향으로 놓여져 있습니다.

Div2J/Div1B. 라이언 동상 구하기

- 신발끈 공식을 응용하여 이 판정을 쉽게 할 수 있습니다.

$$A = \frac{1}{2} \left| \begin{matrix} x_1 & x_2 & x_3 & x_1 \\ y_1 & y_2 & y_3 & y_1 \end{matrix} \right|$$

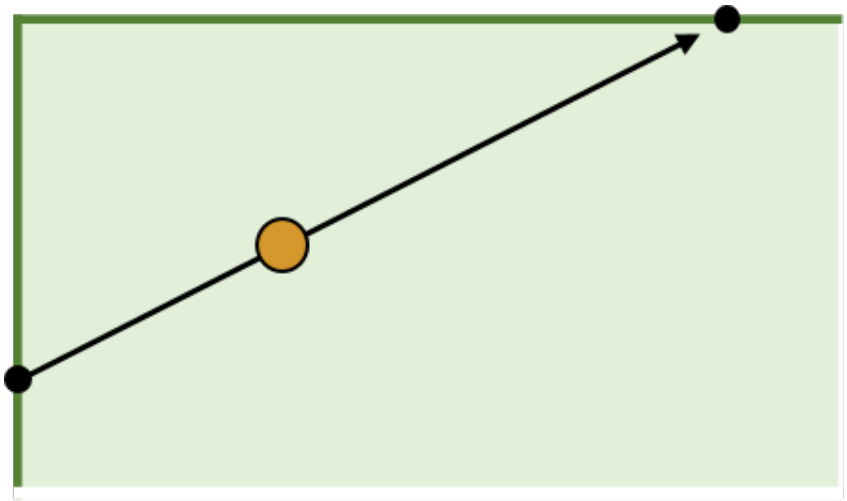
- 결과가 양수가 나오는 경우 세 점이 반시계 방향으로, 결과가 음수가 나오는 경우는 세 점이 시계 방향으로 놓여져 있습니다.
- 시간복잡도는 $\mathcal{O}(N^2 M)$이 됩니다.

Div2J/Div1B. 라이언 동상 구하기

- 사실 조금 더 빠른 방법이 있습니다.

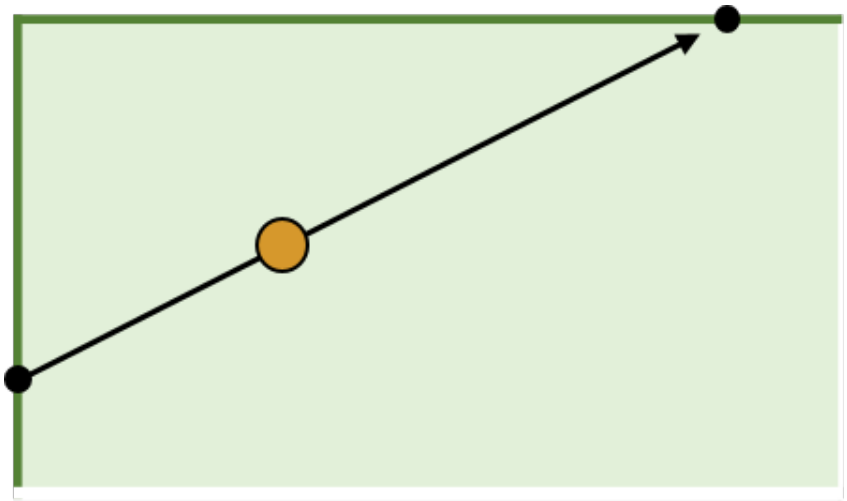
Div2J/Div1B. 라이언 동상 구하기

- 사실 조금 더 빠른 방법이 있습니다.
- 왼쪽 점에서 동상을 지나가는 반직선을 긋습니다.



Div2J/Div1B. 라이언 동상 구하기

- 사실 조금 더 빠른 방법이 있습니다.
- 왼쪽 점에서 동상을 지나가는 반직선을 긋습니다.



- 이 반직선과 위의 선분의 오른쪽 방향으로 위 점을 선택한 경우에 이 동상이 귀퉁이에 포함됩니다.

Div2J/Div1B. 라이언 동상 구하기

- 동상이 포함되게 되는 위 점들 중 제일 왼쪽 값을 선택하여 그 위치에 그 동상의 가치를 더합니다.

Div2J/Div1B. 라이언 동상 구하기

- 동상이 포함되게 되는 위 점들 중 제일 왼쪽 값을 선택하여 그 위치에 그 동상의 가치를 더합니다.
- 이 값을 모든 동상에 대해 계산한 후, 누적합을 계산하면 원하는 값을 얻을 수 있습니다.

Div2J/Div1B. 라이언 동상 구하기

- 동상이 포함되게 되는 위 점들 중 제일 왼쪽 값을 선택하여 그 위치에 그 동상의 가치를 더합니다.
- 이 값을 모든 동상에 대해 계산한 후, 누적합을 계산하면 원하는 값을 얻을 수 있습니다.
- 시간복잡도는 $\mathcal{O}(NM)$이 됩니다.

Div2J/Div1B. 라이언 동상 구하기

- 이제 실제 문제의 답을 구할 차례입니다.

Div2J/Div1B. 라이언 동상 구하기

- 이제 실제 문제의 답을 구할 차례입니다.
- 좌우의 점을 하나씩 고르면 위아래의 점은 서로 독립적으로 계산할 수 있습니다.

Div2J/Div1B. 라이언 동상 구하기

- 이제 실제 문제의 답을 구할 차례입니다.
- 좌우의 점을 하나씩 고르면 위아래의 점은 서로 독립적으로 계산할 수 있습니다.
- 이 과정의 시간복잡도는 $\mathcal{O}(N^3)$이 됩니다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 제출 3회, 정답 1명 (1명 시도, 정답률 33.33%)
- 처음 푼 참가자: 정재윤 (61분)
- 출제자: doju

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 제목과 내용, **문제 순서** 모두 재작년 문제의 오마주입니다.
- 다양한 풀이가 나올 수 있기에 제한을 여유롭게 잡으려고 했습니다.
  - cNM 꼴의 풀이일 때 $c \leq 25$

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 먼저 불가능한 경우를 추려 봅시다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 먼저 불가능한 경우를 추려 봅시다.
- 각 색깔을 다음과 같이 수에 대응시킵니다.
  - $R \to 0$, $B \to 1$, $Y \to 2$

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 먼저 불가능한 경우를 추려 봅시다.
- 각 색깔을 다음과 같이 수에 대응시킵니다.
  - $\mathrm{R} \to 0$, $\mathrm{B} \to 1$, $\mathrm{Y} \to 2$
- 합체를 하더라도 두 수의 합을 3으로 나눈 나머지는 바뀌지 않습니다.
  - $\mathrm{R} + \mathrm{B} = \mathbf{1}$, $\mathrm{Y} + \mathrm{Y} = \mathbf{4}$
  - $\mathrm{R} + \mathrm{Y} = \mathbf{2}$, $\mathrm{B} + \mathrm{B} = \mathbf{2}$
  - $\mathrm{B} + \mathrm{Y} = \mathbf{3}$, $\mathrm{R} + \mathrm{R} = \mathbf{0}$

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 전체 슬라임의 마릿수가 3의 배수가 아니라면 모든 슬라임이 합쳐질 색이 유일하게 결정됩니다.
  - 예제와 같이 "RB BY"라면 수의 합이 **4**이므로 **파란색**으로 합쳐져야 합니다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

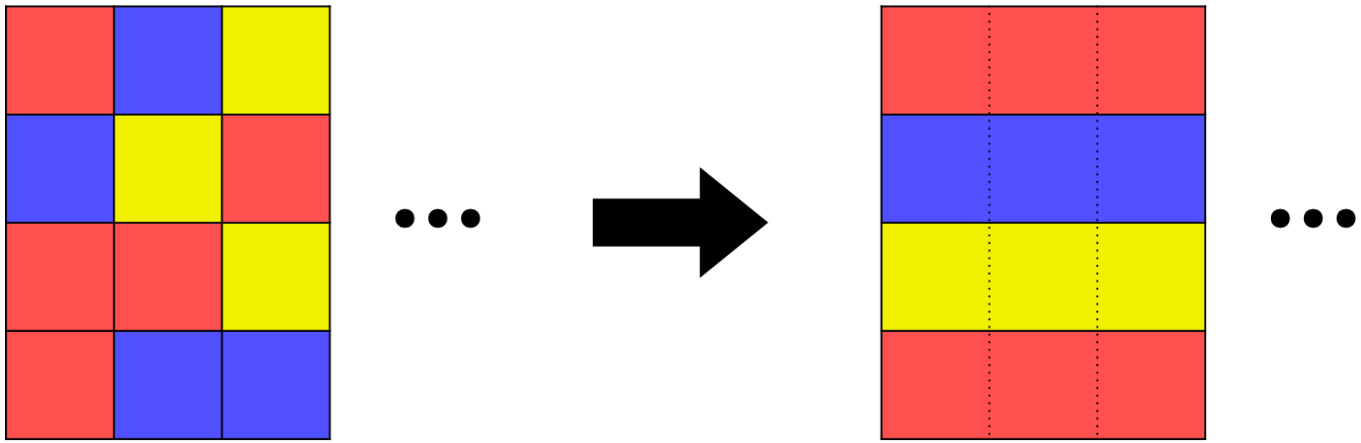
- 전체 슬라임의 마릿수가 3의 배수가 아니라면 모든 슬라임이 합쳐질 색이 유일하게 결정됩니다.
  - 예제와 같이 "RB BY"라면 수의 합이 **4**이므로 **파란색**으로 합쳐져야 합니다.
- 전체 슬라임의 마릿수가 3의 배수라면 수의 합은 반드시 3의 배수여야 합니다.
  - 그렇지 않다면 자명하게 **불가능합니다**.
  - 이를 만족하더라도 어떤 색으로 합쳐질지는 모릅니다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 풀이의 접근 방법은 수의 총합을 3으로 나눈 나머지를 유지하면서 크기를 줄여 나가는 것입니다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 풀이의 접근 방법은 수의 총합을 3으로 나눈 나머지를 유지하면서 크기를 줄여 나가는 것입니다.



- 위와 같이 바꾸면 왼쪽 세 열의 총합은 3의 배수임이 보장되므로 나머지에 영향을 미치지 않습니다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

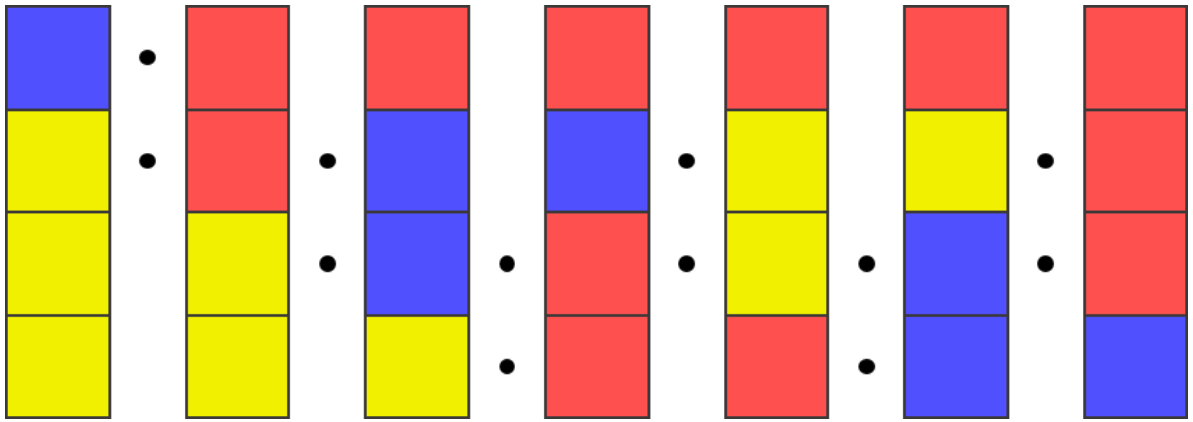
- 아쉽게도 1×3 구간은 하나의 색으로 합칠 수 없는 경우가 있었습니다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 아쉽게도 1×3 구간은 하나의 색으로 합칠 수 없는 경우가 있었습니다.
- 하지만 1×4 구간에서 앞의 세 마리를 같은 색으로 만든다면 어떨까요?

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 아쉽게도 1×3 구간은 하나의 색으로 합칠 수 없는 경우가 있었습니다.
- 하지만 1×4 구간에서 앞의 세 마리를 같은 색으로 만든다면 어떨까요?
  - 최대 6번의 합체로 항상 가능합니다!

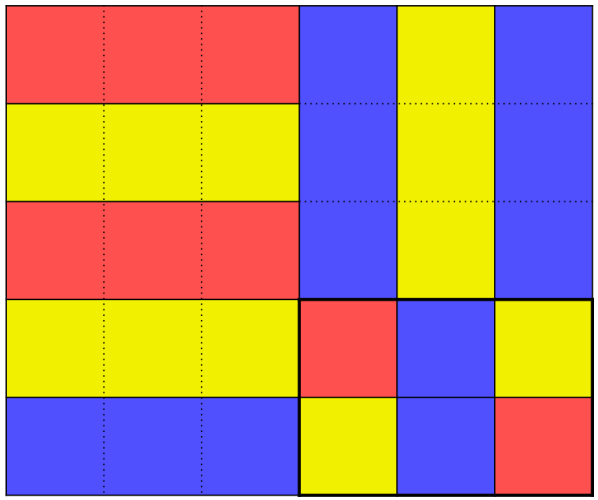


Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 이렇게 세 열씩 줄여 나가면 3개 이하의 열이 남습니다.
- 같은 방법으로 세 행씩 줄여 나가면 3개 이하의 행이 남습니다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 이렇게 세 열씩 줄여 나가면 3개 이하의 열이 남습니다.
- 같은 방법으로 세 행씩 줄여 나가면 3개 이하의 행이 남습니다.



- 이제 3×3 이하 크기의 조그마한 문제가 되었습니다!

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 한 변의 길이가 1 또는 2라면 간단하게 풀 수 있습니다.
- 3×3 크기는 손으로 풀기에는 조금 어렵습니다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

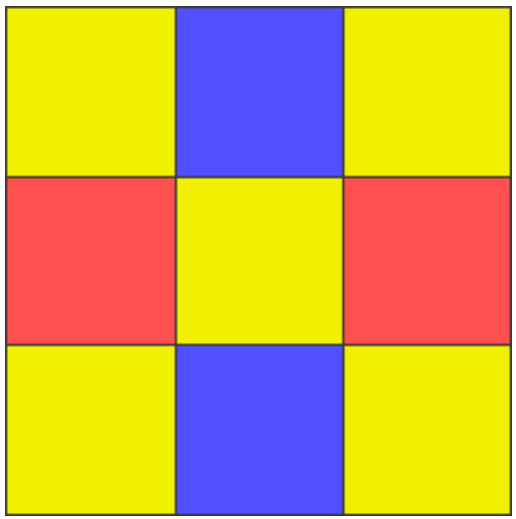
- 한 변의 길이가 1 또는 2라면 간단하게 풀 수 있습니다.
- 3×3 크기는 손으로 풀기에는 조금 어렵습니다.
  - 하지만 우리 앞에는 컴퓨터가 있습니다!
  - 총합이 3의 배수인 3×3 배치는 총 $3^8 = 6,561$개입니다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 완전 탐색을 돌려 보면...

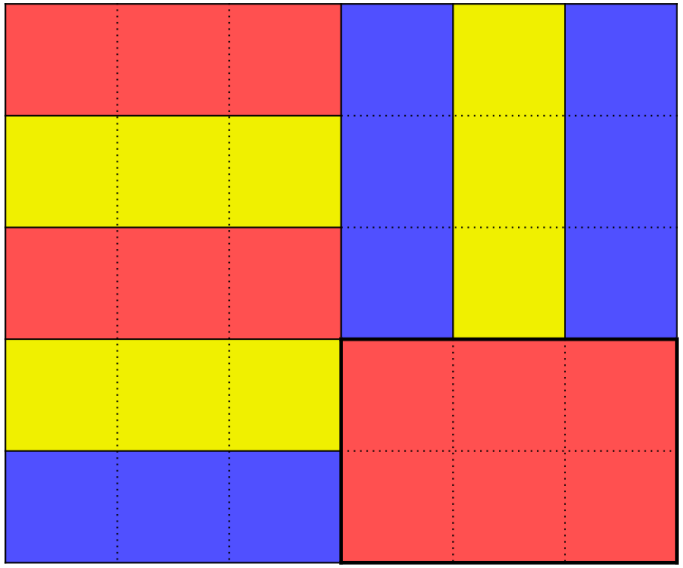
Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 완전 탐색을 돌려 보면...
- 모든 배치에 대해 8번 이하의 합체로 한 가지 색으로 만드는 방법이 존재함을 발견할 수 있습니다.



- 위의 배치를 8번만에 하나의 색으로 합쳐 보세요 :D

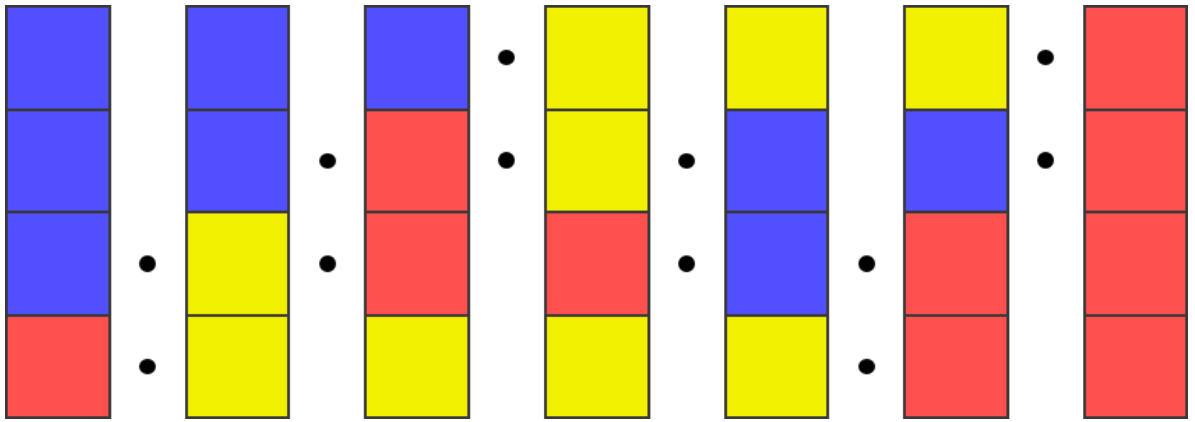
Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여



- 작은 문제를 풀고 나면 위와 같은 상태가 됩니다.
- 이제 모든 슬라임을 작은 문제의 결과로 나온 색으로 바꾸면 문제가 풀립니다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 처음에 세 마리씩 합쳐 두었던 슬라임들을 역순으로 원하는 색으로 바꿔 나갑니다.
  - 역시 6번의 합체로 세 마리를 한꺼번에 원하는 색으로 바꿀 수 있습니다.



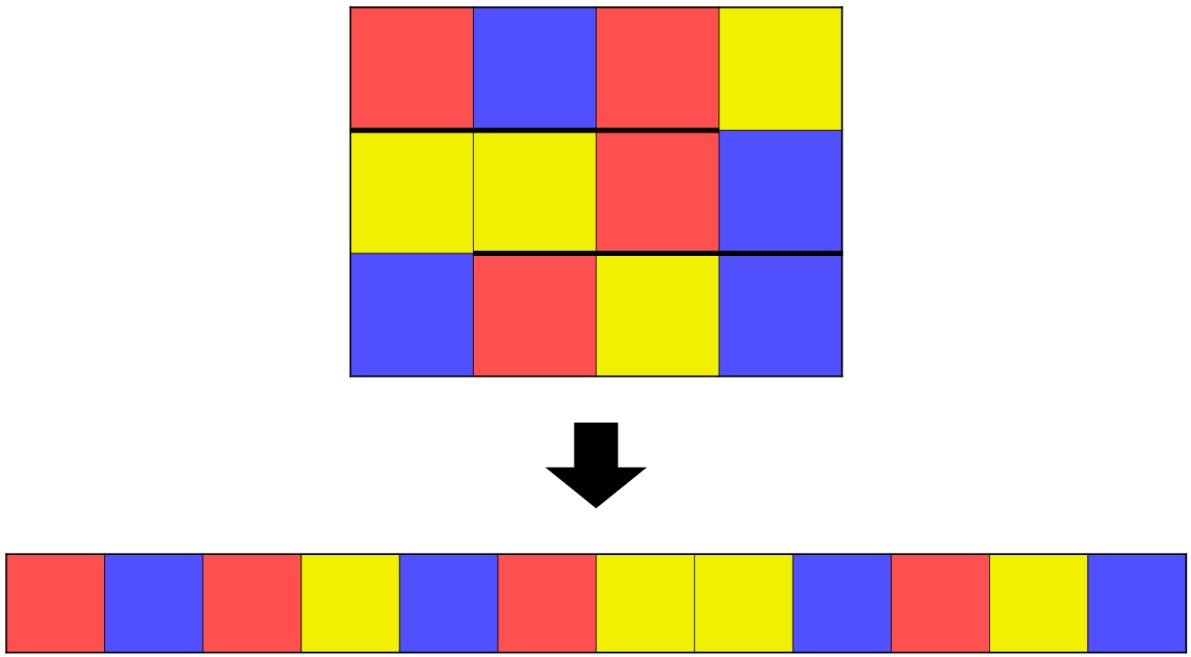
- 이제 모든 슬라임이 하나의 색이 되었습니다!

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 여기까지가 출제자의 풀이였으나, 다른 분들이 더 깔끔한 접근으로 문제를 해결하셨기에 소개합니다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 여기까지가 출제자의 풀이였으나, 다른 분들이 더 깔끔한 접근으로 문제를 해결하셨기에 소개합니다.
- 핵심 아이디어는 배치를 일렬로 펼치는 것입니다.

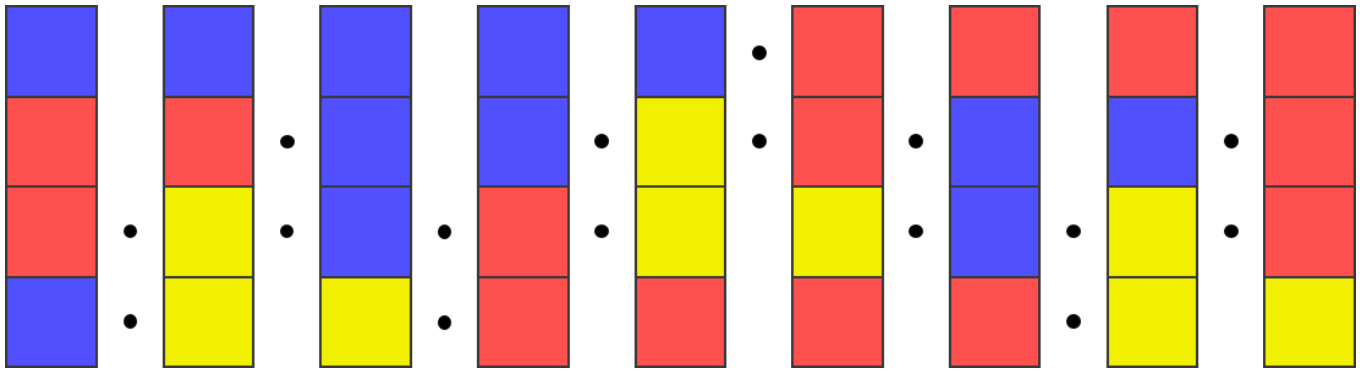


Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 일렬로 펼친 뒤 앞에서와 같은 풀이를 적용하면 1×3 이하 크기의 작은 문제가 남습니다.
- 이는 아주 쉽게 손으로 풀 수 있습니다.
- 따라서 앞의 풀이에서 가장 번거로운 부분이었던 3×3 완전 탐색 없이도 간단하게 문제를 풀 수 있습니다.

Div1A. 전생했더니 슬라임 연구자가 아니었던 건에 대하여

- 이 풀이에서 필요한 합체 횟수는 최대 $4NM$번입니다.
  - 3마리의 슬라임을 합치는 데 최대 12번의 합체가 필요합니다.
- 조금 더 삽질 생각하면 $\frac{8}{3}NM$까지도 개선할 수 있습니다.



Div2J/Div1B. 라이언 동상 구하기

- 제출 51회, 정답 17명 (17명 시도, 정답률 33.33%)
- 처음 푼 참가자: 강태규 (19분)
- 출제자: kazel

Div2H/Div1C. 성냥팔이 소년

- 제출 87회, 정답 17명 (20명 시도, 정답률 19.54%)
- 처음 푼 참가자: 박성관 (23분)
- 출제자: kdh9949

Div1D. 꽃집

- 제출 9회, 정답 0명 (1명 시도, 정답률 0.00%)
- 처음 푼 참가자: 없음
- 출제자: Namnamseo

Div1D. 꽃집

- 분야: 고급 DP 및 convex hull trick의 응용
- 문제를 수식으로 모델링해 봅니다.
- 주어진 수열의 prefix sum 수열을 p_i라고 합시다.
- 즉, $p_i = \sum_{j=1}^{i} v_j$.

Div1D. 꽃집

이제 문제를 다음과 같이 바꾸어 말할 수 있습니다.

문제 모델링

길이 N의 수열을 K개 이하의 구간으로 분할하는 최소 비용을 구하시오.

단, $j+1$번째부터 i번째 원소로 이루어진 구간의 비용은 $(i-j)(p_i - p_j)$이며, 전체의 비용은 각 구간의 비용의 합이다.

Div1D. 꽃집

- 다음과 같은 dynamic programming (DP) 해법이 있습니다.
- $D_{c,i}$를, i번째 원소까지의 수열을 c개의 구간으로 분할하는 최소 비용이라고 하면,

$$D_{c,i} = \min_j (D_{c-1,j} + (i-j) \cdot (p_i - p_j)).$$

- 모든 $1 \le c \le K$, $1 \le i \le N$에 대해 이 값을 구합니다.
- 최솟값이 나오는 j를 찾기 위해서 $\mathcal{O}(N)$ 회의 탐색이 필요하므로 총 시간복잡도는 $\mathcal{O}(N^2K)$입니다.

Div1D. 꽃집

- 이제 K에 관한 항을 줄이는 방법을 설명합니다.
- 다음과 같은 DP를 생각합니다.
- D_i를, i번째 원소까지의 수열을, 구간 개수에 상관 없이 분할하는 최소 비용이라고 하면,

$$D_i = \min_j(D_j + (i-j)\cdot(p_i - p_j)).$$

Div1D. 꽃집

물론 이 풀이로는 문제를 해결할 수 없습니다.
분할 방법이 K개 이하라는 조건을 만족한다는 보장이 없기 때문입니다.

이대로라면, 구간을 잘게 쪼갤수록 비용이 감소하므로,
N개의 **구간**을 쓰게 됩니다.
그렇다면, 구간을 너무 많이 쓰지 않게 해봅시다.

Div1D. 꽃집

- 하나의 구간을 쓸 때마다 **패널티**를 적용한다고 생각합시다.
- D_i를, i번째 원소까지의 수열을, 구간 개수에 상관 없이 분할하는 최소 비용 (패널티 포함)이라고 하면,

$$D_i = \min_j (D_j + (i-j) \cdot (p_i - p_j) + C).$$

- 전에 없던 C라는 항을 추가해 봤습니다.

Div1D. 꽃집

예를 들어 t번 분할하는 방법이 있다면,
이제 그 비용은 원래보다 $+tC$만큼 크게 나타납니다.

만약에 C가 아주 크다면, 구간을 여러 개 사용하는 것이 큰 부담이 되기 때문에, 전체를 하나의 구간으로 묶게 될 것입니다.

이제 C를 천천히 줄여나가면, 최종적으로 D_N이 사용하는 분할 개수는 점차 늘어날 것입니다.
$C = 0$이 되면 N개를 사용하게 될 테구요.

Div1D. 꽃집

또한 D_N이 예를 들어 t개의 구간을 사용한다면, D_N의 분할방법은 t개의 구간을 사용하는 분할방법 중에서도 최저 비용 방법입니다. D_N은 **모든** 분할 방법 중 최적이므로, 패널티의 영향이 동일한 방법이 여럿이라면 그중에서도 최소인 쪽을 사용했을 테니까요. 다시 말해, 처음 식의 D를 가져오면 $D_N = D_{t,N} + tC$인 정수 t가 존재합니다.

Div1D. 꽃집

- 그런 t가 정확히 K가 되는 적절한 패널티 값 C가 있다면...
- 그때의 D_N을 구할 수만 있다면, $D_{K,N} = (D_N - KC)$이므로 답을 알 수 있습니다!
- 여기서 두 가지 문제가 있습니다.
  - 적당한 패널티 값 C가 항상 있을까요? 있다면 어떻게 찾나요?
  - 그 때의 D_N을 $\mathcal{O}(N^2)$ 미만으로 어떻게 구할까요?

Div1D. 꽃집

첫 번째 질문부터 살펴봅니다. 결론부터 말하자면,

정리 1. 패널티값 C의 존재

1 이상 N 이하의 임의의 정수 t에 대해,

t개를 분할하는 방법이 최소가 되도록 하는 정수 패널티 값 C가 존재한다.

Div1D. 꽃집

- 증명을 위해서 먼저 다음 사실을 생각합니다.
- 구간의 수가 많아질수록 비용은 감소합니다. 즉, $D_{c,N} \geq D_{c+1,N}$.
- 그런데 실은 c가 커짐에 따라 이 값 자체도 감소합니다.

보조정리 1. $D_{c,N}$의 볼록성

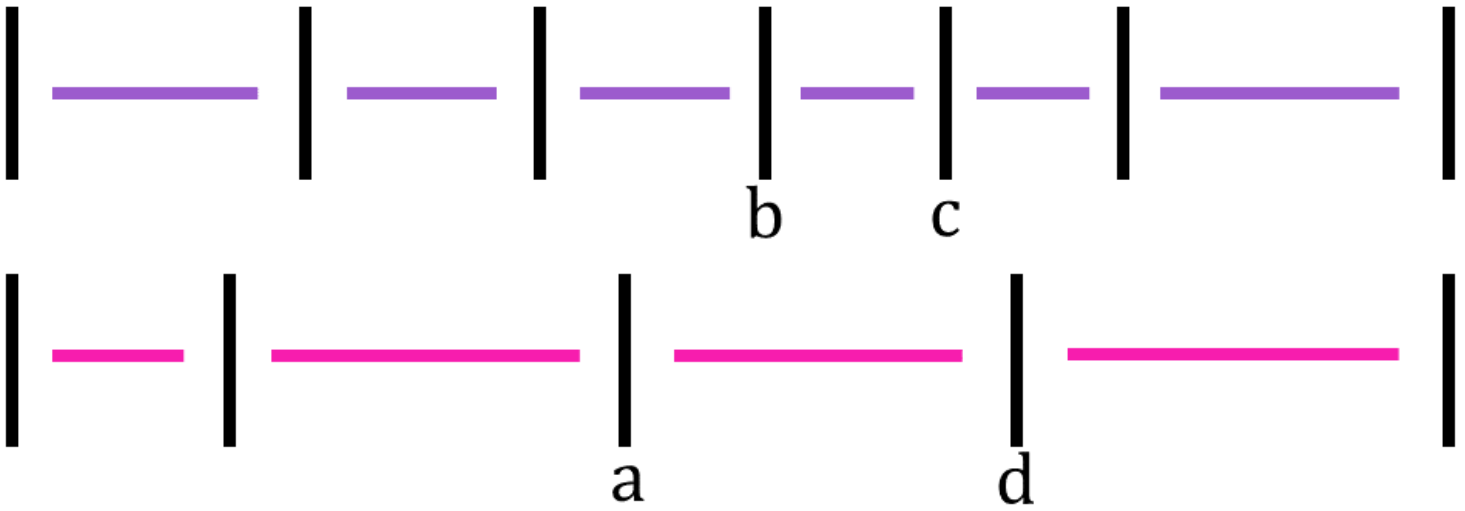
$D_{c,N} - D_{c+1,N} \geq D_{c+1,N} - D_{c+2,N} \geq 0$.

Div1D. 꽃집

왜일까요? 먼저 위의 식을 정리합니다.

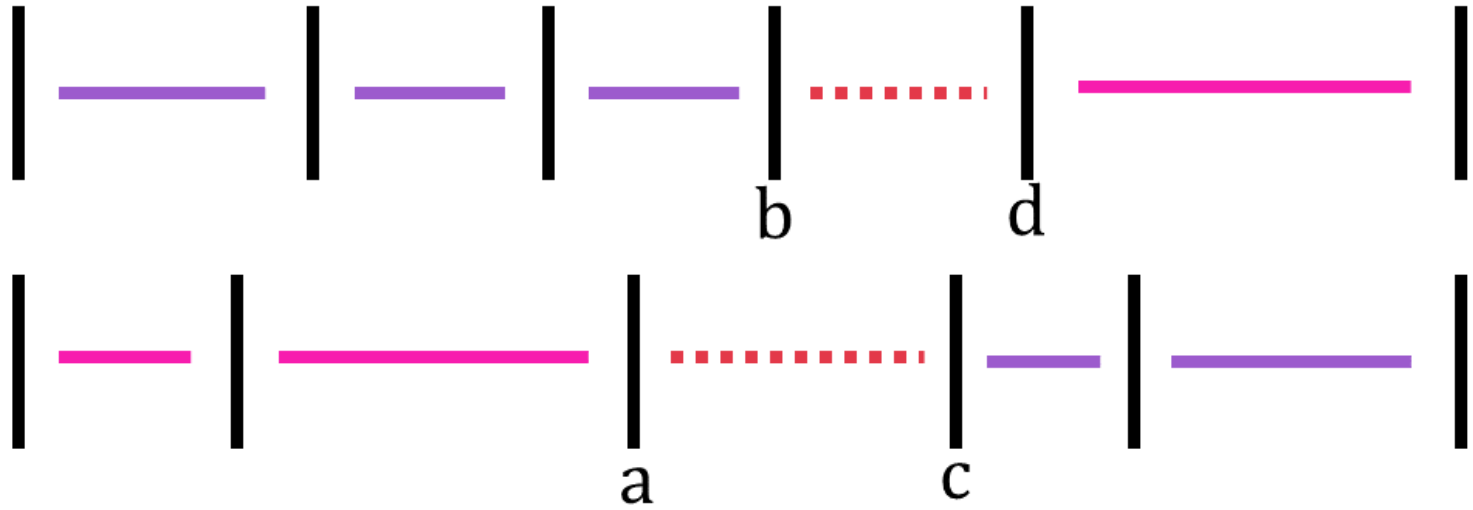
$$D_{c+2,N} + D_{c,N} \geq 2 \times D_{c+1,N}$$

$(c+2)$개를 사용하는 방법과 c개를 사용하는 방법을 나란히 놓아봅니다.



Div1D. 꽃집

이 둘을 아래와 같이 재배열해 봅니다.



비둘기집의 원리에 의해, 그림의 *a*-*b*-*c*-*d*와 같은 모양의 구간이 항상 존재합니다.

Div1D. 꽃집

- $cost(i, j) = (j - i)(p_j - p_i)$라고 쓰겠습니다.
- 첫 번째 방법은 $cost(a, d) + cost(b, c)$가,
- 두 번째 방법은 $cost(a, c) + cost(b, d)$가 듭니다.
- 식을 전개해서 정리해 보면, 전자가 후자보다 크거나 같음을 쉽게 알 수 있습니다.

Div1D. 꽃집

- 따라서 도움정리 1이 참입니다.
- 정리 1의 증명은 도움정리 1을 이용해 쉽게 보일 수 있습니다.
- 이 아이디어는 IOI 2016 Day 2의 3번 문제(Aliens)에서 쓰인 것으로도 유명합니다.

Div1D. 꽃집

- 이제 D_N을 $\mathcal{O}(N^2)$ 미만으로 구하는 방법에 대해 설명합니다.
- 점화식을 다시 살펴봅니다.

$$D_i = \min_j(D_j + (i-j)(p_i - p_j)) + C \tag{1}$$

$$= \min_j((D_j + j \cdot p_j) - (i \cdot p_j + j \cdot p_i)) + (i \cdot p_i + C). \tag{2}$$

Div1D. 꽃집

- 여기서부터는 "convex hull trick"이라는 별칭의 기법에 대한 지식을 가정합니다.
- 먼저 $i \cdot p_j$ 항을 무시하고 생각해봅니다.

$$D_i = \min_j(-j \cdot p_i + (D_j + j \cdot p_j)) + (i \cdot p_i + C).$$

- min 안쪽의 식은 기울기 $-j$, y절편 $D_j + j \cdot p_j$인 일차함수의, $x = p_i$에서의 함수값입니다.

Div1D. 꽃집

- 직선이 추가될 때, 기울기가 감소하는 순으로 추가됩니다.
- 즉 여러 번의 pop back 후 한 번의 push back으로 해결 가능합니다.
- 또한 묻는 x좌표 역시 증가하므로, 여러 번의 pop front로 해결이 가능합니다.
- 즉 여기까지는 $\mathcal{O}(N)$에 가능합니다.
- 이제, $i \cdot p_j$ 항에 대해 생각해봅시다.

Div1D. 꽃집

$-i \cdot p_j$ 항에서 i 부분의 값을 조금씩 키우는 상상을 합니다.

- 직선들이 아래로 떨어집니다...
- 그런데 어떤 직선은 다른 것보다 더 빨리 떨어집니다.

- 직선들의 교점 역시 (왼쪽으로) 움직입니다...
- 그런데 어떤 교점은 다른 교점보다 더 빨리 움직입니다!

Div1D. 꽃집

- 직선의 교점의 이동 속도를 계산해 봅니다.
- convex hull 상에서 인접한 세 직선 사이의, 두 교점을 생각합니다.
- i의 값이 커짐에 따라 이 두 교점은 만납니다.
- 그때 가운데 직선은 hull에서 빠지게 됩니다.

Div1D. 꽃집

- 거꾸로, 두 직선 L_a, L_b 뒤에 L_c를 추가하면?
- L_b가 빠지게 되는 $i_{b,exit}$의 값을 역산할 수 있습니다.
- 물론 이 $i_{b,exit}$는 경우에 따라 존재하지 않을 수 있습니다.

Div1D. 꽃집

- D_i를 구할 때, $i = i_{b,exit}$인 모든 직선을 일단 제거할 수 있습니다.
- 한 직선이 제거되면 그 양쪽의 두 직선이 인접하게 됩니다.
- 즉 linked list에서의 삭제로 처리하면 됩니다.

Div1D. 꽃집

- 총 시간복잡도는 어떻게 될까요?
- 직선의 추가: 정확히 N회.
- 직선의 삭제: 최대 N회.
- 교점의 계산: 최대 N회입니다. 직선이 추가될 때마다 최대 한 번만 계산하기 때문입니다.
- 따라서 $\mathcal{O}(N)$에 D_N을 구할 수 있습니다.

Div1D. 꽃집

- 패널티 값의 범위는 0 이상 $n \cdot p_n$ 이하입니다.
- "D_N이 사용하는 구간의 최소 개수"가 K 이하인 최소의 패널티 값을 이분탐색으로 찾을 수 있습니다.
- 따라서 총 시간복잡도 $\mathcal{O}(N \cdot \log(N \cdot v_{max}))$에 문제를 해결할 수 있습니다

Div2E/Div1E. 순위 계산

- 제출 45회, 정답 21명 (21명 시도, 정답률 46.67%)
- 처음 푼 참가자: 김현수 (22분)
- 출제자: bryan

Div1F. 코포빵 토너먼트

- 제출 16회, 정답 10명 (10명 시도, 정답률 62.50%)
- 처음 푼 참가자: 최석환 (37분)
- 출제자: kdh9949

Div1F. 코포빵 토너먼트

- 우선, 대진표를 트리로 나타내 봅시다.
- 리프 노드는 참가자, 내부 노드는 각 경기에 해당합니다.
- 각 경기, 즉 각 내부 노드마다 수를 하나씩 적을 텐데, 모든 내부 노드에 적히는 숫자 개수의 합의 기댓값을 구하면 되겠습니다.

Div1F. 코포빵 토너먼트

- **기댓값의 선형성**에 대해 아시나요?
- **합의 기댓값**은 **기댓값의 합**과 같습니다.
- 각 내부 노드에 적힐 숫자 개수의 기댓값을 각각 구해서 다 더하면 된다는 뜻입니다!

Div1F. 코포빵 토너먼트

- 각 내부 노드에 적히는 수가 무엇일까요?
- 그 노드를 루트로 하는 서브트리에 적힌 수들 중 최댓값입니다.
- 참가자들이 처음에 임의의 순서로 서기 때문에, 오직 서브트리의 **리프 노드의 수**만이 그 내부 노드에 적히는 수에 영향을 줌을 알 수 있습니다.

Div1F. 코포빵 토너먼트

- 서브트리에 리프 노드를 k개 가진 내부 노드에 적힐 숫자의 기댓값을 $f(k)$라 하고, $f(k)$의 값을 구해 봅시다.
- P_i를 i자리 이상의 수가 적힐 확률이라고 정의하면, $f(k) = \sum_{i=1}^{7} P_i$ 임을 알 수 있습니다.
- 약간 이상하게 보일 수 있는 식인데, 잘 생각해 보시면 맞는 식입니다.

Div1F. 코포빵 토너먼트

- C_i : 참가자들 중 레이팅이 i자리 이상인 사람들의 수라 정의합시다.
- 나한테 적히는 수가 i자리 미만의 수이려면, i자리 이상의 수들이 내 서브트리에 적히면 안 됩니다.
- 즉, $1 - P_i =$ (내 밑에 i자리 이상 수들이 하나도 안 적히는 확률) 입니다.
- n을 총 사람 수라고 하면 $\frac{\binom{n-k}{C_i}}{\binom{n}{C_i}}$ 으로 쓸 수 있습니다.

Div1F. 코포빵 토너먼트

- 이항 계수 $(\mathrm{mod}\ 10^9 + 7)$ 의 계산은 최대 레이팅 범위를 N 이라 하면 $\mathcal{O}(N)$ 전처리를 미리 해 두면 한 번 계산할 때 $\mathcal{O}(1)$ 의 연산량이 듭니다.
- 즉, $\mathcal{O}(\log N)$ 번의 이항 계수 계산으로 $f(k)$를 계산할 수 있습니다.

Div1F. 코포빵 토너먼트

- 물론 모든 내부 노드에 대해서 $f(k)$를 계산해서 더하면 시간 초과를 받습니다.
- 쿼리 한 번당, $f(k)$에 들어가는 서로 다른 k의 개수는 몇 개일까요?
- 놀랍게도, $\mathcal{O}(\log N)$개 입니다.

Div1F. 코포빵 토너먼트

- 코포빵 대결의 매 iteration을 따라가면서, 각 노드의 서브 트리가 가지는 리프 개수를 쭉 나열해 봅시다.
- 경기가 치루어지는 것은 노드 두 개를 없애고, 두 노드의 리프 개수를 더한 새 노드 하나를 만드는 것과 같습니다.
  - Ex) $n = 6$ (참가자가 6명)
  - $1\ 1\ 1\ 1\ 1\ 1 \to 2\ 2\ 2 \to 4\ 2 \to 6$

Div1F. 코포빵 토너먼트

- 그런데 같은 숫자가 엄청 많이 나오는 것 같으니까, RLE(Run Length Encoding)를 해 봅시다.
  - Ex) $n = 6$ (앞의 예시)
  - $(1\ \ 6) \to (2\ \ 3) \to (4\ \ 1)\ \ (2\ \ 1) \to (6\ \ 1)$
- 뭔가 각 iteration마다 덩어리가 1개 아니면 2개인 거 같습니다... 그리고 그게 맞습니다!

Div1F. 코포빵 토너먼트

- 처음 상태는 무조건 `(1 n)` 입니다.
- 가능한 상태가 총 4개 있는데, 이들 안에서 무조건 왔다갔다 함을 보일 수 있습니다.
  - `(x 2k)` $\rightarrow$ `(2x k)`
  - `(x 2k+1)` $\rightarrow$ `(2x k) (x 1)`
  - `(x 2k) (y 1)` $\rightarrow$ `(2x k) (y 1)`
  - `(x 2k+1) (y 1)` $\rightarrow$ `(2x k) (x+y 1)`
- iteration의 횟수는 $\mathcal{O}(\log N)$번이니, 서로 다른 k의 개수도 $\mathcal{O}(\log N)$개 입니다!

Div1F. 코포빵 토너먼트

- 따라서, 저 덩어리들을 매 iteration마다 관리해 주면서 새로 생기는 노드들에 대해 $f(k)$ 값을 구한 뒤 그 개수를 곱해서 답에 더해주면 됩니다.
- 다음 층으로 넘어갈 때 새로 생기지 않고 그냥 넘어오는 노드도 존재하니 구현 시 주의해야 합니다.
- 총 시간복잡도는 $\mathcal{O}(N + Q \log^2 N)$.
- 시간제한을 정말 널널하게 주었으니 (제 코드가 7xx ms에 돕니다) 아무렇게나 짜도 시간초과는 안 날 겁니다.

Div1G. 파리채 만들기

- 제출 1회, 정답 1명 (1명 시도, 정답률 100.00%)
- 처음 푼 참가자: 최석환 (223분)
- 출제자: 16silver

Div1G. 파리채 만들기

- 다각형을 삼각형들로 쪼개서 문제를 풀어보도록 합시다!
- 기준점을 고정하기 위해 다각형을 삼각형 $P_1P_iP_{i+1}$들로 분할합니다.
- 그리고 P_1이 원점이 되도록 평행이동합니다.
- 삼각형 ABC 내부의 임의의 점은 어떻게 나타날까요?
- $A + a\overrightarrow{AB} + b\overrightarrow{AC} (a, b \geq 0, a + b \leq 1)$

Div1G. 파리채 만들기

- 두 삼각형 $P_1P_iP_{i+1}, P_1P_jP_{j+1}$에서 각각 하나씩 뽑은 두 점 $P_1 + a\vec{u_1} + b\vec{v_1}, P_1 + c\vec{u_2} + d\vec{v_2}$ 사이의 거리의 제곱은

$$|a\vec{u_1} + b\vec{v_1} - c\vec{u_2} - d\vec{v_2}|^2$$

$$= (a\vec{u_1} + b\vec{v_1} - c\vec{u_2} - d\vec{v_2}) \cdot (a\vec{u_1} + b\vec{v_1} - c\vec{u_2} - d\vec{v_2})$$

- 이제 a, b, c, d 중 2개의 곱들의 기댓값을 구해봅시다!

Div1G. 파리채 만들기

$$E(f(a,b)) = \frac{\int_0^1 \int_0^{1-a} f(a,b)\,db\,da}{\int_0^1 \int_0^{1-a} db\,da} = 2\int_0^1 \int_0^{1-a} f(a,b)\,db\,da$$

$$E(a^2) = 2\int_0^1 \int_0^{1-a} a^2\,db\,da = 2\int_0^1 a^2(1-a)da = \frac{1}{6}$$

$$E(b^2) = E(c^2) = E(d^2) = \frac{1}{6}, \quad E(ab) = E(cd) = \frac{1}{12}$$

$$E(ac) = E(a)E(c) = \frac{1}{9}$$

$$= E(ad) = E(bc) = E(bd)$$

Div1G. 파리채 만들기

- $\vec{v_i} = \overrightarrow{P_1P_i} = (x_i, y_i)$라 놓습니다.
- 두 삼각형 $P_1P_iP_{i+1}, P_1P_jP_{j+1}$에서 하나씩 점을 뽑았을 때 두 점 사이의 거리의 기댓값은

$$\frac{1}{6}\left(|\vec{v_i}|^2 + |\overrightarrow{v_{i+1}}|^2 + |\vec{v_j}|^2 + |\overrightarrow{v_{j+1}}|^2\right) + \frac{1}{6}\left(\vec{v_i}\cdot\overrightarrow{v_{i+1}} + \vec{v_j}\cdot\overrightarrow{v_{j+1}}\right)$$

$$-\frac{2}{9}\left(\vec{v_i} + \overrightarrow{v_{i+1}}\right)\cdot\left(\vec{v_j} + \overrightarrow{v_{j+1}}\right)$$

- 이제 이 식을 $i, j = 2, 3, \cdots, n-1$에 대해 더해주면 되는데...
- 잠깐! 가중치를 곱해줘야 합니다!

Div1G. 파리채 만들기

- 가중치는 두 삼각형의 넓이의 곱이 됩니다.
- 그런데 생각해보니 오목다각형의 경우에는 삼각형 $P_1P_iP_{i+1}$들이 모두 다각형의 내부가 된다는 보장이 없습니다.
- 괜찮을까요?
- **괜찮습니다!!!** 단, 빼 주어야 하는 삼각형은 가중치가 음수가 되도록 하면 됩니다.
- 삼각형 $P_1P_iP_{i+1}$의 넓이는 $|x_iy_{i+1} - x_{i+1}y_i|$인데, 여기서 절댓값 안에 있는 식이 정확히 우리가 구하려는 가중치입니다.

Div1G. 파리채 만들기

- 이제 다음 식을 계산해주면 됩니다!

$$\sum_{i=2}^{n-1}\sum_{j=2}^{n-1}(x_iy_{i+1}-x_{i+1}y_i)(x_jy_{j+1}-x_{j+1}y_j)\left[\frac{1}{6}\left(x_i^2+x_{i+1}^2+y_i^2+y_{i+1}^2\right.\right.$$
$$\left.+x_j^2+x_{j+1}^2+y_j^2+y_{j+1}^2+x_ix_{i+1}+y_iy_{i+1}+x_jx_{j+1}+y_jy_{j+1}\right)$$
$$\left.-\frac{2}{9}\left((x_i+x_{i+1})(x_j+x_{j+1})+(y_i+y_{i+1})(y_j+y_{j+1})\right)\right]$$

- 변수 분리 등을 통해 계산량을 $O(n)$으로 줄일 수 있습니다!

Div1G. 파리채 만들기

- 다음 네 개의 합을 이용하여 식을 단순화합니다.

$$
\begin{aligned}
S &= \sum_{i=2}^{n-1}(x_i y_{i+1} - x_{i+1} y_i) \\
S_1 &= \sum_{i=2}^{n-1}(x_i y_{i+1} - x_{i+1} y_i)(x_i^2 + x_i x_{i+1} + x_{i+1}^2 + y_i^2 + y_i y_{i+1} + y_{i+1}^2) \\
S_2 &= \sum_{i=2}^{n-1}(x_i y_{i+1} - x_{i+1} y_i)(x_i + x_{i+1}) \\
S_3 &= \sum_{i=2}^{n-1}(x_i y_{i+1} - x_{i+1} y_i)(y_i + y_{i+1})
\end{aligned}
$$

Div1G. 파리채 만들기

$$
\begin{aligned}
(eqn) &= \sum_{i=2}^{n-1}(x_i y_{i+1} - x_{i+1} y_i)\left[\frac{1}{6}S\left(x_i^2 + x_i x_{i+1} + x_{i+1}^2 + y_i^2 + y_i y_{i+1}\right.\right. \\
&\quad \left.\left. + y_{i+1}^2\right) + \frac{1}{6}S_1 - \frac{2}{9}\left((x_i + x_{i+1})S_2 + (y_i + y_{i+1})S_3\right)\right] \\
&= \frac{1}{3}SS_1 - \frac{2}{9}\left(S_2^2 + S_3^2\right)
\end{aligned}
$$

- 이 식을 S^2으로 나눈 값이 최종 답입니다!

Div1G. 파리채 만들기

다른 풀이

- 주어진 다각형 D 내의 두 점 $(x_1, y_1), (x_2, y_2)$에 대해 거리는 $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$입니다.
- 이 값의 제곱인 $(x_1 - x_2)^2 + (y_1 - y_2)^2$의 기댓값을 구해야 합니다.
- 점 하나당 이중적분 한 번, 총 두 번의 이중적분을 합니다.
- 이 값을 다각형의 넓이의 제곱 S^2으로 나누면 됩니다.

Div1G. 파리채 만들기

$$\frac{1}{S^2}\iint_D\iint_D (x_1-x_2)^2+(y_1-y_2)^2\,dx1\,dy1\,dx2\,dy2$$

- 그런데 대체 이 식을 어떻게 계산해야 할까요?
- 일단 전개하고 나누어서 단순화합니다.

Div1G. 파리채 만들기

$$
\begin{aligned}
& \frac{1}{S^2} \iint_D \iint_D (x_1 - x_2)^2 + (y_1 - y_2)^2 \, dx_1 \, dy_1 \, dx_2 \, dy_2 \\
= & \frac{1}{S^2} \iint_D \iint_D (x_1^2 + x_2^2 - 2x_1x_2 + y_1^2 + y_2^2 - 2y_1y_2) \, dx_1 \, dy_1 \, dx_2 \, dy_2 \\
= & \frac{1}{S^2} \left(2S \iint_D x^2 + 2S \iint_D y^2 - 2 \left(\iint_D x \right)^2 - 2 \left(\iint_D y \right)^2 \right)
\end{aligned}
$$

- 이제 $\iint_D x^2, \iint_D x$ 등을 구하면 됩니다.
- How?

Div1G. 파리채 만들기

- **그린 정리**를 이용하면 됩니다!
- 그린 정리(의 쓰기 좋은 한 형태)

$$\int_{\partial D} P(x,y)\,dx + Q(x,y)\,dy = \iint_D \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right) dx\,dy$$

- $P(x,y) = 0, Q(x,y) = \frac{1}{3}x^3$을 대입하면

$$\iint_D x^2\,dx\,dy = \int_{\partial D} \frac{1}{3}x^3\,dy$$

- 오른쪽 적분은 각 변에 해당하는 선분을 매개변수화하여 단순 적분으로 바꾸어 계산한 뒤 더하면 됩니다.

Div1G. 파리채 만들기

- 계산 결과는 다음과 같습니다.

$$\iint_D x^2 = \frac{1}{12}\sum_{i=1}^{n}(x_{i+1}^3 + x_{i+1}^2 x_i + x_{i+1}x_i^2 + x_i^3)(y_{i+1} - y_i)$$

$$\iint_D y^2 = -\frac{1}{12}\sum_{i=1}^{n}(y_{i+1}^3 + y_{i+1}^2 y_i + y_{i+1}y_i^2 + y_i^3)(x_{i+1} - x_i)$$

$$\iint_D x = \frac{1}{6}\sum_{i=1}^{n}(x_{i+1}^2 + x_{i+1}x_i + x_i^2)(y_{i+1} - y_i)$$

$$\iint_D y = -\frac{1}{6}\sum_{i=1}^{n}(y_{i+1}^2 + y_{i+1}y_i + y_i^2)(x_{i+1} - x_i)$$

- 이걸 원래 식에 넣으면 답이 나옵니다.

Div1G. 파리채 만들기

- 사실 이렇게 구한 값들이 삼각형 풀이에서 S, S_1, S_2, S_3와 같음을 확인할 수 있습니다!
- 그래서 가능한 좌표의 최댓값 $M = 10000$에 대해 $S_1 = \iint_D x^2 + \iint_D y^2 \leq M^2 \times (4M^2) \times 2 = 8M^4$이 되어 C++의 long long 범위를 넘지 않음이 보장됩니다.
- SS_1 등의 값은 long long 범위를 넘어갈 수 있으므로 double로 변환한 뒤 계산합시다.

Div1H. 삼분 그래프

- 제출 2회, 정답 2명 (2명 시도, 정답률 100.00%)
- 처음 푼 참가자: 강태규 (154분)
- 출제자: kdh9949

Div1H. 삼분 그래프

- 평면 그래프의 연결 성분 개수와 관련된 유명한 식이 있습니다.

Euler's Formula on Planar Graph

- $V - E + F = C + 1$
- V : 정점의 수, E : 간선의 수, F : 간선들로 둘러싸인 영역의 수 (가장 바깥쪽의 unbounded area 포함), C : 연결 성분의 수

- 아실 분들은 다 아실 겁니다..

Div1H. 삼분 그래프

- 그래프를 수직선 2개로 잘랐을 때 연결 성분 개수의 변화 ΔC는 $\Delta V - \Delta E + \Delta F$가 됩니다.
- 문제 조건에 의해 그래프를 자르기 전에는 $C = 1$임이 보장되므로 쿼리마다 $\Delta C + 1$을 구해서 출력하면 됩니다.
- $x = A$, $x = B$를 각각 직선 A, B라고 하고, 이 두 직선으로 그래프를 자를 때 ΔV, ΔE, ΔF를 각각 구해 봅시다.

Div1H. 삼분 그래프

- $\Delta V = 2\times$ (직선 A를 지나는 간선 개수 + 직선 B를 지나는 간선 개수)
- $\Delta E =$ (직선 A를 지나는 간선 개수 + 직선 B를 지나는 간선 개수)
- $\Delta F = -$(직선 A를 지나거나 직선 B를 지나는 Face 개수)

Div1H. 삼분 그래프

- ΔV와 ΔE는 쉽게 구할 수 있습니다.
- 정점들의 x좌표를 좌표압축 해준 뒤, 누적합 배열을 사용하면 전처리 $O(NlogN + M)$, 쿼리당 $O(1)$에 처리 가능합니다.
- Binary Indexed Tree 같은 자료구조로 log를 하나 더 붙여서 해도 됩니다.

Div1H. 삼분 그래프

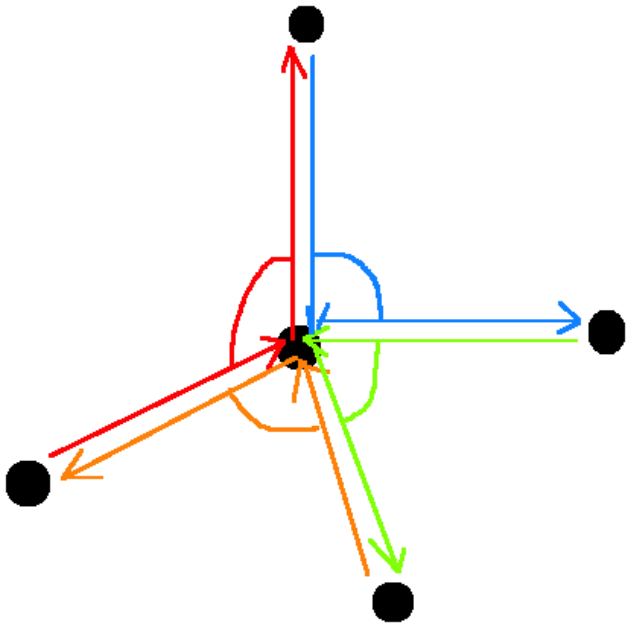
- ΔF를 구하려면 2가지 작업을 해야 합니다.
  - 각 Face가 차지하는 x좌표의 구간 구하기
  - 각 (A,B) 순서쌍이 없애는 Face의 개수 빠르게 구하기
- 사실 둘 다 '웰노운'이라 불리는 문제들입니다.
- 아는 사람들은 다 안다는 소리입니다.
- 하지만 모르시는 분들을 위해 설명을 해야 합니다.

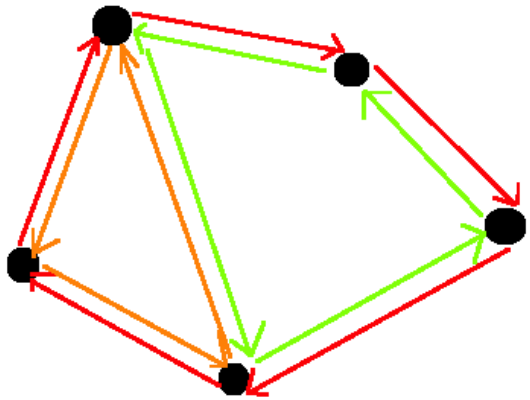
Div1H. 삼분 그래프

- 평면 그래프에서 Face를 따는 방법에는 여러 가지가 있습니다. 그 중 한 가지를 소개하겠습니다.
- 각 간선을 두 개의 방향으로 쪼개 줍니다. 각 정점에서는 그 정점과 연결된 모든 간선에 대해 (나한테 들어오는 간선 번호, 나에서 나가는 간선 번호, 반대편 정점의 좌표)의 tuple로 정보를 저장합니다.
- 각 정점별로 연결된 간선들을 각도 순으로 정렬해준 뒤, 인접해 있는 간선 쌍들에 대해 들어오는 간선과 나가는 간선을 Union-Find로 합쳐 줍니다.

Div1H. 삼분 그래프

- 뭔 말인지 모르시겠다면 아래 그림을 참고해 주세요.
- 손으로 그린 거 맞습니다.





Div1H. 삼분 그래프

- 각 Face에 대해 그 Face가 차지하는 x좌표의 범위는 Union-Find 과정에서 같이 구해줄 수 있습니다.
- 모든 정점에 대해 작업을 수행하고 나면, 그래프 내부에 있는 Face들과 함께 그래프를 한 바퀴 둘러싸는 Face가 같이 구해집니다.
- 그 영역은 바깥쪽의 Unbounded area이므로 ΔF를 구할 때 무시해 주어야 합니다.

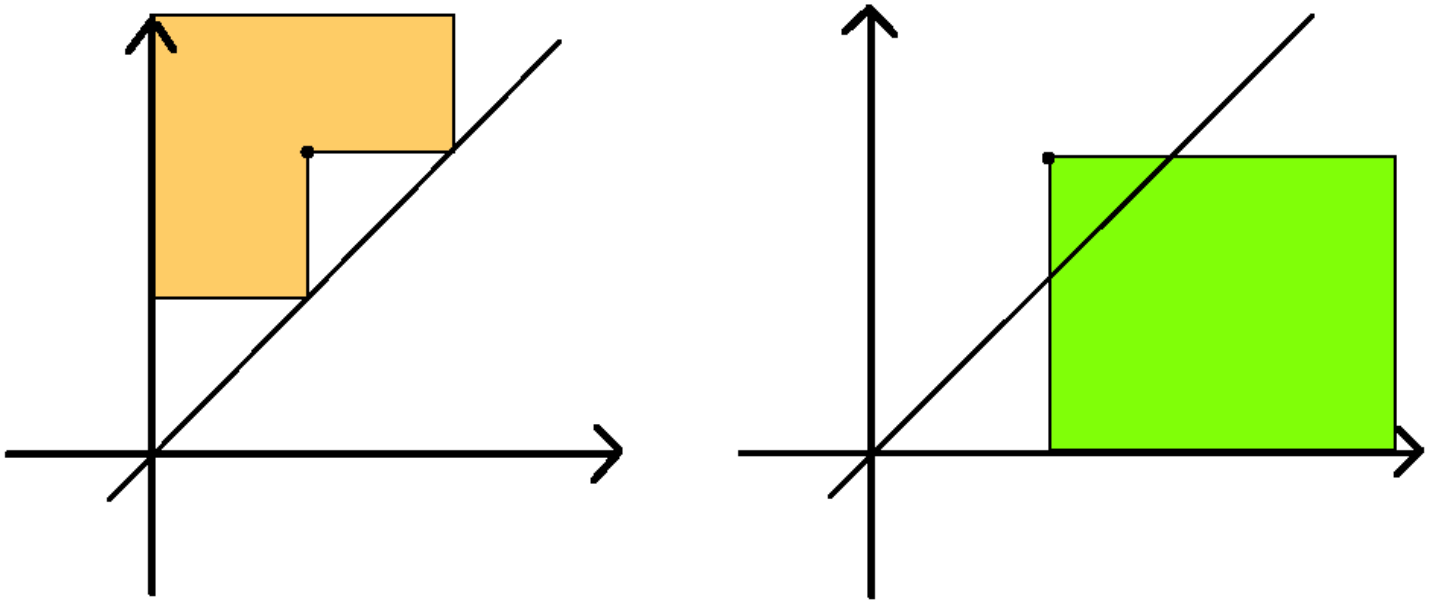
Div1H. 삼분 그래프

- 어떤 Face의 x좌표 구간이 $[L, R]$이라 합시다.
- 실험 (A, B)가 이 Face를 자르기 위한 조건은 무엇일까요?
- $L < A < R$ 이거나 $L < B < R$ 이면 됩니다.
- x좌표에 대해 좌표 압축을 해 주면 모든 좌표가 $O(N)$ 범위에 있게 할 수 있습니다.
- 좌표 압축 방법에 따라 위의 식 어딘가에 등호가 붙을 수도 있습니다. 그건 디테일 차이.

Div1H. 삼분 그래프

- 뭔가 좌표의 순서쌍이 여러 개 있으니 2차원 평면에 점을 찍어 보고 싶습니다.
- (Unbounded area를 제외한) 모든 Face $[L, R]$에 대해, 좌표평면의 (L, R)에 점을 하나씩 찍읍시다.
- 결국 실험 (A, B)에 대한 ΔF는 다음 쪽의 왼쪽 그림이 나타내는 영역에 속한 Face 점의 수입니다.
- $y = x$ 직선 아래에 점이 하나도 없다는 점을 이용하면, 다음 쪽의 오른쪽과 같은 직사각형 영역 안에 있는 점의 수를 세 주기만 하면 됩니다. 아니면, 포함배제를 사용해서 직사각형 쿼리를 3번 써도 됩니다.

Div1H. 삼분 그래프



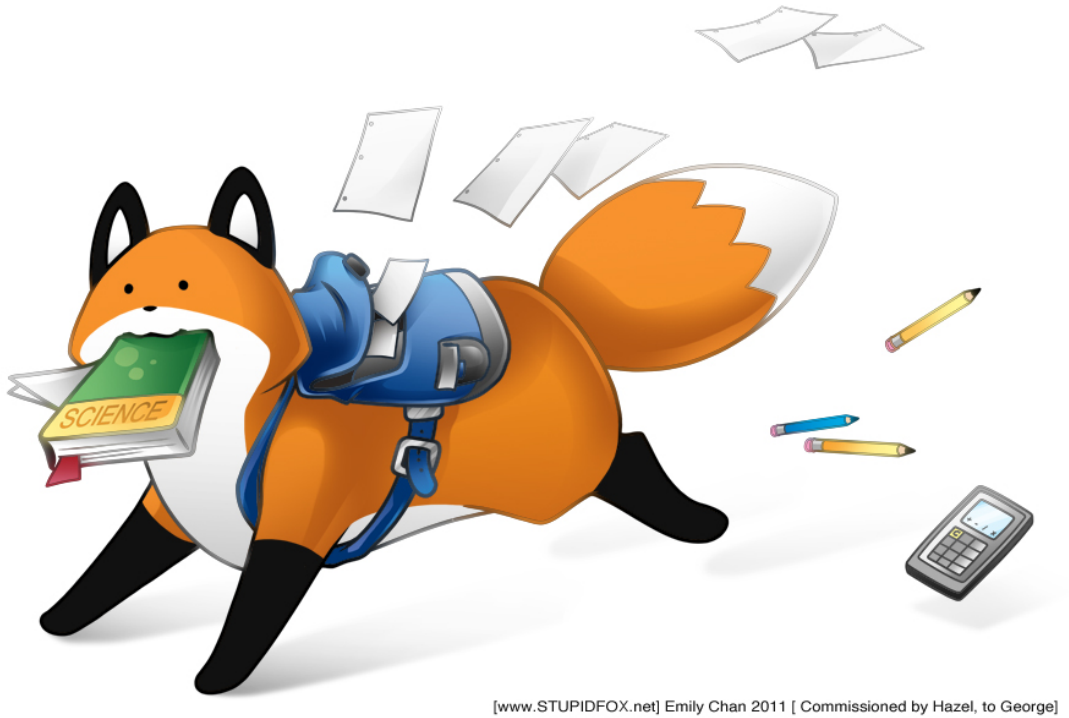
Div1H. 삼분 그래프

- 쿼리를 오프라인으로 처리해도 된다는 점을 이용하여 Plane Sweeping으로 각 쿼리에 대한 답을 한 번에 구해주면 코딩이 가장 편합니다.
- PST / 2D Segment Tree 등의 자료구조로 쿼리를 온라인으로 처리할 수도 있습니다. 대신 이건 코딩이 좀 더 귀찮습니다.
- $O((N+M+Q)\log N)$ 내지는 $O(N+M+Q)\log^2 N)$ 시간복잡도에 푸시면 충분하겠습니다.

Div1I. 여우 퀴즈

- 제출 3회, 정답 1명 (1명 시도, 정답률 33.33%)
- 처음 푼 참가자: 박상수 (224분)
- 출제자: doju

Div1I. 여우 퀴즈



- 여우는 사랑입니다.
- 점점 어려워지는 여우 문제

Div1I. 여우 퀴즈

- F로 바꾸는 과정이 없다면 간단한 구간합 문제입니다.
- 쿼리 없이 전체 범위에 대해 답을 구하는 문제는 간단한 DP 문제입니다.
- 이 문제는 둘을 합쳤으니 간단한 구간 DP 문제입니다.

Div1I. 여우 퀴즈

- 각 문제를 F로 바꿨을 때의 이득을 구해 봅시다.
  - F로 바꿨을 때 FOX가 만들어지지 않는다면 바꿀 필요가 없습니다.
  - 맞은 문제를 바꿔서 FOX를 만든다면 $B - A$점을 얻습니다.
    - $A > B$라면 손해입니다.
  - 틀린 문제를 바꿔서 FOX를 만든다면 B점을 얻습니다.

Div1I. 여우 퀴즈

- 퀴즈 범위가 주어지면 해당 범위에서 F로 바꿔서 이득을 볼 수 있는 문제를 전부 바꾸면 됩니다.

Div1I. 여우 퀴즈

- 퀴즈 범위가 주어지면 해당 범위에서 F로 바꿔서 이득을 볼 수 있는 문제를 전부 바꾸면 됩니다.
- XOXOX와 같은 꼴에서는 성립하지 않습니다.
  - 1번과 3번을 모두 바꾸면 FOFOX가 되고, 보너스 점수를 한 번밖에 받을 수 없습니다.
- 따라서 한 칸 혹은 두 칸 떨어진 두 문제를 동시에 F로 바꾸면 안 된다는 조건이 붙습니다.

Div1I. 여우 퀴즈

- 여기까지만 관찰해도 문제를 풀 수 있습니다.
  - 각 구간에 대해 왼쪽 두 문제와 오른쪽 두 문제의 선택 여부에 따른 최댓값을 저장하는 구간 트리를 만들면 됩니다.
  - 노드마다 16개 또는 9개의 값을 저장하고 있어야 합니다.
- 관찰을 약간 더 해 봅시다.

Div1I. 여우 퀴즈

- 어떤 문제를 F로 바꿨을 때 이득을 얻는다면, 그 앞이나 뒤의 문제는 F로 바꾸더라도 FOX가 만들어지지 않습니다.
- 즉 이웃한 두 문제가 둘 다 F로 바꿨을 때 이득이 되는 경우는 없습니다.

Div1I. 여우 퀴즈

- 어떤 문제를 F로 바꿨을 때 이득을 얻는다면, 그 앞이나 뒤의 문제는 F로 바꾸더라도 FOX가 만들어지지 않습니다.
- 즉 이웃한 두 문제가 둘 다 F로 바꿨을 때 이득이 되는 경우는 없습니다.
- 따라서 문제들을 홀짝성에 따라 나눈 뒤 따로 답을 구하고 나중에 합쳐도 충돌하지 않습니다.
- 이렇게 하면 노드마다 4개의 값을 저장하면서 문제를 풀 수 있습니다.

Div1J. 투명 악어

- 제출 43회, 정답 3명 (14명 시도, 정답률 6.98%)
- 처음 푼 참가자: 박상수 (109분)
- 출제자: doju

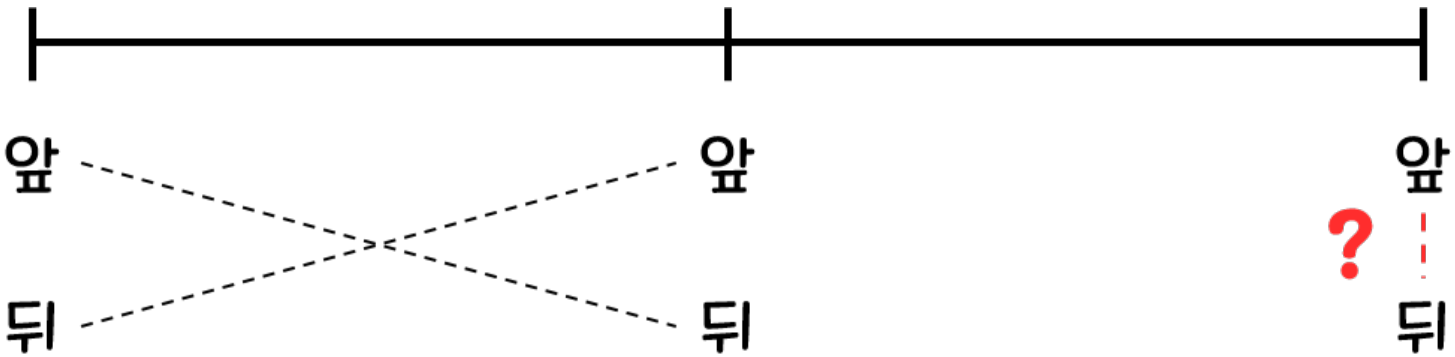
Div1J. 투명 악어

- 수가 주어지면 해당 위치에 앞발과 뒷발이 각각 몇 개인지 알 수 있습니다.
  - $4 =$ 뒷발 한 개
  - $5 =$ 앞발 한 개
  - ...
  - $18 =$ 뒷발 두 개, 앞발 두 개
  - $19 =$ 뒷발 한 개, 앞발 세 개
  - $20 =$ 뒷발 다섯 개 또는 앞발 네 개
- 앞발과 뒷발을 가장 작은 비용으로 전부 짝지어야 합니다.

Div1J. 투명 악어

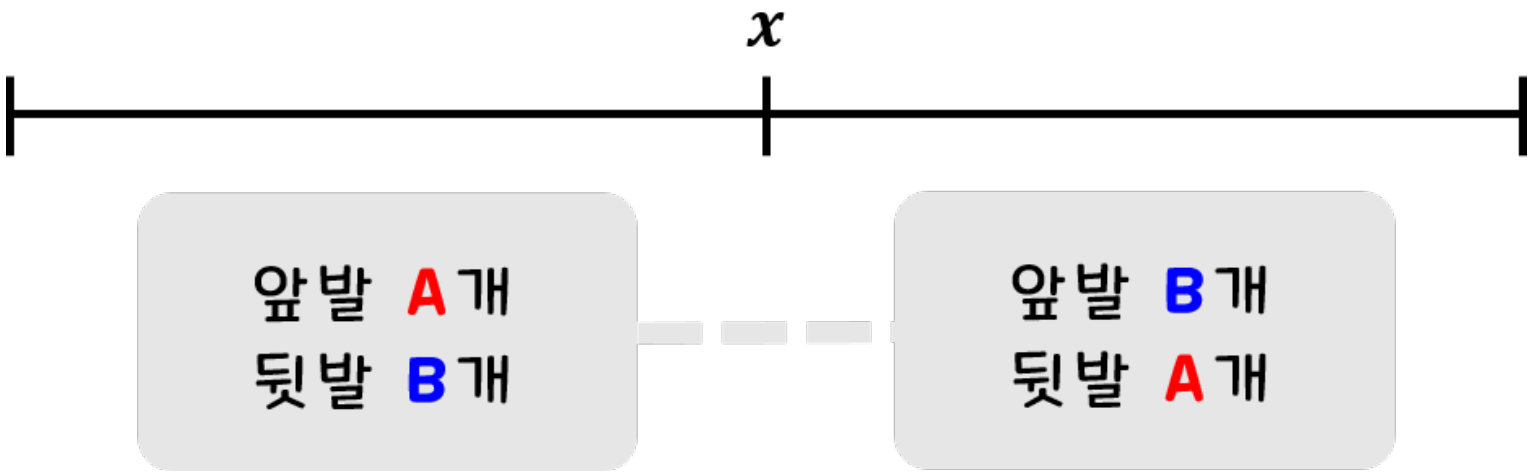
- 이런 문제는 앞에서부터 그리디하게 짝을 짓는 것이 최적임이 잘 알려져 있습니다.
- 이 문제는 같은 위치에 있는 두 발을 짝지을 수 없다는 조건이 붙어 있으므로, 이것만 주의해 주면 될 것 같습니다.

Div1J. 투명 악어



- 잘 안 됩니다.
- 같은 위치에 있으면 짝지을 수 없다는 조건은 생각보다 강력합니다.

Div1J. 투명 악어



- 늪지대를 임의의 x를 기준으로 두 구간으로 나눕니다.
- 왼쪽 구간에 속한 발들끼리 짝을 짓고 앞발 A개와 뒷발 B개가 남았다고 가정합니다.
- 모든 발이 짝지어져야 하므로, 오른쪽 구간에서도 서로 짝지어진 뒤 앞발 B개와 뒷발 A개가 남아야 합니다.

Div1J. 투명 악어

- 한쪽 구간에 앞발과 뒷발이 둘 다 **많이** 남아 있다면 서로 짝지어 주는 것이 더 이득일 것 같습니다.
- **많이**의 기준은 얼마일까요?

Div1J. 투명 악어

- 한쪽 구간에 앞발과 뒷발이 둘 다 **많이** 남아 있다면 서로 짝지어 주는 것이 더 이득일 것 같습니다.
- **많이**의 기준은 얼마일까요?

하나, 둘, 많다?

어떤 답을 임의의 x를 기준으로 잘랐을 때 한쪽 구간에 남아 있는 발의 조합이 **특정 조건**을 만족한다면 더 좋은 답이 존재한다.

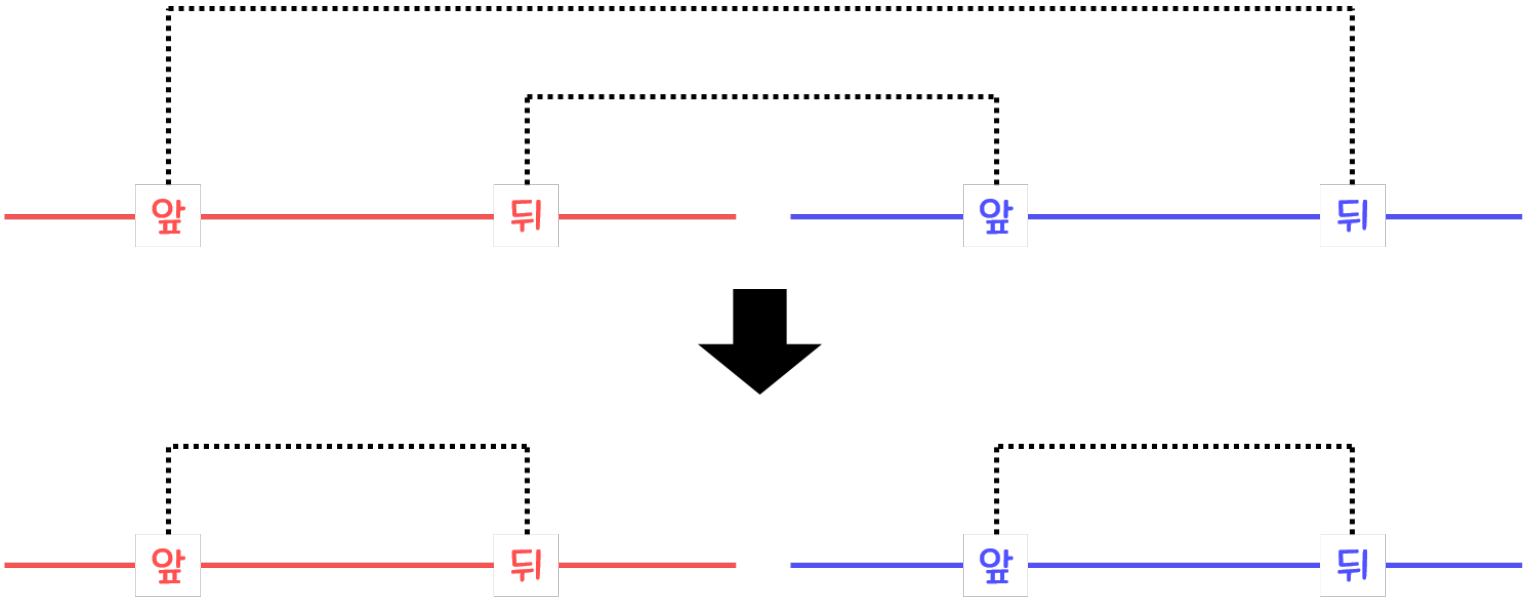
Div1J. 투명 악어

- 왼쪽 구간에 앞발과 뒷발이 **둘 다** 남아 있고, 남아 있는 발들의 발가락 개수의 합이 **20 이상**이라고 가정합니다.
- 오른쪽 구간도 같은 조건을 만족합니다.
  - 증명은 간단하니 직접 해 보세요 :)

Div1J. 투명 악어

- 왼쪽 구간에 앞발과 뒷발이 **둘 다** 남아 있고, 남아 있는 발들의 발가락 개수의 합이 **20 이상**이라고 가정합니다.
- 오른쪽 구간도 같은 조건을 만족합니다.
  - 증명은 간단하니 직접 해 보세요 :)
- 발가락이 20개 이상 남아 있으려면 두 곳 이상의 위치에 발이 놓여 있어야 합니다.
  - 따라서 서로 다른 위치에 있는 앞발과 뒷발을 하나씩 고를 수 있습니다.

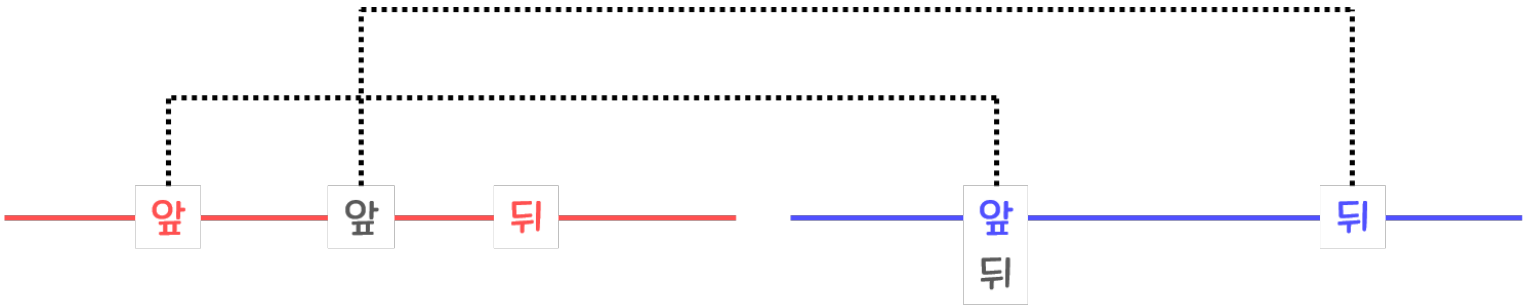
Div1J. 투명 악어



- 목표는 위와 같이 짝지어진 것을 아래와 같이 바꿔서 더 작은 답을 만드는 것입니다.

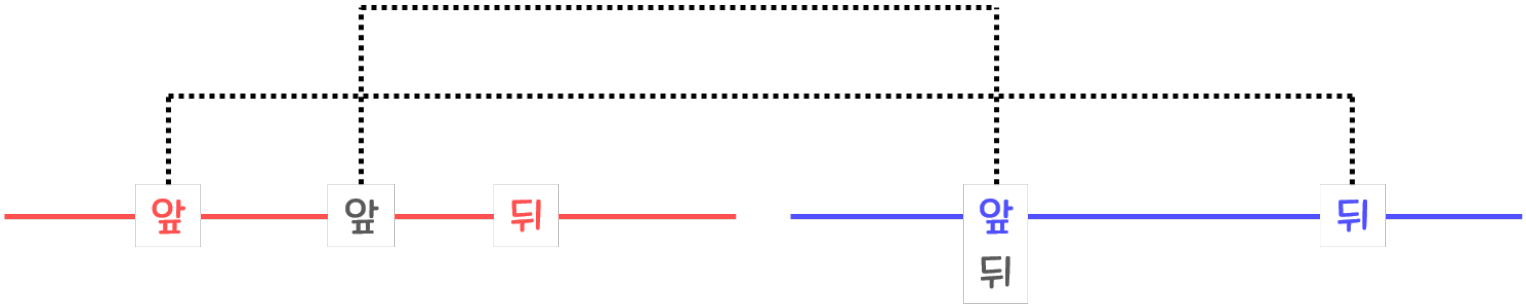
Div1J. 투명 악어

- 처음부터 원하는 대로 짝지어져 있다면 좋겠지만, 그렇지 않은 경우를 가정해 봅시다.



- 왼쪽의 앞발과 오른쪽의 뒷발이 서로 다른 상대와 짝지어져 있습니다.

Div1J. 투명 악어



- 서로의 상대를 바꾸면 처음 골랐던 앞발과 뒷발이 짝지어지며, 이때 비용은 변하지 않습니다.
- 왼쪽의 뒷발과 오른쪽의 앞발에 대해서도 같은 처리를 할 수 있습니다.

Div1J. 투명 악어

- 따라서 양쪽 구간에서 서로 다른 두 위치에 있는 앞발과 뒷발을 고를 수 있다면 항상 더 좋은 답을 만들 수 있습니다.
- 즉 최적의 답에서 왼쪽 구간에 앞발과 뒷발이 모두 남아 있다면 그 발가락 수의 합은 20 미만입니다.
  - 앞발 한 개, 뒷발 세 개 이하
  - 앞발 두 개, 뒷발 두 개 이하
  - 앞발 세 개, 뒷발 한 개

Div1J. 투명 악어

- 정리하면 다음과 같은 상태만을 고려하면 됩니다.
  - 앞발만 남아 있는 경우
  - 앞발이 없는 경우
  - 앞발이 한 개 남은 경우
  - 앞발이 두 개 남은 경우
  - 앞발이 세 개 남은 경우
- 각 경우에 대한 최소 비용을 dynamic programming으로 구하면 $\mathcal{O}(N)$에 문제를 해결할 수 있습니다.

Div1K. range

- 제출 4회, 정답 1명 (1명 시도, 정답률 25.00%)
- 처음 푼 참가자: 박성관 (203분)
- 출제자: kipa00

Div1K. range

- 파이썬에서는 두 `range`가 같다는 것을 그 수열 표현으로 확인함
- 주어진 `range`와 같다고 판별되는 `range`를 하나 출력

Div1K. range

- 파이썬에서는 두 `range`가 같다는 것을 그 수열 표현으로 확인함
- 주어진 `range`와 같다고 판별되는 `range`를 하나 출력
- 여러 개 있으면 아스키 **사전순 가장 작은 것** 출력

Div1K. range

- 파이썬에서는 두 range가 같다는 것을 그 수열 표현으로 확인함
- 주어진 range와 같다고 판별되는 range를 하나 출력
- 여러 개 있으면 아스키 **사전순 가장 작은 것** 출력
- **죄송합니다**

Div1K. range

- 한 글자씩 채워 나가는 전략이 도움이 많이 됩니다
- 수열의 길이가 0인 경우: `range(-1)`
  1. `range(-` 어쩌구가 가능한가? 당연히 가능
  2. `range(-1` 어쩌구가 가능한가? `range(-1,-2)` 등이 있으므로 가능
  3. **다음으로 올 수 있는 가장 작은 문자는)**
  4. `range(-1)`이 가능한가? **가능**

Div1K. range

- 수열의 길이가 1인 경우(원소는 s 하나)
  1. 시작은 무조건 s여야 함: `range(s,`
  2. `range(s,-1`가 가능한가? c의 절댓값을 **엄청나게 키우면** 가능해 보임
  3. `range(s,-1,`가 가능한가? 마찬가지로 가능해 보임
  4. 절댓값이 무진장 큰 c로 무조건 $|c| = 10^k$ 꼴을 선택

Div1K. range

- **틀렸습니다!** c가 작은 경우를 잘 고려합시다.
- 두 개의 예외 사항
  - $s = -1$인 경우 `range(-1,-10,-10)`
  - $s = -2$인 경우 `range(-2,-1)`

Div1K. range

- 수열의 길이가 2 이상인 경우
  1. a와 c가 고정
  2. b로 가능한 것이 적당한 범위 $s \leq b \leq e$에 들어옴
  3. 범위 중 가장 작은 것을 고르기
  4. 자릿수가 같으면 절댓값이 가장 작은 것
  5. 아니면 $\pm 10^k$ 꼴을 고르면 됨

Div1K. range

- **틀렸습니다!**
- $[s, e]$에 -1이 포함되면 무조건 그걸 골라주면 됨

Div1L. 옥상 정원

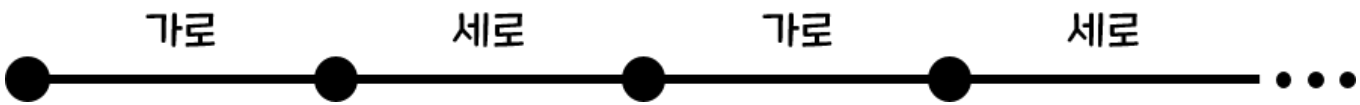
- 제출 2회, 정답 1명 (1명 시도, 정답률 50.00%)
- 처음 푼 참가자: 박범수 (178분)
- 출제자: doju

Div1L. 옥상 정원

- 같은 방향으로 이동하면 안 된다는 조건이 없다면 단순히 오일러 경로를 찾는 문제입니다.
- 깊이 우선 탐색을 이용한 해결 방법이 잘 알려져 있습니다.

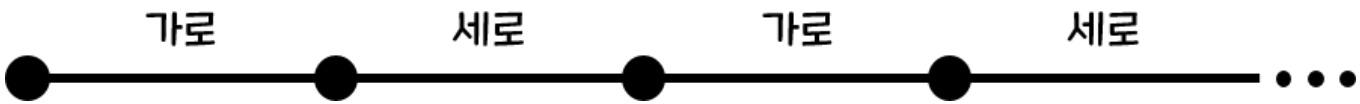
Div1L. 옥상 정원

- 평석이는 항상 세로 방향과 가로 방향을 번갈아가며 이동하게 됩니다.
  - 직진하거나 뒤로 되돌아가는 것을 제외하면 왼쪽이나 오른쪽 길로 갈 수밖에 없습니다.



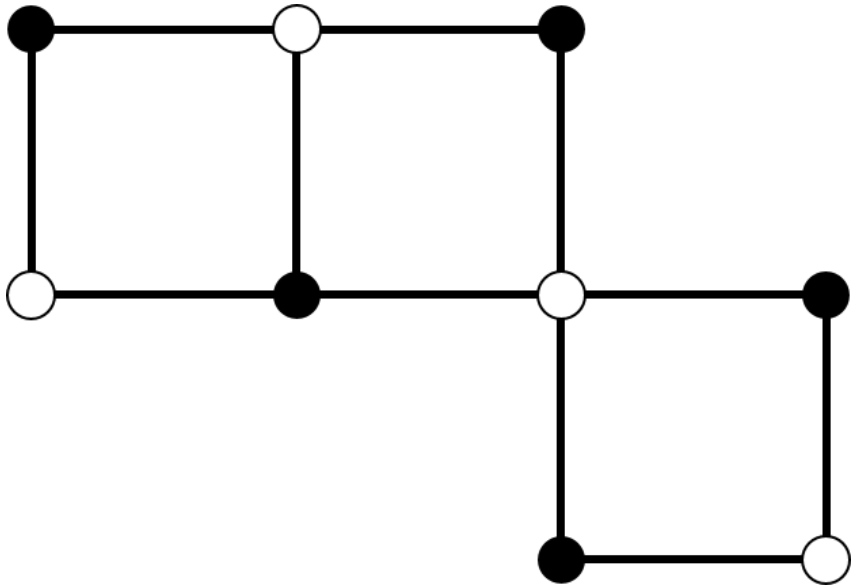
Div1L. 옥상 정원

- 평석이는 항상 세로 방향과 가로 방향을 번갈아가며 이동하게 됩니다.
  - 직진하거나 뒤로 되돌아가는 것을 제외하면 왼쪽이나 오른쪽 길로 갈 수밖에 없습니다.



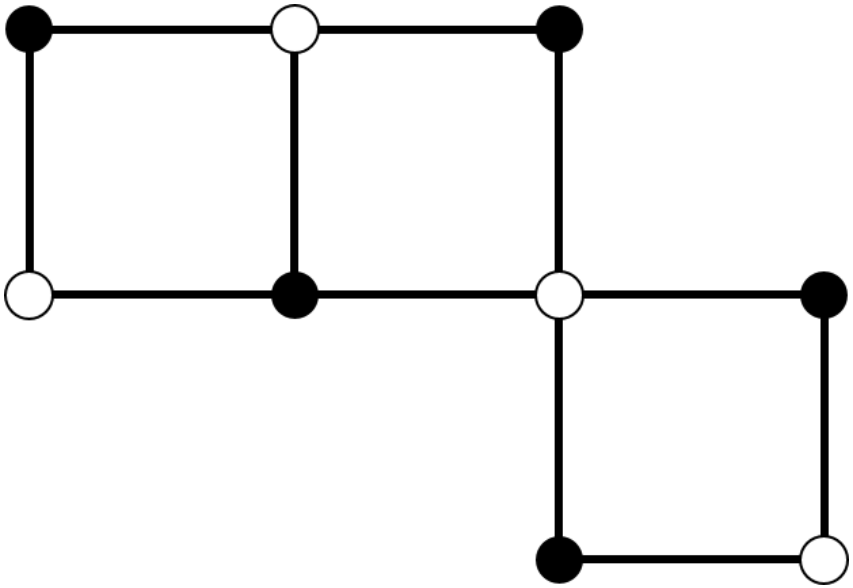
- 어떤 것이 **번갈아가며** 일어난다는 것은 좋은 성질입니다.
- 이 성질을 이용해 길에 방향을 부여해 봅시다.

Div1L. 옥상 정원



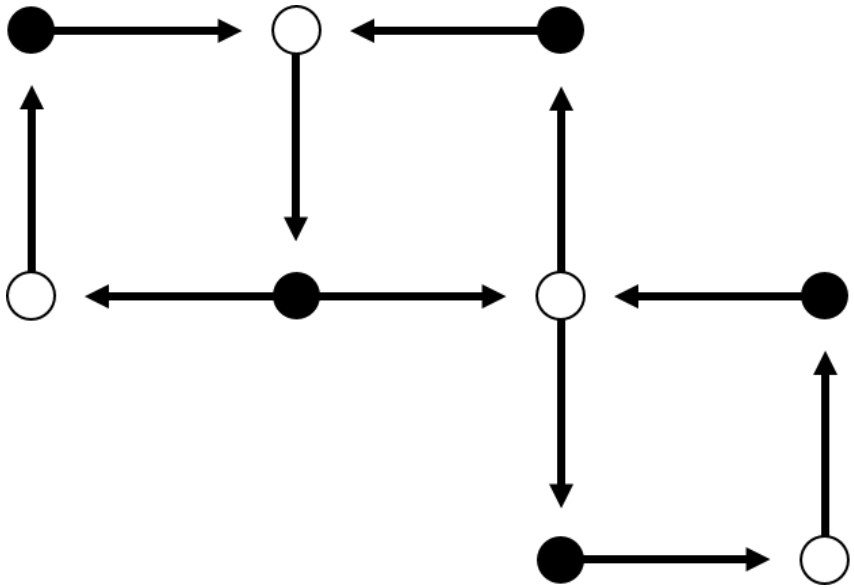
- 그래프의 각 정점을 홀짝성에 따라 색칠합니다.
  - UCPC에 참가하셨던 분들은 데자뷰를 느끼실 것입니다

Div1L. 옥상 정원



- 그래프의 각 정점을 홀짝성에 따라 색칠합니다.
  - UCPC에 참가하셨던 분들은 데자뷰를 느끼실 것입니다
- 이제 정점의 색깔과 이동 방향이 둘 다 번갈아가며 바뀝니다!

Div1L. 옥상 정원

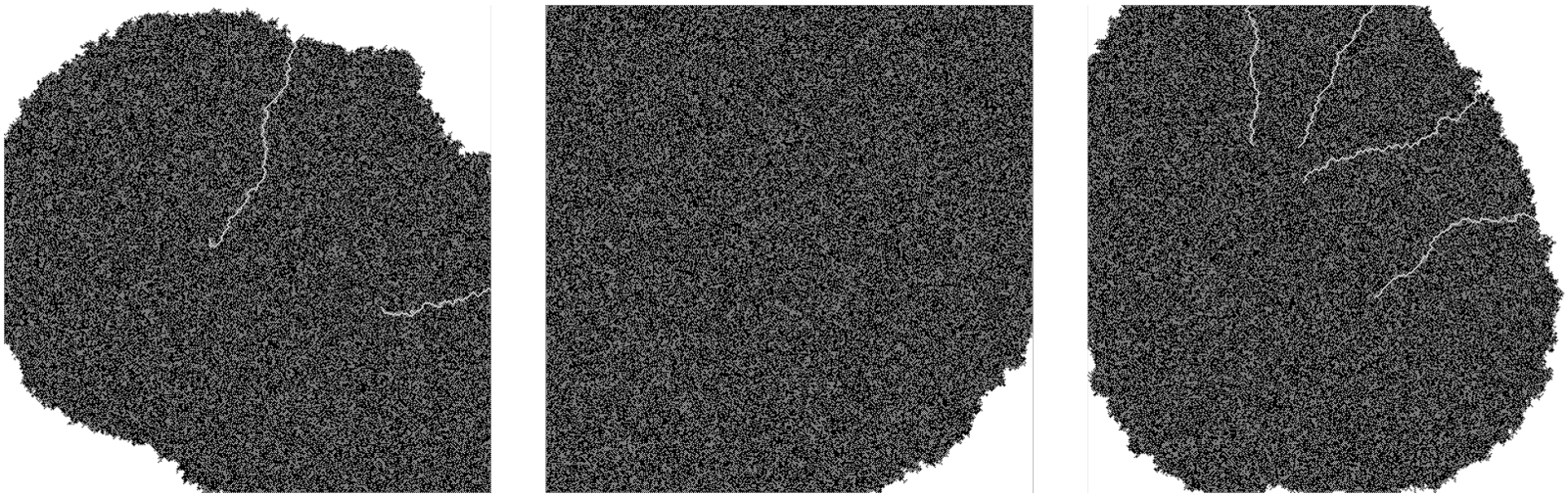


- 정점의 홀짝성에 따라 가로 방향 또는 세로 방향 간선에 방향을 부여해 주면 방향 그래프가 됩니다.
- 방향 그래프에서도 역시 깊이 우선 탐색으로 오일러 경로를 찾을 수 있습니다.

Div1L. 옥상 정원

- 정점의 홀짝성에 따라 가로 방향 또는 세로 방향 간선에 방향을 부여해 주면 방향 그래프가 됩니다.
- 방향 그래프에서도 같은 방법으로 오일러 경로를 찾을 수 있습니다.
  - Q. (홀수–가로, 짝수–세로)와 (홀수–세로, 짝수–가로) 둘 중 하나만 답이 있는 경우는 없나요?
  - A. 경로를 통째로 뒤집으면 되므로 아무거나 골라도 됩니다.

Div1L. 옥상 정원



- Bryan님이 빛나는 데이터를 만들어 주셨습니다!