Good Bye, BOJ 2022!

Official Solutions

프리즈 시점까지

599 명 참가

제출 **1,726** 회

AC **846** 회, 해결 된 문제 **8** 개

총괄

- 이종서 `leejseo` — KAIST 전산학부
- 김준겸 `ryute` — 고려대학교 컴퓨터학과

출제

- ✓ 나정휘 jhnah917 — 숭실대학교 컴퓨터학부
- ✓ 박찬솔 chansol — 숭실대학교 컴퓨터학부
- ✓ 신기준 sharaelong — 서울대학교 컴퓨터공학부
- ✓ 심유근 cozyyg — 서울대학교 컴퓨터공학부
- ✓ 오주원 kyo20111 — 숭실대학교 소프트웨어학부
- ✓ 정우경 man_of_learning — 전북대학교 컴퓨터공학부
- ✓ 정현서 jhwest2 — 서울대학교 컴퓨터공학부
- ✓ 한동규 queued_q — UNIST 컴퓨터공학부

검수

- 김준겸 ryute 고려대학교 컴퓨터학과
- 김준서 junseo 한양대학교 컴퓨터소프트웨어학부
- 노현서 gustjwkd1007 서울대학교 컴퓨터공학부
- 윤시우 cgiosy 서울사이버대학교 컴퓨터공학과
- 이성서 edenooo 숭실대학교 컴퓨터학부
- 이성현 hibye1217 한양대학교 컴퓨터소프트웨어학부
- 이종서 leejseo KAIST 전산학부
- 최준석 stonejjun03 고려대학교

조판

- 박수현 shiftpsh — 서강대학교 컴퓨터공학과
- 이종서 leejseo — KAIST 전산학부

디자인

- 박수현 shiftpsh — 서강대학교 컴퓨터공학과
- 이종서 leejseo — KAIST 전산학부
- 최희원 havana723 — 고려대학교

스트리밍

- ✓ 나정휘 `jhnah917` — 숭실대학교 컴퓨터학부
- ✓ 이종서 `leejseo` — KAIST 전산학부
- ✓ 정우경 `man_of_learning` — 전북대학교 컴퓨터공학부
- ✓ 정재헌 `gravekper`

Sponsors

STARTLINK

Proudly Supported By:

leejseo.com

Sponsors

Sponsors

SAMSUNG SOFTWARE MEMBERSHIP

Presto

문제		의도한 난이도	출제
A	2022년이 아름다웠던 이유	**Easy**	jhnah917
B	나무 블럭 게임	**Easy**	kyo20111
C	데이터 순서 복원	**Medium**	chansol
D	이미지 보정 작업	**Medium**	kyo20111
E	리그전	**Medium**	cozyyg
F	전설의 고대 광산 탈출	**Medium**	queued_q, man_of_learning
G	크루스칼 알고리즘	**Hard**	jhnah917
H	Maxtrix	**Hard**	sharaelong

First to Solve (프리즈 시점 기준)

UCPC greetings 22/23

문제		처음 푼 사람	해결 시각
A	2022년이 아름다웠던 이유	dotorya	1분
B	나무 블럭 게임	dotorya	5분
C	데이터 순서 복원	zoxl	6분
D	이미지 보정 작업	dotorya	17분
E	리그전	패트와매트	33분
F	전설의 고대 광산 탈출	yclock	39분
G	크루스칼 알고리즘	koosaga	40분
H	Maxtrix	육군상병박상수	97분

A. 2022년이 아름다웠던 이유

number_theory

출제진 의도 – **Easy**

- 문제 아이디어: jhnah917
- 문제 세팅: jhnah917

UCPC greetings 22/23

- 어떤 수의 약수를 모두 알고 있다면 그 수가 완전수, 과잉수, 부족수 중 어떤 것인지 판별할 수 있습니다.
- N 의 모든 약수는 $\mathcal{O}(N)$ 또는 $\mathcal{O}\left(\sqrt{N}\right)$ 시간에 구할 수 있습니다.
- 두 방법 모두 통과 가능하도록 문제 제한을 설정했습니다.

B. 나무 블럭 게임

ad_hoc

출제진 의도 – **Easy**

- 문제 아이디어 : kyo20111
- 문제 세팅 : kyo20111

- 풀이에서 같은 주머니에 들어있는 블럭들은 같은 그룹에 포함되어 있다고 표현하겠습니다.
- 우선, 입력으로 들어온 A를 오름차순으로 정렬합니다.
- A_{N-1} 은 항상 답으로 만들 수 있습니다.
  - $\{A_1, A_2, \cdots, A_{N-2}\}, \{A_{N-1}\}, \{A_N\}$
- 그러므로 A_{N-1} 보다 큰 평균을 가진 그룹을 찾아야 답의 후보가 될 수 있고, 그런 그룹은 A_N 을 포함해야 합니다.

UCPC greetings 22/23

- A_N 을 포함하는 그룹의 평균이 정렬했을 때 $\left\lfloor \frac{K+1}{2} \right\rfloor$ 번째에 위치하는 방법은 어떤 그룹의 평균보다 작거나 같은 평균을 가지거나 모든 블럭을 하나의 그룹에 포함시키는 두 가지 방법 뿐입니다.
- A_N 을 제외하고 만들 수 있는 평균이 가장 큰 그룹은 $\{A_{N-1}\}$ 이고, 이 그룹보다 작거나 같은 평균을 가진 그룹을 만들어도 A_{N-1} 을 답으로 만들 수 있으므로 이 경우는 답이 될 수 없습니다.
- 하지만 모든 블럭을 하나의 그룹에 포함시키는 방법은 A_{N-1} 보다 큰 평균을 가질 수 있으므로 이를 $\{A_{N-1}\}$ 과 같이 처리해주면 문제를 해결할 수 있습니다.

C. 데이터 순서 복원

ad_hoc
출제진 의도 – **Medium**

- 문제 아이디어: chansol
- 문제 세팅: chansol

UCPC greetings 22/23

- 간단한 관찰을 해봅시다:

- 데이터에서 i 번째 단계가 있어야 할 인덱스를 a, 변형된 데이터 (입력) 에서의 위치한 인덱스를 b 라고 합시다.
- 1. i 번째 단계가 원래보다 앞에 오는 경우, $b > a$
- 2. 정확히 한 칸 밀리는 경우, $b = a + 1$
- 3. 그렇지 않은 경우, $a = b$

- 각 단계는 최대 한 번만 자신의 위치보다 앞으로 이동할 수 있으므로 각 단계가 위치할 수 있는 서로 다른 인덱스의 개수는 많아 봐야 3개입니다.
- 서로 다른 인덱스의 개수가 3개이면, 그중 중앙값이 원래 있어야 하는 위치입니다.

- 서로 다른 3개의 인덱스에 위치한 단계가 하나라도 있으면:

- 그중 중앙값이 원래 그 단계가 있어야 하는 위치입니다.
- 3개의 변형된 데이터 중 언제 그 단계가 앞으로 갔는지도 알 수 있습니다.
- 이때의 데이터를 기준으로 그 단계가 원래 있어야 할 위치로 복원시켜주면 답을 구할 수 있습니다.

- 서로 다른 3개의 인덱스에 위치한 단계가 없으면:

- 첫 관찰에서 각 단계가 위치하는 인덱스에 따라 1, 2, 3번 경우로 각각 나누었는데, 한 단계가 3개의 변형된 데이터에서 위치할 수 있는 모든 경우를 고려해봅시다.
- A. (1, 2, 2) / B. (1, 3, 3) / C. (2, 3, 3)
- D. (2, 2, 3) / E. (2, 2, 2) / F. (3, 3, 3)
- 위와 같이 6개의 경우로 나눌 수 있습니다.

- B, C, F(한 칸 밀린 경우 2번)인 단계가 하나라도 존재하면 서로 다른 3개의 인덱스에 위치한 단계가 존재합니다.
- 이는 간단히 증명할 수 있습니다.

- 나머지 경우는 A, D, E 중 하나입니다.
- 이 경우들은 원래 있어야 할 인덱스에 2번 이상씩 위치합니다.
- 즉, **서로 다른 3곳에 위치한 단계**가 없을 때, 각 단계가 2번 이상 위치한 자리가 그 단계가 원래 있어야 할 위치입니다.

- 이 외에 다양한 풀이가 있습니다.

- 입력을 뒤에서부터 보면서 많이 등장한 것부터 고릅니다.

- 비교 함수 cmp(a, b)를 정의합니다.
- a가 b보다 앞에 등장한 경우가 많으면 a를 앞으로, 그렇지 않으면 b를 앞으로 보냅니다.

D. 이미지 보정 작업

binary_search, graph_traversal
출제진 의도 – **Medium**

- 문제 아이디어: kyo20111
- 문제 세팅: kyo20111

- x를 인접한 두 구역의 선명도의 가장 큰 차이로 만드는 방법이 있다면 이 방법 그대로 $x+1$을 답으로 만들 수 있습니다.
- 그러므로 가장 큰 차이를 x로 만드는 데 보정해야 하는 구역의 최소 개수는 차이를 $x+1$로 만드는 최소 개수보다 작거나 같고, 이를 이용해 이분탐색을 할 수 있습니다.

- 이분탐색에서 인접한 두 구역의 선명도 차이로 가능한 최댓값 mid을 고정한다고 합시다.
- 어떤 인접한 두 구역이 mid보다 더 큰 차이를 낸다고 하면, 둘 중 하나 혹은 둘 모두의 선명도를 보정해야 합니다.
- 둘 중 선명도가 큰 값을 가진 구역을 보정한다면 차이는 더 벌어지기 때문에 차이가 줄어들 가능성이 있는 더 작은 값을 가진 구역을 우선적으로 보정해야 합니다.

- 그렇기 때문에 $X - mid$ 이상의 선명도를 가진 구역은 보정할 필요가 없습니다.
- $X - mid$ 이상의 선명도를 가진 구역은 서로가 인접해 있을 경우 차이는 mid 보다 작거나 같습니다.
- $X - mid$ 미만의 선명도를 가진 구역과 인접해 있어 차이가 mid 보다 크더라도 우선 $X - mid$ 미만의 구역을 보정해야 하고, 이후 그 구역과 차이는 mid 보다 작거나 같게 됩니다.

- 반면에, $X - mid$ 미만의 선명도를 가진 구역은 차이가 M 초과인 경우가 생긴다면 무조건 보정해야 합니다.
- 앞에서 $X - mid$ 이상의 구역과 차이가 mid 을 초과해서 나는 경우는 설명했습니다.
- 같은 $X - mid$ 미만의 구역들끼리 차이나는 경우 한 구역의 선명도를 X 로 보정해도 차이는 그대로 mid 보다 크기 때문에 나머지 하나도 보정해야 합니다.

UCPC greetings 22/23

- 여기서 알 수 있는건, 선명도가 $X - mid$ 미만의 구역의 경우 인접한 구역이 보정을 받으면 마찬가지로 보정을 받아야 한다는 것입니다.
- 그러므로 선명도가 $X - mid$ 미만이면서 인접한 구역은 한번에 묶어서 처리할 수 있습니다.
- 하나의 묶음 안에서 한 구역이라도 인접한 구역과의 선명도 차이가 mid 을 초과한다면 그 묶음에 속한 모든 구역을 보정해야 합니다.
- 이를 통해 보정해야 하는 구역의 최소 개수를 구할 수 있고, 이 값을 K 와 비교해서 이분탐색을 이어나가면 됩니다.

- 묶음을 찾고 선명도 차이를 확인하는 건 그래프 탐색 알고리즘으로 $\mathcal{O}(NM)$에 해결할 수 있으며, 여기에 이분탐색의 시간복잡도가 붙어 총 $\mathcal{O}(NM \log X)$가 됩니다.
- 추가로 X가 시간복잡도에 들어가지 않는 $\mathcal{O}(NM \log NM)$ 풀이가 존재합니다.

E. 리그전

math, greedy, case_work

출제진 의도 – **Medium**

- 문제 아이디어: cozyyg
- 문제 세팅: cozyyg

- 승리, 무승부, 패배 시 승점이 각각 a, b, c점일 때 n팀 중 k등의 승점의 최댓값과 최솟값을 구해야 합니다.
- 승리 시 승점 a와 패배 시 승점 c가 바뀌어도 답은 동일합니다.
  - 모든 경기의 승패를 뒤집으면 (a, b, c)의 경우와 (c, b, a)의 경우가 일대일 대응됩니다.
- 일반성을 잃지 않고 $c \leq a$라 합시다.
- a, b, c의 크기 관계에 따라 4가지 경우로 나누어 문제를 풀어봅시다.

Case 1. $c \le b \le \dfrac{a+c}{2}$

Case 2. $\dfrac{a+c}{2} \le b \le a$

Case 3. $b \le c$

Case 4. $a \le b$

Case 1. $c \le b \le \dfrac{a+c}{2}$

✓ $a = 3, b = 1, c = 0, n = 4$일 때 최대 / 최소 승점과 그 예는 다음과 같습니다.

1	9	**9 (3-0-0)**	6 (2-0-1)	3 (1-0-2)	0 (0-0-3)
	3	**3 (0-3-0)**	**3 (0-3-0)**	**3 (0-3-0)**	**3 (0-3-0)**
2	7	**7 (2-1-0)**	**7 (2-1-0)**	1 (0-1-2)	1 (0-1-2)
	2	9 (3-0-0)	**2 (0-2-1)**	**2 (0-2-1)**	**2 (0-2-1)**
3	**6**	**6 (2-0-1)**	**6 (2-0-1)**	**6 (2-0-1)**	0 (0-0-3)
	1	7 (2-1-0)	7 (2-1-0)	**1 (0-1-2)**	**1 (0-1-2)**
4	**4**	**4 (1-1-1)**	**4 (1-1-1)**	**4 (1-1-1)**	**4 (1-1-1)**
	0	9 (3-0-0)	6 (2-0-1)	3 (1-0-2)	**0 (0-0-3)**

UCPC greetings 22/23

Case 1. $c \le b \le \dfrac{a+c}{2}$

- k 등의 승점의 최댓값을 구할 때, 1등부터 k 등까지를 A그룹, $k+1$ 등부터 n 등까지를 B 그룹이라고 합시다.

A그룹의 모든 팀이 **모든 B그룹의 팀**을 이긴 경우만 생각해도 됩니다.

- 만약 그렇지 않은 경기가 있다면, 그 경기를 A그룹의 팀이 이긴 것으로 바꿉니다.
- 이때 A그룹의 모든 팀은 승점이 감소하지 않고, B그룹의 모든 팀은 승점이 증가하지 않습니다.
- 따라서 k 등의 승점은 감소하지 않습니다.

Case 1. $c \leq b \leq \dfrac{a+c}{2}$

- 이제 A그룹의 모든 팀은 B그룹과의 경기에서 승점 $(n-k) \cdot a$ 점을 확보했습니다.
- A그룹 팀들의 승점이 비슷할수록 좋을 것 같습니다.
- x 개의 팀이 리그전을 할 때 x 등 팀의 승점의 최댓값을 $f(x)$ 라 합시다.
- 이때 구하려는 정답은 $f(k) + (n-k) \cdot a$ 가 됩니다.
- $f(x)$ 를 구해봅시다.

UCPC greetings 22/23

Case 1. $c \le b \le \dfrac{a+c}{2}$

$x = 2t+1$이면 $f(x) = t \cdot (a+c)$, $x = 2t$면 $f(x) = (t-1) \cdot (a+c) + b$입니다.

- 각 팀의 (승리 횟수) - (패배 횟수)를 모두 더하면 0이므로, **승리가 패배보다 많지 않은 팀**이 존재합니다.
- $2b \le a+c$이므로 무승부 2개를 1승 1패로 바꾸면 승점이 더 늘어납니다.
- 또한 패배를 무승부로 바꾸면 승점이 더 늘어납니다.
- 따라서 **승리가 패배보다 많지 않은 팀**의 승점이 최대인 경우는 승리 횟수와 패배 횟수가 같고, 무승부가 최대 1개 있습니다.
- 그러므로 승점의 최댓값은 $x = 2t+1$면 $t \cdot (a+c)$, $x = 2t$면 $(t-1) \cdot (a+c) + b$입니다.

Case 1. $c \le b \le \dfrac{a+c}{2}$

$x = 2t+1$ 이면 $f(x) = t \cdot (a+c)$, $x = 2t$ 면 $f(x) = (t-1) \cdot (a+c) + b$ 입니다.

- 위에서 구한 최댓값을 달성할 수 있습니다.
- x 개의 팀을 원형으로 배치한 다음, 시계방향의 호가 더 짧은 쪽이 이기도록 하면 됩니다.
- x 가 짝수인 경우 반대편 팀과의 경기 결과는 무승부입니다.
  - ex1. $x = 5 \Rightarrow 1 \to (2,3), 2 \to (3,4), 3 \to (4,5), 4 \to (5,1), 5 \to (1,2)$
  - ex2. $x = 6 \Rightarrow 1 \to (2,3), 2 \to (3,4), 3 \to (4,5), 4 \to (5,6), 5 \to (6,1), 6 \to (1,2),$
    $1 \leftrightarrow 4, 2 \leftrightarrow 5, 3 \leftrightarrow 6$

Case 1. $c \le b \le \dfrac{a+c}{2}$

- 최솟값을 구할 때도 비슷하게, 1등부터 $k-1$ 등까지를 A그룹, k 등부터 n 등까지를 B그룹으로 놓은 뒤 A그룹이 B그룹을 모두 이긴 경우만 생각해도 됩니다.
- x 개의 팀이 리그전을 할 때 1등 팀의 승점의 최솟값을 $g(x)$ 라 합시다.
- 구하려는 정답은 $g(n-k+1)+(k-1)\cdot c$ 이고, $g(x)=(x-1)\cdot b$ 입니다.

UCPC greetings 22/23

Case 2. $\frac{a+c}{2} \le b \le a$

- Case 1과 비슷하게 풀 수 있습니다.
- 결과적으로 f와 g만 바뀐다는 사실을 알 수 있습니다.
- 참고: $-c, -b, -a$에 대한 답을 구한 뒤 부호를 바꾸는 방식으로도 풀 수 있습니다.

UCPC greetings 22/23

Case 3. $b \le c$

- 모든 경기가 무승부라면 모든 팀의 승점이 $(n-1) \cdot b$이고 이는 가능한 승점의 최솟값입니다.
- 그러므로 k의 값에 관계 없이 승점의 최솟값은 $(n-1) \cdot b$입니다.
- 최댓값을 구할 때, 무승부 경기에 임의로 승패를 주면 모든 팀의 승점이 감소하지 않으므로 무승부가 없다고 생각해도 됩니다.
- Case 1과 비슷하지만, $x = 2t$일 때 $f(x) = (t-1) \cdot (a+c) + c$입니다.

Case 4. $a \le b$

- Case 3과는 반대로 최댓값이 $(n-1) \cdot b$로 정해지며, 최솟값을 구할 때 $x = 2t$면 $g(x) = (t-1) \cdot (a+c) + a$가 됩니다.

F. 전설의 고대 광산 탈출

dp, physics

출제진 의도 – **Medium**

- 문제 아이디어: queued_q
- 문제 세팅: man_of_learning

- 당신과 타고 있는 수레의 질량의 합이 M, 속도가 V 이고, 수레에 실으려는 광석 주머니의 위치가 x, 질량이 m, 그리고 주머니를 싣고 난 뒤의 속도가 v 라고 합시다.
- 운동량은 항상 일정하므로 $P = MV = (M+m)v$ 입니다.

UCPC greetings 22/23

- 광석 주머니의 위치에서 목적지까지 도달하는데 걸리는 시간을 구해봅시다.
- 광석 주머니를 싣지 않는 경우 시간은 아래와 같습니다.

$$t_0 = \frac{x}{V} = \frac{Mx}{MV} = \frac{Mx}{P}$$

- 광석 주머니를 싣는 경우 시간은 아래와 같습니다.

$$t = \frac{x}{v} = \frac{(M+m)x}{(M+m)v} = \frac{(M+m)x}{P} = t_0 + \frac{mx}{P}$$

- 광석 주머니를 싣지 않는 경우 시간: $t_0 = \frac{Mx}{P}$
  광석 주머니를 싣는 경우 시간: $t = t_0 + \frac{mx}{P}$
- $\frac{mx}{P}$ 는 수레의 초기 무게 M 과 초기 속도 V 에 영향을 받지 않고,
  광석 주머니에 대해 항상 일정한 값입니다.
- 즉, 현재 수레에 광석 주머니가 얼마나 담겨 있든 상관없이,
  광석 주머니를 하나 추가할 때 탈출에 걸리는 시간은 일정한 값만큼만 늘어납니다!

- 따라서 우리의 목표는

$$\frac{MX}{P} + \sum \frac{m_i x_i}{P} \le T$$

를 만족하면서 광석의 가치 합이 최대가 되는 주머니의 집합을 고르는 일입니다.

- 실수 자료형을 사용하지 않아야 합니다. 양변에 $P = MV$를 곱해주면,

$$MX + \sum m_i x_i \le MVT$$

와 같이 식에 정수들만 남습니다.

- 이처럼 0-1 냅색 문제로 환원할 수 있고, 다이나믹 프로그래밍을 이용하여 $\mathcal{O}(NMVT)$에 문제를 풀 수 있습니다.

G. 크루스칼 알고리즘

graphs, disjoint_set, linear_algebra, gaussian_elimination

출제진 의도 – **Hard**

- 문제 아이디어: jhnah917
- 문제 세팅: jhnah917

- 아래 세 가지 관찰이 필요하고, 모두 어렵지 않게 증명할 수 있습니다.
  1. 최소 신장 트리에 추가되는 간선의 가중치는 최대 $N-1$ 가지입니다.
  2. 가중치가 같은 간선을 모두 추가하면, 정렬 결과에 관계 없이 포레스트의 구성은 동일합니다.
  3. 최소 신장 트리에 추가되는 간선의 가중치 중복 집합은 항상 동일합니다.
- 최소 신장 트리를 구성하는 간선들의 가중치를 $w_1, w_2, \cdots, w_k (w_{i-1} < w_i)$ 라고 합시다.
- 가중치가 $w_1, w_2, \cdots, w_{t-1}$ 인 간선을 모두 추가한 상황에서
- 가중치가 w_t 인 간선을 추가하는 경우의 수를 구해 봅시다.

- 정렬 결과에 관계 없이 포레스트의 구성은 항상 동일합니다.
- 따라서 각 컴포넌트를 하나의 정점으로 압축할 수 있습니다.
- 간선을 최대한 많이 추가한 포레스트를 만드는 경우의 수를 구하면 됩니다.
- 이는 포레스트를 구성하는 트리를 만드는 경우의 수들의 곱과 동일합니다.
- 따라서 연결 그래프가 주어졌을 때 신장 트리를 만드는 경우의 수를 구하면 됩니다.

UCPC greetings 22/23

- n개의 정점으로 구성된 연결 그래프에서 신장 트리를 만드는 경우의 수는 (가능한 신장 트리의 가짓수) $\times(n-1)!$으로 구할 수 있습니다.
- 이때 가능한 신장 트리의 가짓수는 Kirchhoff's theorem을 이용해 구할 수 있습니다.
- 다음과 같은 행렬 $L_{n\times n}$를 정의합시다.

$$L_{i,j} := \begin{cases} \deg(v_i) & \text{if } i = j \\ -c & \text{if } i \neq j \text{ and } v_i \text{ is connected with } v_j \text{ by } c \text{ edges} \\ 0 & \text{otherwise} \end{cases}$$

- $L[i,j]$를 L의 i번째 행과 j번째 열을 제거한 행렬이라고 정의합시다.
- 임의의 i, j에 대해 $\det(L[i,j])$는 가능한 신장 트리의 가짓수와 동일합니다.

- $k \times k$ 행렬의 행렬식은 가우스 소거법을 이용해 $\mathcal{O}\left(k^3\right)$에 구할 수 있습니다.
- 따라서 k개의 압축 정점이 있는 포레스트를 만드는 경우의 수를 $\mathcal{O}\left(k^3\right)$에 구할 수 있습니다.
- 크기가 $k \times k$인 행렬의 행렬식을 한 번 구할 때마다 정점의 개수가 $k-1$개씩 감소합니다.
- 따라서 $\mathcal{O}\left(N^3\right)$에 전체 문제를 해결할 수 있습니다.
- Berlekamp-Massey를 이용해 행렬식을 구하면 $\mathcal{O}\left(NM + N \log MOD\right)$에 문제를 해결할 수 있습니다.

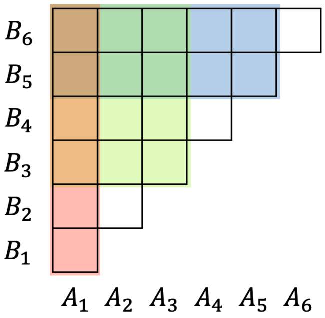
H. Maxtrix

divide_and_conquer, convex_hull_trick
출제진 의도 – **Hard**

- ✓ 문제 아이디어: sharaelong
- ✓ 문제 세팅: sharaelong, jhwest2

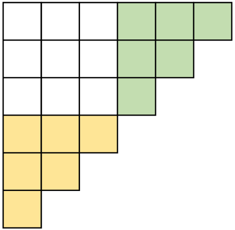
H. Maxtrix

- 구하는 값을 차례대로 $ans[1], \cdots, ans[N]$ 이라고 합시다.
- 이 때 이 값들은 각각 아래 그림의 영역에서의 최댓값입니다.



H. Maxtrix

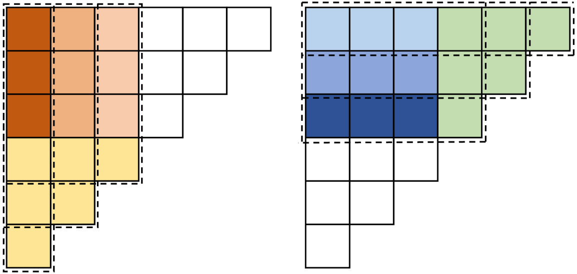
- 먼저 문제를 절반으로 나누어봅시다. 아래 그림의 영역으로 문제를 제한시킬 때의 답을 알고 있다고 가정합니다.



- 그러면 계산된 적이 없는 $1 \le i \le \frac{N}{2}, \frac{N}{2} + 1 \le j \le N$ 영역만 추가적으로 고려하면 됩니다.

H. Maxtrix

- 만약 그림과 같은 각각의 직사각형 영역에서의 최댓값을 전부 알고 있다면, max를 누적하여 계산하는 것으로 작은 문제의 결과를 큰 문제의 답으로 확장이 가능하다는 사실을 관찰할 수 있습니다.



H. Maxtrix

- 이 때 주황색 영역의 각 칸은 $-ji + B[j]$라는 i에 대한 일차함수의 $i = 1, \cdots, \frac{N}{2}$일 때의 값입니다. ($\frac{N}{2} + 1 \le j \le N$)
- 푸른색 영역의 칸들도 $-ij + A[i]$라는 일차함수에 대한 값이 됩니다.
- 즉 여러 개의 직선 ($y = ax + b$꼴) 집합에 대해, 특정 x좌표에서의 최댓값을 빠르게 계산할 수 있다면 문제가 풀립니다.
- 그런데 직선 기울기 ($-i$ 또는 $-j$)에 단조성이 있으므로, Convex Hull Trick(CHT)을 사용할 수 있습니다!

H. Maxtrix

- CHT로 총 N개의 직선 영역에서 최댓값을 계산해야 하므로, 크기 N짜리 문제를 푸는데는 $T(N) = 2T\left(\frac{N}{2}\right) + O(N \log N) = O(N \log^2 N)$ 시간이 소요됩니다.
- 이 문제의 경우 최댓값을 계산해야 하는 x좌표가 무엇인지도 사전에 알고 있으므로, 쿼리에도 단조성이 있는 CHT를 푸는 것으로 볼 수 있습니다.
- 따라서 CHT를 $O(N)$ 시간에 계산할 수도 있고, 이 경우 전체 문제의 시간복잡도는 $O(N \log N)$입니다.
- 두 방법 모두 통과되도록 제한이 설정되어 있습니다.