# Hello, BOJ 2025!

Official Solutions

온사이트 콘테스트에

**61**명 참가

---

제출 **535**회

AC **215**회, 해결된 문제 **6**개

**총괄 및 운영**

| | |
|---|---|
| ✓ 김준겸 `ryute` | 고려대학교 컴퓨터학과 |
| ✓ 나정휘 `jhnah917` | Quora |
| ✓ 이종서 `leejseo` | Moloco |

### 출제

- ✓ 김동현 kdh9949 — FuriosaAI
- ✓ 송준혁 songc — KAIST 전산학부
- ✓ 이종영 moonrabbit2 — KAIST 전산학부
- ✓ 조승현 ainta — Georgia Tech College of Computing

### 검수

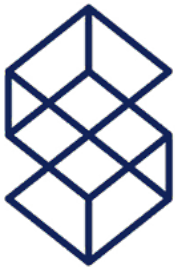
- ✓ 김세린 serin — KAIST 수리과학과
- ✓ 박상수 cki86201 — 삼성전자
- ✓ 오주원 kyo20111 — 삼성전자
- ✓ 이지후 silverwolf — 서울대학교 수리과학부
- ✓ 최준석 stonejjun03 — 삼성전자

## Sponsors






## Sponsors



SAMSUNG
SOFTWARE
MEMBERSHIP



| 문제 | | 의도한 난이도 | 출제 |
|---|---|---|---|
| **A** | 2025는 무엇이 특별할까? | **Easy** | kdh9949 |
| **B** | 극적인 승리 | **Easy** | kdh9949 |
| **C** | 둘이 한 팀 | **Medium** | kdh9949 |
| **D** | 사격 | **Medium** | songc |
| **E** | GB(Good Bye) | **Hard** | ainta |
| **F** | 셋이 한 팀 | **Hard** | kdh9949 |
| **G** | 조 나누기 | **Hard** | moonrabbit2 |

# A. 2025는 무엇이 특별할까?

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

- ✓ 처음 푼 사람: **ychangseok**, 59초
- ✓ 문제 아이디어: kdh9949
- ✓ 문제 세팅: kdh9949

## A. 2025는 무엇이 특별할까?

- ✓ 4자리 수 $Y$에 대해,
  - 앞 2자리는 $Y$를 100으로 나눈 몫,
  - 뒤 2자리는 $Y$를 100으로 나눈 나머지로 각각 구할 수 있습니다.
- ✓ 둘을 더해서 제곱한 값이 $Y$인지 확인하면 $Y$년이 특별한 해인지 알 수 있습니다.
- ✓ 제한이 작기 때문에 $Y$를 $X+1$부터 하나씩 늘려가며 모두 시도해 볼 수 있습니다.

# B. 극적인 승리

`sorting`, `binary_search`
출제진 의도 – **Easy**

- 처음 푼 사람: **tlwpdus**, 9분
- 문제 아이디어: kdh9949
- 문제 세팅: kdh9949

## B. 극적인 승리

- 우선, 한 손만 과녁을 맞히는 경우 / 두 손 다 안 맞히는 경우를 따로 먼저 처리합시다.
- 그런 방법은 $O(N)$ 가지 있으므로 그냥 다 해보면 됩니다.
- 이제 두 손 모두 과녁을 맞히는 경우에 대해 빨리 풀면 됩니다.

## B. 극적인 승리

- ”양 손으로 같은 과녁을 맞히면 0점이다” 라는 조건을 제외하면, 어디서 많이 본 문제가 됩니다.
- 두 정수 $A, B$와 길이 $N$의 두 배열 $L, R$이 주어지면, $L[i] + R[j]$가 $A - B$ 초과이면서 가장 작은 $(i, j)$ 쌍을 구하면 됩니다.
- 각 $L[i]$마다 $R[j] \geq A - B - L[i] + 1$를 만족하는 가장 작은 $R[j]$를 구하면 됩니다.
- $R$에서 (값, 원래 인덱스) 쌍을 값 순으로 정렬해 둔 뒤 이분 탐색을 이용하면 $i$ 하나당 $O(\log N)$, 총 $O(N \log N)$에 풀 수 있습니다.
- C++의 경우 `lower_bound`, Python의 경우 `bisect_left` 등의 표준 라이브러리 함수가 있습니다.

## B. 극적인 승리

- 이 문제에서는, $i = j$ 인 경우를 제외해야 합니다.
- 앞서 설명한 방법과 비슷하게 $R$을 정렬해 놓고 $A - B - L[i] + 1$ 이상인 최초 원소를 찾았다고 합시다.
- 그때 찾은 것이 하필 (원래 배열에서) $R[i]$ 일 경우만 문제가 되는데, 그 경우에는 정렬된 배열의 바로 다음 원소를 보면 됩니다.

# C. 둘이 한 팀

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

- 처음 푼 사람: **jhwest2**, 16분
- 문제 아이디어: kdh9949
- 문제 세팅: kdh9949

## C. 둘이 한 팀

- 팀의 능력치를 어떻게 구하는지 생각해 봅시다.
- $A_i \geq B_i$ 인 $i$ 들에 대해서는 $A_i$ 를 더하고, 그렇지 않은 $i$ 들에 대해서는 $B_i$ 를 더하면 됩니다.
- 이것을 빠르게 하고 싶습니다.

## C. 둘이 한 팀

- 능력치들이 초기 $A_i - B_i$ 의 오름차순으로 나열되도록 정렬했다고 합시다.
- 각 쿼리 이후, 지금까지 $A_i$ 들에 더해진 총 $X_j$ 값을 $T_A$, $B_i$ 들에 더해진 것을 $T_B$ 라고 합시다.
- $A_i + T_A \geq B_i + T_B$ 라는 조건은 $A_i - B_i \geq T_B - T_A$ 로 바꾸어 쓸 수 있습니다.
- 즉, 정렬된 배열에서 오른쪽 몇 개는 A가 이기고, 나머지는 B가 이기는 꼴이 됩니다.

## C. 둘이 한 팀

- 경계선이 되는 위치, 즉 $A_i - B_i < T_B - T_A$ 인 가장 큰 $i$를 구했다고 합시다.
- (그런 $i$가 존재하지 않는 경우 $0$)
- 이 때 팀의 능력치는 $\sum_{k=1}^{i} B_i + T_B + \sum_{k=i+1}^{N} A_i + T_A = \sum_{k=1}^{i} B_i + iT_B + \sum_{k=i+1}^{N} A_i + (N-i)T_A$ 입니다.
- 이분 탐색으로 $i$를 구하고, 누적합 배열로 팀의 능력치를 구하면 쿼리당 $O(\log N)$ 에 풀 수 있습니다.

# D. 사격

`dp`, `sweeping`, `segtree`, `lazyprop`
출제진 의도 – **Medium**

- 처음 푼 사람: **urd05**, 46분
- 문제 아이디어: songc
- 문제 세팅: songc

## D. 사격

- ✓ 모든 점을 x좌표 오름차순으로 정렬합시다.
- ✓ y축 사격은 인덱스 순서대로 하면 되므로, 점화식에서는 x축 사격만 고려해도 됩니다.
- ✓ 점화식을 아래와 같이 정의합시다.

  $D(i, y) := i$번째 점까지, x축 사격으로 제거한 y좌표의 최댓값이 $y$ 일 때 가능한 최대 점수
- ✓ $D(i, *)$를 모두 계산했고, $(i + 1)$번 점을 처리하는 상황을 생각해 봅시다.

## D. 사격

- 만약 $y > Y_{i+1}$ 이라면, $(i+1)$ 번째 점은 이미 x축 사격으로 없어졌을 것입니다.
- 따라서 이 경우에는 $D(i+1, y) \leftarrow D(i, y) + Y_{i+1}$ 입니다.
- 그렇지 않다면, $(i+1)$ 번 점은 x축 사격과 y축 사격 모두를 이용해 제거할 수 있습니다.
- x축에서 사격하는 경우, $D(i+1, Y_{i+1}) \leftarrow \max_{t<Y_{i+1}} D(i, t) + Y_{i+1}$ 입니다.
- y축에서 사격하는 경우에는 $y$ 값이 바뀌지 않으므로 $D(i, y) \leftarrow D(i, y) + X_{i+1}$ 입니다.

## D. 사격

- ✓ 정리하면, 아래 식을 이용해 $D(i, *)$ 에서 $D(i+1, *)$ 로 전이할 수 있습니다.
  1. $y > Y_{i+1}$ 이면 $D(i+1, y) \leftarrow D(i, y) + Y_{i+1}$
  2. $y < Y_{i+1}$ 이면 $D(i+1, y) \leftarrow D(i, y) + X_{i+1}$
  3. 또한, $D(i+1, Y_{i+1}) \leftarrow \max_{t<Y_{i+1}} D(i, t) + Y_{i+1}$
- ✓ 위 전이 과정을 다음과 같이 생각할 수도 있습니다.
  1. $D(i, *)$ 의 값을 $D(i+1, *)$ 로 복사
  2. $D(i+1, < Y_{i+1})$ 에 $X_{i+1}$ 을 더함
  3. $D(i+1, > Y_{i+1})$ 에 $Y_{i+1}$ 을 더함
  4. $D(i+1, Y_{i+1})$ 을 $\max_{t<Y_{i+1}} D(i, t) + Y_{i+1}$ 로 갱신
- ✓ Segment Tree + Lazy Propagation 을 이용해 $O(N \log N)$ 에 문제를 해결할 수 있습니다.

# E. GB(Good Bye)

`dp, combinatorics`
출제진 의도 – **Hard**

- 처음 푼 사람: **aeren**, 102분
- 문제 아이디어: `ainta`
- 문제 세팅: `ainta`

## E. GB(Good Bye)

- 빈 문자열에서 시작하여 GBGB 또는 BGBG를 추가하여 만들 수 있는 문자열에 대해 생각해 봅시다.
- 제거되는 문자들은 순서대로 읽었을 때 위 방법으로 만들어지는 문자열들 중 하나여야 합니다.

## E. GB(Good Bye)

- G와 B로 이루어진 문자열이 앞서 말한 경우에 해당될 조건은
- 1. 길이 $m$은 4의 배수
- 2. B와 G의 개수가 동일
- 3. 첫 $i$개 문자 중 (G의 개수) - (B의 개수)를 $S_i$라 할 때, $S_i$들의 최댓값과 최솟값의 차이가 $m/4$ 이하
- 3가지 조건을 모두 만족하는 것입니다.

## E. GB(Good Bye)

- $D[i][j][mn][mx][cur]$를 첫 $i$개 문자열의 subsequence 중 길이 $j$, $S$ 값의 현재까치 최댓값 $mn$, 최솟값 $mx$, 현재 $S$ 값 $cur$인 경우의 수로 정의하면
- 다이나믹 프로그래밍으로 $O(N^5)$ 시간에 문제를 해결할 수 있습니다.
- 위 정의로 문제를 해결하면 시간 또는 메모리가 제한을 초과할 수 있습니다.

## E. GB(Good Bye)

- 최종으로 제거되는 문자열의 길이 $m = 4k$라 할 때, $k \leq N/4$를 만족합니다.
- 중간에 $S$ 값의 최댓값과 최솟값은 최대 $k$만큼만 차이나야 합니다.
- $k, mn$을 미리 정해놓으면 $D[i][j][cur]$로 DP 식을 정의하여 $O(kN^2)$ 시간 및 공간복잡도에 문제를 해결할 수 있습니다.
- 이를 이용하면 문제를 총 $(N/4)^3 * N^2/2$ 정도의 연산으로 해결할 수 있습니다. 이는 시간제한을 5배 이상의 여유를 두고 통과할 수 있습니다.

# F. 셋이 한 팀

`segtree`, `sweeping`
출제진 의도 – **Hard**

- 처음 푼 사람: **arnold518**, 131분
- 문제 아이디어: kdh9949
- 문제 세팅: kdh9949

## F. 셋이 한 팀

- 이제는 사람이 세 명으로 늘었습니다.
- C번에서 했던 접근을 좀 더 확장해 봅시다.
- $A_i \geq B_i, A_i \geq C_i$를 만족하는 $i$에 대해 $A_i$들을 더하고,
- $B_i > A_i, B_i \geq C_i$를 만족하는 $i$에 대해 $B_i$들을 더하고,
- $C_i > A_i, C_i > B_i$를 만족하는 $i$에 대해 $C_i$들을 더합니다.
- 부등호와 등호를 적절하게 배치해야 능력치 값이 같을 때 여러 번 더하는 경우가 없습니다.
- 위에 제시한 것이 이를 만족하는 하나의 예시입니다.

## F. 셋이 한 팀

- C번 설명과 유사하게, 각 쿼리 이후 A, B, C에 각각 더해진 값의 총합을 $T_A, T_B, T_C$ 라 합시다.
- 우선 $A_i$ 를 더해야 하는 경우 먼저 봅시다.
- $A_i + T_A \geq B_i + T_B, A_i + T_A \geq C_i + T_C$ 라는 조건은,
- $B_i - A_i \leq T_A - T_B, C_i - A_i \leq T_A - T_C$ 로 각각 쓸 수 있습니다.
- 2차원 평면을 그리고 각 $i$ 마다 $(B_i - A_i, C_i - A_i)$ 좌표에 점을 찍었다고 합시다.
- 조건을 만족하는 $i$ 들은 2차원 평면에서 특정 직사각형 영역에 속하는 점들에 해당합니다.

## F. 셋이 한 팀

- 각 쿼리마다 2차원 영역에 속하는 $i$들에 대해 $A_i + T_A$를 모두 더해야 합니다.
- 2차원 상에서 각 점에 가중치가 있다고 할 때, 특정 직사각형 안의 점 개수 및 가중치 합을 구하면 됩니다.
- 쿼리를 먼저 다 받은 뒤 오프라인으로 처리해도 되므로, Fenwick tree 등을 활용한 sweeping으로 구현하는 것이 가장 간편합니다. 총 $O((N+Q)\log N)$ 시간에 가능합니다.
- 똑같은 것을 $B$, $C$에 대해서도 각각 해 주면 팀의 능력치를 구할 수 있습니다.

# G. 조 나누기

`graphs`, `combinatorics`, `fft`

출제진 의도 – **Hard**

- ✓ 문제 아이디어: `moonrabbit2`
- ✓ 문제 세팅: `moonrabbit2`

## G. 조 나누기

- 2종 스털링 수를 구하는 방법들 중, 포함 배제를 통해 구하는 방법을 생각해 봅시다.
- $S(N, M) = \frac{1}{M!}\sum_{n=0}^{M}(-1)^n \binom{M}{n}(M-n)^N$ 입니다.
- 이 식의 원리는 우선 조들에 번호를 매긴 뒤, 반드시 비어있는 조의 수를 정하는 것입니다.
- $(M-n)^N$ 부분을 제외하면 이 문제에도 똑같이 적용할 수 있음을 알 수 있습니다.
- $(M-n)^N$ 대신 $n$개의 조로 나누는 경우의 수를 넣으면 됩니다. 그래프 채색 문제와 동일함을 알 수 있습니다.

## G. 조 나누기

- 채색다항식을 $F$ 라고 합시다.
- 각 $M$ 에 대해 답은 $\frac{1}{M!}\sum_{n=0}^{M}(-1)^n\binom{M}{n}F(M-n)$ 입니다.
- 정리하면 $\sum_{n=0}^{M}\frac{(-1)^n}{n!}\times\frac{F(M-n)}{(M-n)!}$ 입니다. Convolution 꼴이 나옴에 주목합시다.
- FFT를 통해 모든 $M$ 에 대한 답을 $O(N\log N)$ 에 구해줄 수 있습니다.

## G. 조 나누기

- 이제 채색다항식을 구해봅시다.
- $M$에 대한 채색다항식을 먼저 구해봅시다.
- 사이클에 포함되지 않은 정점들(트리 부분에 속함)은 부모 정점의 색을 제외한 $M-1$개의 색을 가질 수 있습니다.
- 크기 $C$의 사이클의 채색다항식은 $(x-1)^C+(-1)^C(x-1)$임이 알려져 있습니다. 포함 배제 등으로 어렵지 않게 도출할 수 있습니다.
- 모두 곱해주면 채색다항식 $F$를 구할 수 있습니다.

## G. 조 나누기

- 병목은 사이클들의 채색다항식들을 곱해주는 부분과 $F(1), F(2), \ldots, F(N)$을 계산하는 부분입니다.
- 분할 정복과 FFT를 사용하면 첫 번째 부분을 $O(N \log^2 N)$에 해결할 수 있습니다.
- 마찬가지로 분할 정복과 FFT를 통한 Multipoint evaluation을 사용하면 두 번째 부분을 $O(N \log^2 N)$에 해결할 수 있습니다.
- 따라서 문제를 $O(N \log^2 N)$에 해결할 수 있습니다.
- Challenge: $F$를 $O(N \log N)$에 구해보세요.

## G. 조 나누기

- 다른 방법도 있습니다. 사이클들의 크기의 합은 $N$ 이하임을 이용합시다.
- 서로 다른 사이클 크기의 개수는 $O(\sqrt{N})$ 개입니다.
- 하나의 $C$ 에 대해, 모든 $x$ 에 대해 답을 구하는 것은 $O(N \log N)$ 에 가능합니다.
- 따라서 문제를 $O(N\sqrt{N} \log N)$ 에 해결할 수 있습니다. 이 풀이로는 최적화가 잘 된 것이 아니면 통과하기 어렵습니다.

## G. 조 나누기

- 크기 $c$의 사이클의 개수를 $cnt_c$라고 합시다.
- $\log cnt_c$의 합은 $O(\sqrt{N})$임을 증명할 수 있습니다.
- 사이클 크기들 $C_1, \cdots, C_K$에 대해 $\sum_{i=1}^{K} \log(C_i - C_{i-1})$ 또한 $O(\sqrt{N})$임을 증명할 수 있습니다. ($C_0 = 0$)
- 하나의 사이클 크기 $c$에 대해 $0^c, \ldots, (N-1)^c$를 구하는 것은 소수는 $O(\log c)$에, 합성수는 약수들의 값을 이용해 $O(1)$에 구하면 소수 분포가 $O(\frac{N}{\log N})$이므로 $O(N)$입니다.
- 위 내용들을 사용해 적절히 최적화를 하면 $O(N\sqrt{N})$에 $F(1), \ldots, F(N)$을 구할 수 있습니다.