Good Bye, BOJ 2023!

Official Solutions



프리즈 시점까지

284명 참가

제출 **1672**회

AC **689**회, 해결된 문제 **6**개



총괄

- ✓ 김준겸 `ryute` — 고려대학교 컴퓨터학과
- ✓ 이종서 `leejseo` — KAIST 전산학부, Moloco

운영

- ✓ 김영현 `kipa00` — 서울대학교 컴퓨터공학부
- ✓ 정우경 `man_of_learning` — 전북대학교 컴퓨터공학부



출제

✓ 김영현	kipa00	서울대학교 컴퓨터공학부
✓ 곽우석	bubbler	SK 하이닉스
✓ 김기범	ibm2006	경기과학고등학교
✓ 김동현	kdh9949	Moloco
✓ 나정휘	jhnah917	숭실대학교 컴퓨터학부
✓ 오주원	kyo20111	숭실대학교 소프트웨어학부
✓ 이한길	wapas	중앙대학교 예술공학부

검수

- ✓ 김준서 junseo — 한양대학교 컴퓨터소프트웨어학부
- ✓ 모현 ahgus89 — 가톨릭대학교 의예과
- ✓ 박선재 cs71107 — 서울대학교 컴퓨터공학부
- ✓ 윤시우 cgiosy — 서울사이버대학교 컴퓨터공학과
- ✓ 정현서 jhwest2 — 서울대학교 컴퓨터공학부
- ✓ 최준석 stonejjun03 — 고려대학교

스트리밍

- ✓ 윤창기 TAMREF — 서울대학교 수리과학부

Sponsors





Sponsors

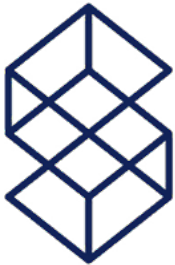






Sponsors

GCPC greetings 23/24



SAMSUNG
SOFTWARE
MEMBERSHIP

FURIOSA

Sponsors

HYUNDAI AutoEver

Presto

LG전자



문제		의도한 난이도	출제
A	2023은 무엇이 특별할까?	**Easy**	kipa00
B	거짓말	**Easy**	kyo20111
C	스티커 재배치	**Medium**	wapas
D	23E	**Medium**	kdh9949
E	전기 전송	**Medium**	kyo20111
F	등수	**Medium**	bubbler
G	오직 5%의 사람들만이 이 문제를 풀 수 있습니다	**Hard**	ibm2006

First to Solve (프리즈 시점 기준)

문제		처음 푼 사람	해결 시각
A	2023은 무엇이 특별할까?	ncy09	0분
B	거짓말	molamola	3분
C	스티커 재배치	molamola	14분
D	23E	GetRusty	17분
E	전기 전송	molamola	50분
F	등수	petamingks	41분
G	오직 5%의 사람들만이 이 문제를 풀 수 있습니다	?	?분

A. 2023은 무엇이 특별할까?

`arithmetic`
출제진 의도 – **Easy**

- ✓ 문제 아이디어: `kipa00`
- ✓ 문제 세팅: `jhnah917`

- ✓ $(N+1)\ \%\ (N\ \%\ 100) = 0$ 인지 확인하면 됩니다.

B. 거짓말

`prefix_sum`
출제진 의도 – **Easy**

- ✓ 문제 아이디어: kyo20111
- ✓ 문제 세팅: kyo20111

- K 명이 거짓말을 하는 상황이 가능한지 확인하는 문제를 생각해 봅시다.
- 거짓말하는 사람의 수가 정확히 K 가 될 수 있다면 K 는 정답에 포함됩니다.
- 각 사람의 주장이 거짓말이 되는 K 는 구간의 형태로 나타납니다.
- 따라서 누적 합 배열을 이용해

$$S_i := [\text{정답이 } i \text{ 일 때 거짓말하는 사람의 수}]$$

를 계산할 수 있습니다.

- $S_i = i$ 인 i 들이 정답이므로 전체 문제를 $\mathcal{O}(N)$ 시간에 해결할 수 있습니다.

C. 스티커 재배치

`bruteforcing`, `greedy`
출제진 의도 – **Medium**

- ✓ 문제 아이디어: wapas
- ✓ 문제 세팅: wapas

- 문자열 S에 존재하는 문자 $S[k]$와 같은 문자 s_j가 없으면 정답이 존재하지 않습니다.
- 그 외의 경우 항상 정답이 존재합니다.
- 스티커 문자열 U가 S를 부분 문자열로 포함하도록 만들어야 합니다.
- 즉, U의 구간 $[l, l+K)$이 S와 일치하는 l이 존재하도록 만들어야 합니다.
- 모든 $[l, l+K)$ 구간에 대해서 문자열 S를 만들 수 있는지 확인하는 방법을 알아봅시다.

C. 스티커 재배치

- ✓ Case 1. $S[k]$와 $U[l+k]$가 같은 경우
  - 그대로 두는 것이 항상 최소 비용입니다.
- ✓ Case 2. $S[k]$와 $U[l+k]$가 다른 경우
  - $l+k$ 위치에 붙여진 스티커를 반드시 떼야 합니다.
  - 빈칸에 스티커를 붙일 때 다음 3가지 종류의 스티커를 고려할 수 있습니다.
    - A 타입 : $[l, l+K)$ 구간에서 뗀 스티커 (비용 : 0)
    - B 타입 : $[l, l+K)$ 구간이 아닌 구간에서 뗀 스티커 (비용 : d_j)
    - C 타입 : 새로 구매한 스티커 (비용 : a_j)
  - $d_j, a_j > 0$이므로 가능하다면 A 타입의 스티커를 붙이는 것이 좋습니다.
  - A 타입의 스티커를 모두 소모했다면 B 타입과 C 타입 중 비용이 더 적은 스티커를 붙입니다.
  - B 타입의 스티커를 모두 소모했다면 C 타입의 스티커를 붙입니다.

- 시간 복잡도
  - $[l, l+K)$ 구간은 $N-K+1$ 개 존재합니다.
  - $[l, l+K)$ 구간의 길이는 K 입니다.
  - A, B, C 타입의 스티커 중 가장 비용이 적은 스티커를 찾는데 $\mathcal{O}(N)$ 시간이 소요됩니다.
  - 따라서 전체 시간 복잡도는 $\mathcal{O}((N-K+1) \times K \times N)) = \mathcal{O}\left(N^3\right)$ 입니다.

D. 23E

greedy

출제진 의도 – **Medium**

- ✓ 문제 아이디어: kdh9949
- ✓ 문제 세팅: kdh9949

- 우선, 문자열에서 180도 돌렸을 때 서로 "매칭되는" 위치끼리 짝지어 묶어 봅시다.
- S_1과 S_N, S_2와 S_{N-1}, ... 와 같이 묶음이 지어지게 될 것입니다.
- 각 묶음마다, 해당 묶음이 2개의 2, 또는 3과 E가 각각 1개씩으로 이루어져 있을 때 장식물의 아름다움이 2씩 늘어나게 됩니다.

D. 23E

- 논의를 편하게 하기 위해, 각 문자에 가중치를 주어 봅시다.
- 3에 -1, 2에 0, E에 1의 가중치를 각각 주었다고 합시다.
- 장식물의 아름다움은 $2\times$ (가중치 합이 0인 묶음의 개수) 입니다.

- 이제 교환을 적절히 수행해서 가중치 0인 묶음의 개수를 최대한 늘리면 됩니다.
- 가중치 0 묶음의 개수를 늘리는 교환은 아래의 3가지로 분류할 수 있습니다. (편의상 각 묶음의 가중치만을 표시하였습니다)
  1. -2, +2 $\rightarrow$ 0, 0 / -1, +1 $\rightarrow$ 0, 0
  2. -1, +2 $\rightarrow$ 0, +1 / +1, -2 $\rightarrow$ 0, -1
  3. -1, -1 $\rightarrow$ 0, -2 / +1, +1 $\rightarrow$ 0, +2
- (1)은 답을 4 증가시키고, (2)와 (3)은 답을 2 증가시킵니다.
- 매번 가능한 교환 중 답을 가장 많이 늘리는 것을 수행하는 그리디 전략을 생각해 봅시다.
- 다행히도 모든 k에 대해 그리디 전략이 최적임을 증명할 수 있습니다!

- 가중치가 -2, -1, 1, 2인 묶음의 개수를 각각 A, B, C, D라 합시다. 일반성을 잃지 않고 $A \leq D$라고 할 수 있습니다.
- 모든 묶음의 가중치 합을 S라고 합시다. 최종 상태 (가중치 0 묶음이 가장 많은 상태) 는
  - $S \geq 0$일 경우 2가 $\lfloor S/2 \rfloor$ 개, 1이 $S\%2$ 개인 상태
  - $S < 0$일 경우 -2가 $\lfloor -S/2 \rfloor$ 개, -1이 $-S\%2$ 개인 상태입니다.
- 그리디 전략이 최종 상태에 도달할 때까지의 모든 과정에서 최선의 답을 냄을 증명해 봅시다.

- Case 1: $B \leq C$.
- 이 경우에 그리디 전략은 (1)을 $A + B$번 수행하고, 그 뒤 (3)을 $\lfloor (C - B)/2 \rfloor$ 번 수행하여 최종 상태에 도달합니다.
- $k \leq A + B$ 일 경우에는 (1)을 k번 수행하는 것보다 더 좋은 경우가 없으므로 최적입니다.
- $A + B < k$ 일 경우에는, $A \leq D$ 및 $B \leq C$ 라는 조건 때문에 (2)나 (3)을 먼저 수행해도 (1)을 $A + B$번보다 더 수행할 수 없습니다. 따라서, 가능한 (1)을 모두 소진한 다음에 (3)을 가능한 한 반복하는 것이 최적입니다.

- Case 2: $B > C$.
- 이 경우에 그리디 전략은 (1)을 $A + C$번 수행하고, 그 다음 (2) → (1) 또는 (3) → (1)을 $\min(\lfloor (B - C)/2 \rfloor, D - A)$ 번 수행하고, 남은 묶음들을 가지고 (2) 또는 (3)을 수행하여 최종 상태에 도달합니다.
  - (2) → (1)은 묶음 3개를 -1, -1, 2 → -1, 0, 1 → 0, 0, 0 과 같이 변화시킵니다.
  - (3) → (1)은 묶음 3개를 -1, -1, 2 → -2, 0, 2 → 0, 0, 0 과 같이 변화시킵니다.

- $k \leq A + C$일 경우에는 역시 (1)만 수행하는 것이 최적입니다.
- $A + C < k$일 경우에는 (2)나 (3)을 한 번 수행하여 (1)의 수행 횟수를 한 번 늘릴 수 있습니다.
- 이때, $D - A$가 1 줄어들고 $B - C$가 2 줄어듭니다. 둘 중 하나라도 0 이하가 되는 순간 (1) 수행 횟수가 더 늘지 않기 때문에, 이는 최대 $\min(\lfloor (B - C)/2 \rfloor, D - A)$ 번까지만 가능합니다.
- 위 기회까지 모두 소진한 다음에는 (2)나 (3)을 가능한 한 반복하면 됩니다.

- 사실 위 증명에서 먼저 보여야 하는 사실이 하나 있는데, 바로 “가중치 0인 묶음이 늘어나지 않는 교환은 할 필요가 없다” 입니다.
- 우선, 가중치 0인 묶음 개수가 그대로인 교환은 묶음 가중치의 조성을 바꾸지 않으므로 할 필요가 없습니다.
- 현재 상태가 “최종 상태”가 아니면 무조건 답을 2 이상 증가시킬 수 있으므로, 답을 감소시키는 교환이 언젠가 더 이득이 되려면 해당 교환 한 번으로 (1) 수행 가능 횟수를 최소 3번은 늘려야 합니다.
- 답을 감소시키는 교환을 모두 고려해도, 많아야 2번까지만 늘릴 수 있습니다. 따라서 이 역시 할 필요가 없습니다.

- 그리디 전략의 구현은 매우 간단하게 $\mathcal{O}(N+K)$ 로 할 수 있습니다.
- 각 가중치별로 묶음이 몇 개 있는지를 관리하면, 매번 가능한 교환 중 가장 답을 많이 늘리는 교환을 $\mathcal{O}(1)$ 에 쉽게 찾을 수 있습니다.
- 그리디가 옳다는 확신을 빠르게 가지는 것이 중요한 문제였습니다.

E. 전기 전송

`trees`, `lca`

출제진 의도 – **Medium**

- ✓ 문제 아이디어: kyo20111
- ✓ 문제 세팅: kyo20111

- a에서 b로 가는 경로에 $z_i > 1$인 간선이 $\log e$개 이상 있으면 정답은 무조건 0이 됩니다.
- $z_i = 1$인 연속한 간선들을 한 번에 빠르게 처리할 수 있다면 문제를 해결할 수 있을 것입니다.
- $\mathrm{up}(x) := [x$번 정점에서 $z_i = 1$인 간선을 타고 올라갈 수 있는 가장 높은 정점$]$을 구합시다.
- DFS를 이용해서 $\mathcal{O}(N)$, 또는 Sparse Table을 이용해 $\mathcal{O}(N \log N)$에 구할 수 있습니다.

- ✓ 트리의 경로는 $l = LCA(a, b)$ 를 기준으로 $a \to l$ 와 $l \to b$ 로 나눠서 생각할 수 있습니다.
- ✓ $a \to l$ 경로를 지나는 동안 e 값의 변화를 빠르게 처리하는 방법을 알아봅시다.
- ✓ $z_i > 1$ 인 간선을 $\log e$ 번 지나면 항상 0이 되므로 $\log e$ 개 미만인 경로만 고려해도 됩니다.

- 최대 $\log e$ 개의 $z_i > 1$ 간선과 $z_i = 1$ 인 연속된 간선 덩어리의 나열 형태의 경로만 고려합시다.
- $z_i = 1$ 인 연속된 간선 덩어리는 $\mathrm{up}(x)$ 를 이용해 $\mathcal{O}(1)$ 에 처리할 수 있습니다.
- 따라서 쿼리는 LCA를 한 번 구한 뒤, $\mathcal{O}(\log e)$ 번의 상수 시간 연산으로 처리할 수 있습니다.
- 따라서 전체 문제를 $\mathcal{O}(N \log N + Q(\log N + \log e))$ 시간에 해결할 수 있습니다.

- ✓ 구현에 따라 $\mathcal{O}(N + Q(\log N + \log e))$, $\mathcal{O}(N + Q \log e)$ 등도 가능합니다.
- ✓ $z_i = 1$ 인 간선을 Union-Find 로 묶어서 구현하는 방식의 풀이도 존재합니다.

F. 등수

dp

출제진 의도 – **Medium**

- 문제 아이디어: bubbler
- 문제 세팅: bubbler

- 참가자가 n 명인 대회에서 금성이가 k 등을 한 경우의 수를 $a(n, k)$,
  참가자가 n 명인 대회에서 나올 수 있는 모든 결과의 경우의 수를 $b(n)$ 이라고 합시다.
- 금성이의 등수는 1 등에서 n 등까지 될 수 있으므로 $b(n) = \sum_{k=1}^{n} a(n, k)$ 입니다.

- 금성이가 1등이면, 다른 1등이 0명부터 $n-1$명까지 존재할 수 있습니다.
- 다른 1등이 i명이라고 가정하고 이들을 고정하면, 나머지 $n-i-1$명의 가능한 모든 결과는 $n-i-1$명만 참가한 대회의 결과의 모든 등수에 $i+1$을 더해서 얻을 수 있습니다.
- 이러한 경우의 수는 $b(n-i-1)$입니다.
- 따라서 $a(n,1)$은 각각의 i에 대해 $n-1$명 중 i명을 고르는 방법의 수와 $n-i-1$명의 대회 결과의 수를 곱한 것의 합이 되며, 수식으로 나타내면 다음과 같습니다.

$$a(n,1) = \sum_{i=0}^{n-1} \binom{n-1}{i} b(n-i-1)$$

- 금성이가 1등이 아니라면, 금성이보다 높은 등수를 갖는 참가자가 $k-1$ 명이 있고, 금성이를 포함한 나머지 참가자 중에서는 금성이가 1등인 상황이 됩니다.
- 따라서 $k \neq 1$ 일 때 $a(n,k)$ 는 금성이보다 높은 $(k-1)$ 명을 고르는 경우의 수, 그들의 대회 결과의 수, 그리고 금성이를 포함한 $(n-k+1)$ 명 중에서 금성이가 1등인 대회 결과의 수의 곱이 됩니다.

$$a(n,k) = b(k-1)\binom{n-1}{k-1}a(n-k+1,1)$$

- 이제 초깃값을 $b(0) = 1$로 두면 입력 범위 내의 모든 값을 재귀적으로 계산할 수 있습니다.
- 수식에 등장하는 이항 계수 역시 DP로 $\mathcal{O}(n^2)$에 계산해 놓을 수 있습니다.
- $n' < n$인 모든 n'에 대해 $a(n', k)$와 $b(n')$의 값을 알고 있을 때, 모든 $a(n, k)$와 $b(n)$을 구하기 위해 필요한 연산량은 다음과 같습니다.
  - $a(n, 1)$: $\mathcal{O}(n)$
  - $a(n, k)$ ($k \neq 1$): 각각 $\mathcal{O}(1)$, 총 $\mathcal{O}(n)$
  - $b(n)$: $\mathcal{O}(n)$
- 따라서 범위 내의 모든 $a(n, k)$를 구하기 위해 필요한 연산량은 $\mathcal{O}(n^2)$입니다.

G. 오직 5%의 사람들만이 이 문제를 풀 수 있습니다

`ad_hoc`, `constructive`
출제진 의도 – **Hard**

- ✓ 문제 아이디어: `ibm2006`
- ✓ 문제 세팅: `ibm2006`

- ✓ 일단, $k = 1$인 경우를 다루어야 하므로 공과 목표지점은 무조건 일직선상에 있어야 합니다.
- ✓ 정확히 2^x 번 움직이게 되는 비교적 작은 경로를 만들 수 있다면, 이 경로들을 잘 활용해서 문제를 풀 수 있을 것 같습니다.
- ✓ $k - 1$의 이진법 표현을 $a_m a_{m-1} \ldots a_0$으로 표기하겠습니다.

G. 오직 5%의 사람들만이 이 문제를 풀 수 있습니다

GCPC greetings 23/24

- 다음과 같은 배치를 고려해 봅시다:

앞	뒤
414	444
312	312
343	313
343	313
321	321

- 공을 맨 왼쪽 위에서 시작해서 다시 맨 왼쪽 위까지 오는 데에는 정확히 17번 소요됩니다.
- 만약 공이 처음부터 더 왼쪽에서부터 오고 있었다면 16번 소요됩니다.

- 공이 1행 2열 혹은 2행 2열에 있다면, 공은 그 아래에 있는 두 칸에 대해, 최초의 앞면까지 등장하는 모든 칸을 뒤집습니다. 만약 두 칸이 전부 뒷면이라면 탈출합니다.
- 이는 이진수에 1을 더하는 연산, 즉 맨 뒤에 있는 연속한 1들을 전부 0으로 뒤집고, 최초로 등장하는 0을 1로 뒤집는 연산에 대응됩니다.
- 공이 1행 2열 혹은 2행 2열에서 다시 2행 2열로 돌아오는 데에는 4턴이 지나야 합니다.
- 아래 두 칸에 대응되는 이진수가 $2^2 - 1$일 때 탈출하므로, 턴이 소요되는 횟수가 4×2^2가 되는 것입니다.
- 비슷한 원리로, 343/313층의 개수를 조절하면 4 이상의 2^x에 대해 유사한 회로를 만들 수 있습니다.

- ✓ 따라서 $k-1$의 하위 2비트는 $k-1$을 4로 나눈 나머지에 따라 예외 처리하고, 나머지 비트는 각 비트에 대응되는 2^x 회로를 활성화하면 됩니다.
- ✓ 4로 나눈 나머지에 대한 예외처리는 최대 2번의 수정으로 가능합니다.
- ✓ 이후 $k-1$의 각 비트마다 최대 한 번 수정하므로 최대 20번의 수정으로 문제를 풀 수 있습니다.

- ✓ 앞에서 언급한 2^x 회로를 $x = 20$에 대해 미리 만든 뒤, k가 주어지면 내부에 있는 이진수 표현을 미리 수정하는 풀이도 존재합니다.
- ✓ 이렇게 하면 임의의 4의 배수를 만들 수 있으므로, k를 4로 나눈 나머지에 따른 처리만 하면 됩니다.