Hello, BOJ 2023!

Official Solutions

92명 참가

프리즈 시점까지

제출 **350**회

AC **178**회, 해결 된 문제 **4**개

총괄

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

출제

- 박원 chunghan — UNIST 컴퓨터공학부
- 오주원 kyo20111 — 숭실대학교 소프트웨어학부
- 이종서 leejseo — KAIST 전산학부
- 정우경 man_of_learning — 전북대학교 컴퓨터공학부
- 정현서 jhwest2 — 서울대학교 컴퓨터공학부
- 최준석 stonejjun03 — 고려대학교
- 한동규 queued_q — UNIST 컴퓨터공학부

검수

김세린	serin	KAIST
김준겸	ryute	고려대학교 컴퓨터학과
김준서	junseo	한양대학교 컴퓨터소프트웨어학부
나정휘	jhnah917	숭실대학교 컴퓨터학부
노현서	gustjwkd1007	서울대학교 컴퓨터공학부
모현	ahgus89	가톨릭대학교 의예과

검수

- 박찬솔 chansol — 숭실대학교 컴퓨터학부
- 신기준 sharaelong — 서울대학교 컴퓨터공학부
- 윤시우 cgiosy — 서울사이버대학교 컴퓨터공학과
- 이성서 edenooo — 숭실대학교 컴퓨터학부
- 이성현 hibye1217 — 한양대학교 컴퓨터소프트웨어학부

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`

STARTLINK

Proudly Supported By:

leejseo.com

Sponsors

HYUNDAI MOBIS

한빛미디어 Hanbit Media, Inc.

HYUNDAI AutoEver

Sponsors

SAMSUNG SOFTWARE MEMBERSHIP

Presto

문제		의도한 난이도	출제
A	2023년이 기대되는 이유	**Easy**	leejseo, jhwest2
B	청소	**Easy**	kyo20111
C	카드 플러리쉬	**Medium**	queued_q
D	컵 쌓기	**Medium**	queued_q
E	신도시 개발	**Hard**	queued_q, chunghan
F	줄넘기	**Hard**	queued_q, man_of_learning
G	편지 배달 2	**Challenging**	queued_q
H	정밀지도 제작	**Challenging**	stonejjun03

First to Solve (프리즈 시점 기준)

문제		처음 푼 사람	해결 시각
A	2023년이 기대되는 이유	ainta	5분
B	청소	koosaga	10분
C	카드 플러리쉬	kclee2172	33분
D	컵 쌓기	existar	47분
E	신도시 개발	-	-분
F	줄넘기	-	-분
G	편지 배달 2	-	-분
H	정밀지도 제작	-	-분

A. 2023년이 기대되는 이유

`number_theory, brute_force`
출제진 의도 – **Easy**

- 프리즈 전까지 제출 109번, 정답 78명 (정답률 71.56%)
- 처음 푼 사람: **ainta**, 5분
- 문제 아이디어: `jhwest2`, `leejseo`
- 문제 세팅: `jhwest2`

A. 2023년이 기대되는 이유

- 10^9 이하의 자연수는 최대 10개의 자릿수로 이루어져 있으므로, 사이사이에 더하기 기호를 넣는 방법의 수가 많지 않습니다.
- m을 1부터 늘려가면서, 덧셈 기호를 2^9 가지 방법으로 끼워 보고 등식이 성립하는 경우가 있는지 확인해봅시다.
- m을 어디까지 확인하면 좋을까요?

- 만약 2 이상인 자릿수가 하나라도 있다면, $m = 30$ 만 보더라도 $2^{30} > 10^9$ 가 됩니다. 따라서 이 경우 30 이하의 m에 대해서만 확인해보아도 됩니다.
- 2 이상인 자릿수가 하나도 없다면 어떻게 될까요?
- 이 경우 각 자릿수는 0 또는 1 이므로, 모든 양의 정수 m 이 답이 됩니다.
- 예를 들어서, $n = 10110$ 인 경우 모든 양의 정수 m에 대해서 $1^m + 0^m + 1^m + 1^m + 0^m = 1 + 0 + 1 + 1 + 0$ 이 성립합니다.
- 따라서 Hello, BOJ 2023!을 출력하면 됩니다.

B. 청소

`sliding_window, tree_set`

출제진 의도 – **Easy**

- 프리즈 전까지 제출 122번, 정답 70명 (정답률 57.38%)
- 처음 푼 사람: **koosaga**, 10분
- 문제 아이디어: kyo20111
- 문제 세팅: kyo20111

- 연속한 구역을 구간이라고 표현하겠습니다.
- 이동 거리를 구해놓은 구간이 있다고 합시다.
- 이 구간에 오른쪽으로 인접한 구역이 추가된다면 이동 거리가 어떻게 변하는지 알아봅시다.
  - $[L, R] \to [L, R+1]$

- 추가된 구역의 우선순위가 구간 내에서 x번째로 크다고 했을 때

1. $x = 1$
   - 2번째로 큰 우선순위를 가진 구역과 추가된 구역을 연결하는 경로가 추가됩니다.
2. $x =$ 구간의 길이
   - $x - 1$번째로 큰 우선순위를 가진 구역과 추가된 구역을 연결하는 경로가 추가됩니다.
3. $1 < x <$ 구간의 길이
   - $x + 1$번째와 $x - 1$번째가 연결된 경로가 제거되고, 두 구역과 추가된 구역을 연결하는 경로가 각각 추가됩니다.

- 추가되는 경로와 제거되는 경로를 각각 이동 거리에서 빼고 더하면 구간을 늘린 이후의 이동 거리를 구할 수 있습니다.

- 구간 내의 가장 왼쪽 구역을 구간에서 제거한다면 이동 거리가 어떻게 변하는지 알아봅시다.
  - $[L, R] \to [L+1, R]$
- 가장 왼쪽 구역이 제거된 구간에서 왼쪽으로 인접한 구역을 추가한다고 생각하고 앞서 설명한 방법을 이용하면 이동 거리에 더해질 값을 알아낼 수 있고, 그만큼 이동 거리에서 빼주면 됩니다.

- $[1, K]$ 에서 시작해 앞에서 설명한 추가/제거를 $[N-K+1, N]$ 이 될 때까지 한 번씩 반복한다면 길이 K 의 구간을 모두 보기 때문에 정답을 구할 수 있는데,
  - $[1, K]$ 의 이동 거리는 $[1, 1]$ 에서 구간 내 가장 오른쪽 구역이 K 가 될 때까지 추가만 하면 구할 수 있습니다.
- $\mathcal{O}(N)$ 번의 삽입/삭제 연산과 삽입된 값에서 X 보다 큰 값 중 최솟값과 X 보다 작은 값 중 최댓값을 구하는 연산을 하게 됩니다.

- 그러므로 이 모든 연산을 빠른 시간에 할 수 있는 방법이 필요합니다.
  - C++ 와 같이 bbst를 지원하는 언어의 경우 bbst를 이용해 모든 연산을 $\mathcal{O}(\log K)$에 할 수 있습니다.
  - Python 등 bbst를 지원하지 않는 언어의 경우 세그먼트 트리나 제곱근 분할법을 이용해 최대 $\mathcal{O}(\log N)$ 혹은 $\mathcal{O}\left(\sqrt{N}\right)$에 할 수 있습니다.
- $\mathcal{O}(N \log K), \mathcal{O}(N \log N), \mathcal{O}\left(N\sqrt{N}\right)$ 세 가지 방법 모두 문제를 해결할 수 있습니다.

C. 카드 플러리쉬

constructive, ad_hoc
출제진 의도 – **Medium**

- 프리즈 전까지 제출 77번, 정답 18명 (정답률 23.38%)
- 처음 푼 사람: **kclee2172**, 33분
- 문제 아이디어: queued_q
- 문제 세팅: queued_q

- 설명의 편의를 위해, 원하는 순서로 섞는 대신 정렬하는 문제로 바꿔서 생각합시다.
- 설명에서 “i 번 카드” 대신 “최종 상태의 i 번째 카드”를 대입하면 원래 문제를 풀 수 있습니다.
- 또한 찰리어 컷은 묶음 하나의 크기가 0 인 트리플 컷으로 생각합니다.
- **핵심 관찰:** 맨 왼쪽에서부터 a 개의 카드 묶음을 생각합시다. 트리플 컷을 적절히 수행하면, 이 묶음의 왼쪽에 원하는 카드를 붙일 수 있습니다.
  - 원하는 카드의 위치가 b 일 때, a 와 b 위치에서 묶음을 나누는 트리플 컷을 수행하면 됩니다.
  - 그 결과로 a 개 카드 묶음은 맨 오른쪽으로 이동하고, 그 왼쪽에 b 번째 카드가 위치하게 됩니다.

풀이 1. (Original solution by queued_q)

- 정렬된 묶음을 왼쪽과 오른쪽에 번갈아 두면서 카드를 하나씩 붙여나갑니다.
- $N/2$번 카드에서 시작해서 묶음의 양 옆에 카드를 하나씩 붙여 나가면 N회가 됩니다.
- 그런데 사실 맨 처음에 $N/2$번 카드를 왼쪽 또는 오른쪽에 잘 배치해서, $N-1$번째 단계에 1번부터 $N-1$번까지의 묶음이 왼쪽에 가도록 만들면 $N-1$번만에 정렬됩니다.

풀이 2. (Based on chunghan's idea)

- 번호가 연속한 카드들을 하나의 묶음으로 생각하는 아이디어를 통해 해결할 수 있습니다.
- 가장 왼쪽의 카드가 i 번이라면, 이 묶음의 왼쪽에 $i-1$ 번으로 끝나는 묶음을 붙이는 트리플 컷을 수행합니다.
- 연산을 할 때마다 두 묶음이 합쳐져서 묶음의 수가 하나 줄어듭니다.
- 가장 왼쪽의 카드가 1 번인 경우를 처리하기 위해 N 번 다음 카드를 1 번으로 정의합니다.
- 최대 $N-2$ 번의 연산 후에는 두 개의 묶음이 남는데, 사실 이는 하나의 묶음입니다(!)
- 따라서 마지막으로 1 번 카드가 왼쪽에 오도록 찰리어 컷을 수행해 주면 됩니다.

두 풀이 모두 N 회를 $N-1$ 회로 줄이기 위한 디테일이 필요합니다.

D. 컵 쌓기

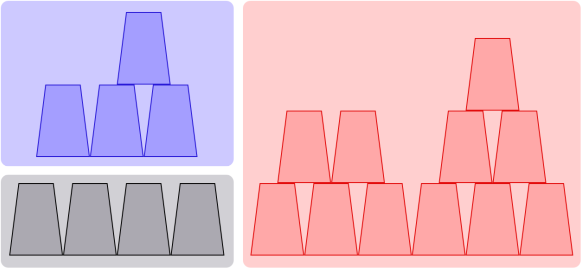
dp
출제진 의도 – **Medium**

- 프리즈 전까지 제출 32번, 정답 12명 (정답률 37.50%)
- 처음 푼 사람: **existar**, 47분
- 문제 아이디어: queued_q
- 문제 세팅: queued_q

- 조건에 맞게 컵을 쌓은 모양을 컵 스택이라고 합시다.
- 컵 a개를 쌓는데, 그 중 컵 i개를 1층에 쌓는 경우의 수를 셀 것입니다. 이렇게 쌓은 모양을 $C(a,i)$, 그 경우의 수를 $D(a,i)$로 표기합시다.
- 2층은 여러 개의 작은 컵 스택들로 구성됩니다.

- 1층 제일 왼쪽 두 컵의 바로 위에 있는 컵을 포함하는, 2층 제일 왼쪽에 있는 컵 스택을 생각합시다. 이 스택의 모양이 $C(b, j-1)$ 이라고 하겠습니다.
- 그러면 다음과 같이 전체 컵 스택을 세 부분으로 분리할 수 있습니다.
  - 2층 제일 왼쪽의 컵 스택 $C(b, j-1)$
  - 1층에서 이를 받치는 j 개의 컵
  - 그 외 나머지 $C(a-b-j, i-j)$
- 여기서 중요한 것은 임의의 $C(a, i)$ 에 대해 위와 같이 작은 컵 스택으로 분리하는 방법이 유일하며, 각각의 작은 컵 스택은 독립적으로 어떤 모양이든 가질 수 있다는 사실입니다.

- 그림으로 표현하면 다음과 같습니다. (표시한 색깔은 영역을 구분하기 위함이며, 컵의 색깔이 아닙니다.)



- 1층에 있는 j개의 컵이 2층의 $C(b, j-1)$을 받치려면, j개 컵 중 어떤 연속한 두 컵도 모두 파란 컵이 아니어야 합니다.
- 이렇게 배치하는 경우의 수는 피보나치 수 F_j와 같습니다. (초항은 $F_0 = 1, F_1 = 2$로 둡니다.)
- 정리하면, 다음 식을 만족합니다.

$$D(a,i) = \sum_{b<a} \sum_{j<i} D(b, j-1) D(a-b-j, i-j) F_j$$

- 다이나믹 프로그래밍을 통해 $O(N^4)$에 문제를 해결할 수 있습니다.

E. 신도시 개발

greedy, sliding_window
출제진 의도 – **Hard**

- 프리즈 전까지 제출 9번, 정답 0명 (정답률 0%)
- 처음 푼 사람: -, -분
- 문제 아이디어: queued_q, chunghan
- 문제 세팅: queued_q

- 용어를 간단하게 하기 위해 이미 분양된 M 개의 토지를 “기존 토지”, 새롭게 분양할 K 개의 토지를 “새 토지”, 두 토지를 통틀어 “분양 토지” 로 부르겠습니다.
- 어떤 시점에서 지금까지 분양된 전체 토지 수를 t, i 번 토지의 왼쪽에 있는 분양 토지 수를 l_i, i 번 토지의 오른쪽에 있는 분양 토지 수를 r_i 라고 합시다.

- 그러면 i 번 토지를 분양할 때 $|l_i - r_i| = t - 2\min(l_i, r_i)$ 만큼 할인된 가격으로 분양하게 됩니다.
- 새 토지 K 개를 배치하는 동안 t 는 M 부터 $M + K - 1$ 까지 1 씩 증가하므로, t 의 합은 고정된 값입니다.
- 따라서 우리의 목표는 새 토지들의 $\min(l_i, r_i)$ 의 합을 최대화하는 것입니다.
- 이것을 토지의 가치라고 부르겠습니다. 토지의 가치는 지금까지 분양한 다른 토지들에 따라 변할 수 있습니다.

- 이제 새 토지 K 개를 고정했을 때 최적의 분양 순서를 찾아봅시다.
- 이를 위해 토지를 분양하는 순서의 역순을 생각합니다.
- 즉, "이미 배치된 새 토지를 제거할 때의 가치" 합을 최대화할 것입니다.
- 어떤 토지를 제거할 때, 다른 새 토지들의 가치는 그대로 유지되거나 1 감소합니다.
- 따라서 새 토지들의 가치가 감소하지 않도록 제거 순서를 정한다면 가치의 총합이 최대가 됩니다.

- 최적의 순서는 정 중앙, $c = \left\lfloor \frac{M+K+1}{2} \right\rfloor$ 번째에 가까운 토지부터 제거하는 것입니다.
- 전체 분양 토지 중 $c, c+1, c-1, c+2, c-2, \ldots$ 번째의 토지를 확인하면서, 해당 토지를 제거 가능한 경우 제거합니다.
- 예를 들어 $i \le c$일 때 i번째 토지를 제거하는 순간을 생각합시다.
  - i를 c에 대해 대칭시킨 값을 j라고 하면, i번째 토지의 왼쪽과 j번째 토지의 오른쪽은 아직 토지가 제거된 적 없으므로 동일한 개수의 분양 토지가 있습니다.
  - 두 토지 사이에 있는 분양 토지까지 고려하면, i번째 토지의 가치는 $\min(l, r) = l$과 같습니다.
  - 이때 l은 감소한 적 없으므로 이 토지의 가치는 처음과 동일합니다.

- $i > c$인 경우 마찬가지 논리를 적용할 수 있습니다.
- 최적의 순서는 이밖에도 여러 가지 존재하며 그 조건을 구하는 것이 가능하지만, 이 문제를 풀기 위해서는 최적의 순서의 존재성만 필요합니다.

- 이제 어떤 토지 K 개를 고를지 찾아야 합니다.
- 중앙 (c번째) 토지의 위치 i부터 고정하고 그 왼쪽과 오른쪽에 적절한 수의 새 토지를 배치하는 방법으로 토지를 분양할 것입니다.
- 왼쪽에 있는 새 토지들만 고려했을 때, 최대 가치의 합 $L(i)$를 계산합시다.
- 새 토지들을 최대한 오른쪽으로 붙여야 $L(i)$가 최대가 됨을 알 수 있습니다.
- 마찬가지로 오른쪽만 고려했을 때 최대 가치의 합 $R(i)$를 계산하면, $L(i) + R(i)$가 최대인 i를 완전 탐색으로 찾을 수 있습니다.
- $L(i)$와 $R(i)$는 각각의 i에 대해 선형 시간에 계산할 수 있지만, 그러면 느립니다.

- 왠지 $L(i) + R(i)$ 가 볼록한 함수일 것 같지만, 그렇지 않습니다.
- 따라서 **삼분탐색 등의 풀이는 불가능합니다.**
- 대신 모든 i 에 대해 $L(i)$ 와 $R(i)$ 를 구해서 최솟값을 찾아줍시다.
- 모든 $L(i)$ 를 빠르게 계산하기 위해 슬라이딩 윈도우를 이용합시다.
- 왼쪽에 있는 새 토지들을 큐로 관리하면, 다음과 같이 $L(i)$ 의 값을 통해 $L(i+1)$ 의 값을 계산할 수 있습니다.

- 만약 $i+1$번 토지가 기존 토지라면, 왼쪽 분양 토지의 수를 c로 맞추기 위해 새 토지가 하나 줄어야 합니다.
  - 큐에서 제일 왼쪽 토지를 제거합니다. 해당 토지의 가치, 즉 해당 토지보다 왼쪽에 있는 기존 토지의 수만큼 가치의 합이 줄어듭니다.
  - 나머지 토지의 비용이 1 감소해야 하므로, 큐의 크기만큼 가치의 합이 줄어듭니다.

- 만약 $i + 1$번 토지가 빈 토지라면, 제일 왼쪽에 배치했던 새 토지를 이곳으로 옮겨야 합니다.
  - 위와 동일한 방법으로 큐에서 제일 왼쪽 토지를 제거합니다.
  - 큐의 오른쪽에 $i + 1$번 토지를 넣습니다. 해당 토지의 가치, 즉 해당 토지보다 왼쪽에 있는 기존 토지의 수와 큐의 크기만큼 가치의 합이 늘어납니다.

- 이와 같이 큐를 관리하며 모든 $L(i)$ 와 $R(i)$ 를 선형 시간에 계산할 수 있습니다. 큐 대신 포인터를 관리해도 됩니다.
- 답은

$$\frac{(2M+K-1)K}{2} - 2\max_i(L(i)+R(i))$$

- 시간복잡도는 $O(N)$ 입니다.

F. 줄넘기

geometry, line_intersection, sweeping
출제진 의도 – **Hard**

- 프리즈 전까지 제출 0번, 정답 0명 (정답률 -%)
- 처음 푼 사람: -, -분
- 문제 아이디어: queued_q
- 문제 세팅: man_of_learning

- 어떤 시점에서 평면 위의 점 P 는, 각 밧줄의 시계 방향/반시계 방향에 있는지를 0 과 1 로 나타낸 이진 문자열 $B(P)$ 로 표현할 수 있습니다.
- 한별이가 도착할 위치를 T 라고 하면, S 에서 T 로 이동하는 데 필요한 줄넘기 횟수는 적어도 $B(S)$ 와 $B(T)$ 의 서로 다른 비트의 수(해밍 거리) 이상이어야 합니다.
- 한별이가 S 에서 T 로 등속도로 이동한다면 그러한 최소 줄넘기 횟수를 달성할 수 있습니다.

- 이제 문제는 최종 상태에서 밧줄로 나눠진 모든 영역에 대해 대응되는 문자열을 구하고, $B(S)$와의 해밍 거리 중 최솟값을 구하는 문제가 됩니다.
- 모든 영역을 탐색하는 것은 어려우니 다음과 같은 아이디어를 사용합시다.
- 각 영역은 최소 하나의 밧줄과 인접하므로, 모든 영역을 탐색하는 대신 모든 밧줄에 대해 밧줄과 인접한 영역을 탐색하는 방법을 생각할 수 있습니다.

- 먼저 i 번 밧줄의 한쪽 끝(무한원점)에 점 T 를 두고, $B(T)$ 를 계산합니다.
- 이때, $B(T)$ 의 i 번째 비트는 밧줄의 시계 방향 영역이냐 반시계 방향 영역이냐에 따라 0 과 1 모두 선택할 수 있으므로, $B(S)$ 와 일치하도록 선택합시다.
- 밧줄의 반대쪽 끝을 향해 T 를 이동시키면서, 다른 밧줄을 만날 때마다 $B(T)$ 에서 해당하는 위치의 비트를 뒤집으면, i 번 밧줄에 접한 영역들의 문자열을 모두 구할 수 있습니다.

- 다른 밧줄을 만날 때마다 해밍 거리를 $\mathcal{O}(1)$ 에 갱신할 수 있습니다.
- 처음에 $B(T)$ 를 계산하는데 $\mathcal{O}(N)$, 다른 밧줄들을 만나는 순서를 정렬하는데 $\mathcal{O}(N \log N)$, 다른 밧줄들을 만나는 이벤트를 모두 처리하는데 $\mathcal{O}(N)$ 의 시간이 듭니다.
- 모든 밧줄에 대해 연산을 반복하면 총 $\mathcal{O}(N^2 \log N)$ 의 시간복잡도로 문제를 해결할 수 있습니다.

- **이론적으로 더 빠른 풀이도 존재합니다!**
- DCEL (doubly connected edge list) 자료구조를 사용하여 직선의 배열 (arrangement of lines)를 관리할 수 있습니다.
- Zone theorem에 의해, 자료구조에 직선을 하나 추가하는 데 $\mathcal{O}(N)$의 시간이 보장됩니다.
- N개의 직선을 추가하여 만든 arrangement of lines의 dual graph를 생각하면, 각 영역이 정점이고 인접한 영역끼리 간선으로 이어진 그래프가 됩니다.
- 모든 영역을 DFS/BFS로 탐색하면서, 앞의 풀이와 같이 $\mathcal{O}(1)$에 해밍 거리를 업데이트하면 답을 구할 수 있습니다.
- $\mathcal{O}(N^2)$에 문제를 해결할 수 있습니다.

G. 편지 배달 2

stack, pst

출제진 의도 – **Challenging**

- 프리즈 전까지 제출 0번, 정답 0명 (정답률 -%)
- 처음 푼 사람: -, -분
- 문제 아이디어: queued_q
- 문제 세팅: queued_q

- 모든 편지가 $a < b$를 만족한다고 가정합시다.
- $b < a$인 경우는 따로 모아서 대칭적으로 해결해 주면 됩니다.
- 동규가 매번 s반에서 e반으로 이동할 때 새롭게 전달되는 편지를 셉시다.

- $s < e$일 경우, 편지가 새롭게 전달되기 위해서는 다음과 같은 조건을 만족해야 합니다.
  - 편지가 아직 전달되지 않았어야 합니다. 즉, 동규가 a반을 마지막으로 지난 뒤에 방문한 가장 오른쪽 반 번호를 $f(a)$라고 할 때, $f(a) < b$를 만족해야 합니다.
  - 편지가 이제 전달되어야 합니다. 즉, $b \le e$를 만족해야 합니다.
  - 정리하면 $f(a) < b \le e$
- 동규의 현재 위치를 c라고 하면,
  - $x \le c$일 때 $f(x)$는 계단식으로 감소하는 함수이고
  - $x > c$일 때 $f(x) = x$입니다.

- $x \le c$에 대해 f가 감소하는 지점들과, 그때의 함숫값 (l_i, r_i) 목록을 관리합시다. 앞 슬라이드의 조건 $f(a) < b \le e$를 다시 정리하면 다음과 같습니다.
  - $a \ge s$인 경우: $s \le a < b \le e$
  - $a < s$인 경우: $l_i \le a < l_{i+1}$이고 $r_i < b \le e$
- 해당 조건을 만족하는 쌍 (a, b)의 개수는 PST(Persistent Segment Tree)를 사용해서 직사각형 영역들의 2차원 부분합을 계산하면 $O((\text{영역 개수}) \times \log N)$에 구할 수 있습니다.

- 이동하고 나면 함수 f를 업데이트해야 합니다.
- $f(x) \leq e$인 위치들의 함숫값이 e로 증가해야 하므로, 스택에서 $r_i \leq e$인 원소들을 제거한 뒤 마지막에는 원소를 제거하는 대신 r_i를 e로 업데이트하면 됩니다.

UCPC greetings 22/23

- $s > e$ 인 경우, 반대로 $b \le f(a)$ 를 만족하는 편지들만 새롭게 전달됩니다.
- 비슷한 방법으로 직사각형 영역들의 부분합을 통해 새롭게 전달되는 편지의 수를 계산할 수 있습니다.
- a 지점을 처음 방문하는 편지들은 세지 않도록 주의합시다.
- 함수 f 를 업데이트할 때는, 스택에서 $l_i \ge e$ 인 원소들을 제거한 뒤 (e, e) 를 넣으면 됩니다.

- 이와 같이 매번 이동하며 새롭게 전달되는 편지의 개수를 구해서 합하면 답을 구할 수 있습니다.
- $(N, L, M$ 구분 없이) 함수 f 를 나타내는 스택에서 원소를 뽑을 수 있는 횟수는 최대 $O(N)$ 번이고, 2차원 부분합을 구할 때마다 스택에서 원소를 뽑으므로, $O(N \log N)$ 의 시간이 걸립니다.
- PST 전처리도 마찬가지로 $O(N \log N)$ 이 걸리므로 전체 시간복잡도는 $O(N \log N)$ 입니다.

H. 정밀지도 제작

`graph_theory, binary_search, divide_and_conquer`

출제진 의도 – **Challenging**

- 프리즈 전까지 제출 1번, 정답 0명 (정답률 0%)
- 처음 푼 사람: -, -분
- 문제 아이디어: `stonejjun03`
- 문제 세팅: `stonejjun03`

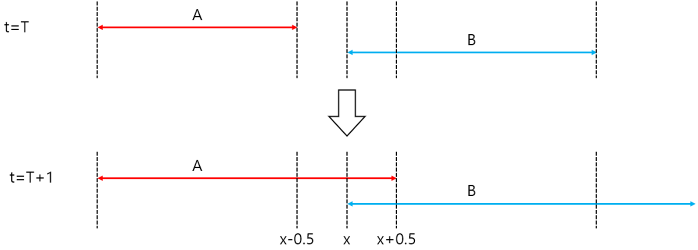
- 모든 도로들을 분석 시작 시간인 S_i 에 대해서 정렬을 합니다. (이후 서술하는 도로의 번호는 S_i 기준 정렬된 후의 번호로 가정하겠습니다.)
- 또한, 건물이 있는 모든 교차로가 연결되어 있는 즉, 데이터를 얻을 수 있는 지도를 가능한 그래프로 지칭하겠습니다.
- 첫 번째 목표는 모든 i 에 대해서 $i \sim X_i$ 번 간선이 가능한 그래프를 이루는 최소 X_i 를 모든 i에 대해서 구하는 것입니다.

X_i 를 구하는 방법에는 크게 두 가지가 있습니다.

- 첫 번째 방법은 divide and conquer을 이용하는 것입니다.
  - 도로가 시간에 대해서 정렬되어 있으며 지속시간이 동일하기 때문에 X_i 는 단조증가 한다는 것을 알 수 있습니다.
  - 이를 이용하여 s, e에 대해서 X_s, X_e 를 알고 있다면 $s < m < e$ 인 m에 대해서 X_m 을 구하기 위해서는 $[X_s, X_e]$ 구간만 탐색하면 됩니다.
  - dnc 아이디어를 사용하여 $m = (s + e)/2$ 로 잡고 탐색하는 과정에서 Union & Find를 이용하여 연결된 건물의 개수를 관리하면 이 과정 전체를 $\mathcal{O}(M \log M \log N)$ 에 해결할 수 있습니다.

- 두 번째 방법은 Queue Undo Technique를 이용하는 것입니다.
  - Union & Find 연산에서 Union 된 결과물에서 모든 건물이 연결되어 있는 지의 여부는 연산의 순서와 관계 없습니다.
  - 또한 모든 도로의 지속시간은 동일하며 S_i 가 모두 다르기 때문에 먼저 분석되기 시작한 도로가 먼저 분석할 수 없게 됩니다.
  - 따라서 가장 먼저 실행한 연산부터 롤백하는 과정을 amortized $\mathcal{O}(M \log M)$ 에 할 수 있게 해주는 Queue-Undo 의 아이디어를 사용해도 이 과정을 $\mathcal{O}(M \log M \log N)$ 에 해결할 수 있습니다.

- 이제 가능한 최소 t를 구해봅시다.
  - $t = T$에서 $t = T + 1$이 될 때 어떻게 되는지 관찰해봅시다.
  - 아래의 그림과 같이 A, B번 도로에 대해서 A번 도로의 분석 종료와 B번 도로의 분석 시작 간의 순서 관계가 변하는 경우가 생기게 됩니다.



- 이런 경우 분석이 가능했던 나머지 도로 집합 S가 존재할 때, $t = T$ 일 때 도로의 집합은 $\{S, A\} \rightarrow \{S\} \rightarrow \{S, B\}$으로 변했지만, $t = T + 1$로 바뀜에 따라 $\{S, A\} \rightarrow \{S, A, B\} \rightarrow \{S, B\}$ 으로 변하게 됩니다.
- t가 증가함에 따라서 도로의 집합에 도로가 추가되는 경우만 존재하기 때문에 '가능한 그래프'의 수는 t가 증가함에 따라서 단조증가하게 됩니다.
- 따라서 가능한 음이 아닌 최소 정수 t는 이분 탐색을 통해서 구할 수 있습니다.
- 이 과정은 가장 큰 K_i의 값 X에 대해서 $\mathcal{O}(N \log X)$가 걸리기 때문에 전체 문제를 $\mathcal{O}(N \log X + M \log M \log N)$에 해결할 수 있습니다.

- 추가적으로 이분 탐색 내에서 Offline Dynamic Connectivity 등을 실시하는 $\mathcal{O}(N \log X \log M \log N)$ 풀이는 통과하지 못하도록 시간을 설정하였습니다.