22

You, a minute ago · acmicpc.net

<h1>

Hello, BOJ 2022!

</h1>

Official Solutions

프리즈 시점까지

540명 참가

제출 **1,892**회
AC **441**회, WA **903**회, TLE **333**회, RE **140**회

총괄

- ✓ 이종서 `leejseo` Sendbird KAIST 전산학부

운영

- ✓ 김준겸 `ryute` 고려대학교 컴퓨터학과
- ✓ 박원 `chunghan` UNIST 컴퓨터공학과

출제

✓ 김준겸	ryute		고려대학교 컴퓨터학과
✓ 모현	ahgus89		가톨릭대학교 의예과
✓ 박선재	cs71107		서울대학교 컴퓨터공학부
✓ 박수현	shiftpsh	NEXON	서강대학교 컴퓨터공학과
✓ 이혜아	ho94949	[URGENT]	KAIST 수리과학과
✓ 한동규	queued_q		UNIST 컴퓨터공학과

검수

✓ 김세린	serin		KAIST 새내기과정학부
✓ 김준서	junseo		선린인터넷고등학교
✓ 김재환	jhhope1	하이퍼리즘	서울대학교 물리천문학부
✓ 나정휘	jhnah917		숭실대학교 컴퓨터학부
✓ 박원	chunghan		UNIST 컴퓨터공학과
✓ 서태수	cheetose		고려대학교 전기전자공학부

검수

✓ 배근우	functionx		고려대학교
✓ 오주원	kyo20111		숭실대학교 소프트웨어학부
✓ 윤지학	cubelover	NEXON	서울대학교 컴퓨터공학부
✓ 윤창기	TAMREF	AIRS medical	서울대학교 수리과학부
✓ 이종서	leejseo	Sendbird	KAIST 전산학부

스트리밍

- ✓ 윤창기 TAMREF — AIRS medical — 서울대학교 수리과학부

디자인

- ✓ 김현채 r4bb1t — Promedius — 고려대학교
- ✓ 박수현 shiftpsh — NEXON — 서강대학교 컴퓨터공학과
- ✓ 최희원 havana723 — 고려대학교 컴퓨터학과

조판

- ✓ 박수현 shiftpsh — NEXON — 서강대학교 컴퓨터공학과

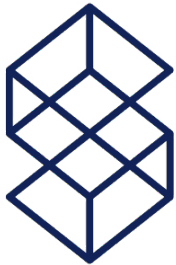

STARTLINK


SOLVED.
AC


NEXON


FURIOSA


HYUNDAI
AutoEver



SAMSUNG
SOFTWARE
MEMBERSHIP

NAVER D2

문제		의도한 난이도	출제
A	2022는 무엇이 특별할까?	**Easy**	ho94949, ryute
B	랜드마크 건설	**Easy**	ahgus89, ho94949
C	미니 버킷 리스트	**Medium**	havana723, ho94949
D	xor^2	**Medium**	shiftpsh
E	루트 노드가 많은 트리일수록 좋은 트리이다	**Hard**	shiftpsh
F	교통량 분석	**Hard**	ho94949, cs71107
G	구불구불 경주 트랙	**Hard**	queued_q
H	공정한 동전	**Challenging**	ho94949

A. 2022는 무엇이 특별할까?

`ad_hoc`, `math`
출제진 의도 – **Easy**

- 프리즈 전까지 제출 1,270번, 정답 279명 (정답률 22.835%)
- 처음 푼 사람: **dotorya**, 1분
- 문제 아이디어: ho94949
- 문제 세팅: `ryute`

- d 진수의 모든 숫자가 한 번씩 나와야 하므로, 모든 숫자의 순열 중 맨 앞이 0 으로 시작하지 않는 것을 모두 따져보면 됩니다.
- 시간복잡도는 $\mathcal{O}(d!)$ 입니다.

B. 랜드마크 건설

`ad_hoc`, `math`, `constructive`

출제진 의도 – **Easy**

- 프리즈 전까지 제출 290번, 정답 27명 (정답률 9.310%)
- 처음 푼 사람: **cki86201**, 12분
- 문제 아이디어: ahgus89, ho94949
- 문제 세팅: ahgus89, ho94949

세 변의 길이가 택시 거리로 a, b, c인 삼각형을 만들어야 합니다.
그러기 위해서는 아래 조건을 만족해야 합니다.

- ✓ $a \leq b + c$
- ✓ $b \leq c + a$
- ✓ $c \leq a + b$
- ✓ $a + b + c$가 짝수

처음 세 조건은 삼각부등식을 의미합니다.

한 점에서 다른 점으로 갈 때, 다른 점을 거치는 경우가 이동 거리가 더 짧을 수 없다는 것을 의미합니다.

다음 조건은 세 길이의 합이 짝수여야 한다는 조건입니다.

택시 좌표계에서, 거리 1 만큼 이동하면 $x+y$의 홀짝성이 바뀝니다.

한 점에서 출발해 다른 두 점을 지나 돌아오는 경로를 생각해봅시다.

총 이동거리는 $a+b+c$이고, 원래 점으로 돌아왔으므로 $x+y$의 홀짝성은 그대로입니다.
따라서 $a+b+c$가 짝수여야 합니다.

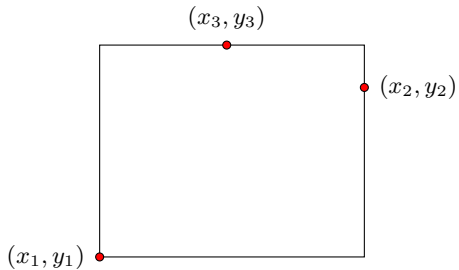
위 조건을 만족하면 항상 삼각형을 만들 수 있을까요?

위 조건을 만족하면 항상 삼각형을 만들 수 있을까요?

가능합니다.

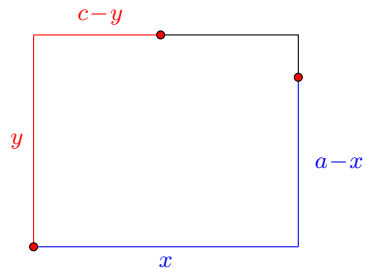
삼각형은 여러 가지 방법으로 만들 수 있습니다.

세 점을 아무렇게나 찍고, 세 점을 지나는 가장 작은 직사각형을 생각해봅시다.



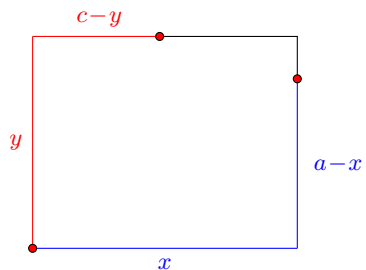
직사각형의 가로 길이를 x, 세로 길이를 y라고 합시다.

첫 번째 점과 두 번째 점 사이의 거리가 a, 첫 번째 점과 세 번째 점 사이의 거리가 c가 되도록 길이를 표시하면 다음과 같습니다.



두 번째 점과 세 번째 점 사이의 거리가 b가 되려면, $2x + 2y - a - c = b$를 만족해야 합니다.

따라서 $x + y = \frac{a + b + c}{2}$를 만족하면 됩니다.



또한 $a-x, c-y$는 음이 아닌 길이여야 합니다. 그러기 위해서는 b가 크지 않으면 된다는 것을 알 수 있습니다.

a, b, c 중 최솟값이 b인 경우에
$x = \left\lfloor \frac{a+b+c}{4} \right\rfloor, y = \left\lceil \frac{a+b+c}{4} \right\rceil$ 로 두면 항상 $a-x, c-y$는 0 이상이 됩니다.

$b \le a, b \le c, c \le a+b, a \le b+c$를 이용하면 증명할 수 있습니다.

a, b, c 중 최솟값이 a 나 c 라면, 왼쪽 아래에 두는 점을 각각 (x_3, y_3) 이나 (x_2, y_2) 로 두고 비슷한 방법으로 세 점을 배치할 수 있습니다.

출력하는 좌표가 1 이상 8×10^8 이하라는 제한이 있는데, $a = b = c = 10^9$ 인 경우에도 $x = y$ 가 최대 7.5×10^8 이므로 왼쪽 아래에 있는 점을 $(1, 1)$ 로 두면 항상 제한을 만족합니다.

따라서 항상 조건을 만족하는 삼각형을 만들 수 있습니다.

C. 미니 버킷 리스트

`math`, `ad_hoc`

출제진 의도 – **Medium**

- 프리즈 전까지 제출 256번, 정답 73명 (정답률 28.906%)
- 처음 푼 사람: **gs18103**, 12분
- 문제 아이디어: havana723, ho94949
- 문제 세팅: ho94949

- 길이가 K 인 공간에 서로 겹치지 않도록 N 개의 구간을 배치하는 문제입니다.

- 가장 간단한, 모든 구간의 길이가 1 인 경우를 생각 해 봅시다.
- 모든 구간의 시작 시간이 다르기만 하면, 배치는 올바릅니다.
- 가능한 시작시간으로 K 종류가 있고, 여기서 N 개를 중복 없이 골라야합니다.
- 가능한 경우의 수는 $P(K, N) = K \times (K-1) \times \cdots \times (K-N+1)$ 입니다.
- 우리는 **다른 방식**으로 해당 구간을 세어볼 것 입니다.

- 일의 순서와 구간 사이의 쉬는 시간을 정한다고 생각해 봅시다.
  - N 개의 일을 처리 할 순서를 정합니다.
  - 쉬는 시간으로 주어진 $K - N$ 만큼의 단위시간을, $N + 1$ 개의 연속된 시간으로 분할합니다.
- 이 방법으로도 모든 경우의 수를 셀 수 있고, 경우의 수는 역시 $P(K, N)$ 입니다.
- 주어진 쉬는시간인 X 와, 일의 수 N 에 관한 식으로 표현해 봅시다.
- $X = K - N$ 이라 할 때 경우의 수는 $(X + 1) \times (X + 2) \times \cdots \times (X + N)$ 입니다.

- 이제 구간의 길이가 주어졌습니다. 똑같은 방법으로 경우의 수를 세어봅시다.
  - N 개의 일을 처리 할 순서를 정합니다.
  - 쉬는 시간으로 주어진 $X = K - (S_1 + S_2 + \cdots + S_N)$ 만큼의 단위시간을, $N + 1$ 개의 연속된 시간으로 분할합니다.
- 경우의 수는 $(X + 1) \times (X + 2) \times \cdots \times (X + N)$ 입니다.

D. xor²

`trie`, `segtree`

출제진 의도 – **Medium**

- 프리즈 전까지 제출 48번, 정답 29명 (정답률 60.417%)
- 처음 푼 사람: **dotorya**, 19분
- 문제 아이디어: `shiftpsh`
- 문제 세팅: `shiftpsh`

인덱스에 XOR 연산을 취해야 합니다.
쿼리를 나이브하게 처리하기에는 업데이트 쿼리가 존재하며 구간이 너무 깁니다.

쿼리를 선형 시간 미만에 처리하고 싶습니다. 세그먼트 트리를 쓸 수 있을지 생각해 봅시다.

- ✓ 일반성을 잃지 않고 $0 \leq (i \oplus x) \leq r$ 쿼리만 생각하면 된다는 것을 보일 수 있습니다.
- ✓ $\oplus$의 항등원은 0이므로 $N = 262\,144 = 2^{18}$으로 두고 부족한 값을 전부 0으로 채운다고 생각합시다.

이제 적당한 r 과 x 에 대해 어떤 수들이 i 로 가능한지 확인해 봅시다. $r = 6, x = 1100_2$ 라고 두면 가능한 i 는

$$1100 \quad 1101 \quad 1110 \quad 1111$$
$$1000 \quad 1001 \quad 1010$$

이 존재합니다.

- 가능한 i는 자명하게 $r+1$개입니다($N = 2^{18}$로 생각하기로 했으므로).
- 가능한 i들을 다음과 같은 **뭉탱이**로 나눠볼 수 있습니다.

$$\begin{aligned} 11{**}&: \quad 1100 \quad 1101 \quad 1110 \quad 1111 \\ 100{*}&: \quad 1000 \quad 1001 \\ 1010&: \quad 1010 \end{aligned}$$

- 접두사가 특정 문자열인 모든 값의 합 쿼리가 가능한 자료구조는 **트라이**로 구현됩니다.
- 이진 문자열만이 존재하는 경우 이진 트라이가 되며, 이는 **세그먼트 트리**이기도 합니다.

언급한 완전 이진 세그먼트 트라이에서, 2^4의 자리에서 하나씩 내려오면서 생각해 봅시다.

$$r = 6 \qquad x = 1100_2$$

- ✓ 왼쪽 노드는 0***을, 오른쪽 노드는 1***을 포함합니다.
- ✓ 우리는 $r + 1 = 7$개의 값을 더 XOR해야 합니다. 이는 한 노드가 포함하는 값 $2^3 = 8$개보다 적습니다.
- ✓ 1100_2의 2^3의 자리가 1이므로, 왼쪽 노드를 무시하고 오른쪽 노드로 이동해 줍시다.

이번에는 2^3 의 자리에서 생각해 봅시다.

$$r = 6 \qquad x = 1100_2$$

- ✓ 왼쪽 노드는 10**을, 오른쪽 노드는 11**을 포함합니다.
- ✓ 우리는 $r + 1 = 7$개의 값을 더 XOR해야 합니다. 이는 한 노드가 포함하는 값 $2^2 = 4$개보다 많습니다.
- ✓ 1100_2 의 2^2 의 자리가 1 이므로, 11**은 전부 XOR해야 합니다.
  - 따라서 이 구간을 누적한 후 10**쪽 노드로 이동해 줍시다.
  - 4개를 XOR해 줬으므로 r 에서 4를 뺍시다.

하나 더 내려와서 2^2 의 자리에서 생각해 봅시다.

$$r = 2 \qquad x = 1100_2$$

- ✓ 왼쪽 노드는 $101*$을, 오른쪽 노드는 $100*$을 포함합니다.
- ✓ 우리는 $r + 1 = 3$ 개의 값을 더 XOR해야 합니다. 이는 한 노드가 포함하는 값 $2^2 = 2$ 개보다 많습니다.
- ✓ 1100_2 의 2^1 의 자리가 0 이므로, $100*$은 전부 XOR해야 합니다.
  - 이 구간을 누적한 후 $101*$쪽 노드로 이동해 줍시다.
  - 2 개를 XOR해 줬으므로 r 에서 2 를 뺍시다.

이전 슬라이드들과 같이, 완전 이진 세그먼트 트라이의 2^{18} 의 자리부터 내려오면서 2^i 번째 자리를 관리하는 노드에서

- ✓ x 의 2^{i-1} 번째 자릿수를 보고 이동할 노드를 결정하고
- ✓ r 과 자식 노드의 크기를 비교하면서 더할 노드를 결정하는 방법

으로 재귀적으로 문제를 해결할 수 있습니다.

E. 루트 노드가 많은 트리일수록 좋은 트리이다

`euler_tour_trick`, `segtree`, `offline_queries`, `lazy_propagation`
출제진 의도 – **Hard**

- 프리즈 전까지 제출 21번, 정답 19명 (정답률 90.476%)
- 처음 푼 사람: **ainta**, 40분
- 문제 아이디어: `shiftpsh`
- 문제 세팅: `shiftpsh`

8

다음 중 트리에 대한 설명으로 옳은 것은?

① 루트 노드가 많은 트리일수록 좋은 트리이다.
② 트리와 관련된 알고리즘을 재귀적인 방식으로 구현하면 실행 시간이 빨라진다.
③ 트리의 최대레벨과 트리의 높이와는 무관하다.
④ 트리의 노드 중 차수(degree)가 0인 노드를 리프(leaf) 노드라고 한다

(2010_7급 공개경쟁채용시험_ 자료구조론(고) 자료구조론)

BOJ의 밈으로 자리잡은 이 문장은 무려 2010년 **7급 공채**에 출제되었던 문제의 보기로 등장했던 문장입니다.
이 문제의 정답은 몇 번일까요?

일단은 쿼리들의 유형을 다음 단계로 쪼개 보겠습니다.

1. 무방향 간선일 때, 임의로 부여한 부모-자식 관계가 있다면 부모 노드에서 자식 노드 방향으로 방향성이 부여되는 쿼리
2. 무방향 간선일 때, 임의로 부여한 부모-자식 관계가 있다면 자식 노드에서 부모 노드 방향으로 방향성이 부여되는 쿼리
3. 방향이 있는 간선의 방향성을 제거하는 쿼리

가장 먼저, 트리의 모든 간선이 무방향 간선일 때 어떤 한 무방향 간선 $u \leftrightarrow v$의 방향을 정하는 쿼리를 생각해 봅시다.

$u \rightarrow v$로 간선을 정하면,

- ✓ v 쪽 노드들에서 u 쪽 노드들로 이동할 수 없게 됩니다.
- ✓ 따라서 v 쪽 노드들 전부가 루트 후보에서 빠지게 됩니다.
- ✓ u 쪽 노드들에서는 여전히 모든 노드로 이동할 수 있습니다.

이 쿼리를 빠르게 처리할 수 있는 방법이 있을까요?

생각하기 쉽도록 트리에 마음대로 부모와 자식이 존재하도록 '방향'을 부여해 봅시다(쿼리로 들어오는 방향과는 다릅니다!).

무방향 간선에 **부모에서 자식 방향으로** 방향을 부여하는 쿼리(1단계)만 들어온다면 다음과 같이 생각할 수 있습니다.

부모 u에서 자식 v로 방향이 부여되는 경우, v의 서브트리를 전부 지운다

오일러 투어 트릭을 수행하면 서브트리 쿼리를 구간 쿼리로 바꿀 수 있습니다.
서브트리 크기를 전처리하면, 오일러 투어 트릭으로 새로 붙은 노드 번호들에 대해 어떤 노드 v의 서브트리는 노드 번호가

$$[v, v + \text{size}(v))$$

에 들어오는 노드들을 포함합니다.

$u \to v$로 방향을 부여한다면 v의 서브트리를 지워 줘야 하므로, 초기에 모든 노드의 값을 1로 두고 $[v, v + \text{size}(v))$만 0으로 설정해 주는 쿼리로 충분합니다. 레이지 프로퍼게이션 등으로 어떻게든 해결할 수 있어 보입니다. (1단계 해결)

이번에는 무방향 간선에 **자식에서 부모 방향으로** 방향을 부여하는 쿼리(2단계)가 들어온다고 생각해 봅시다.

$v \to u$로 쿼리가 들어온다면 u 쪽 노드들을 전부 지워 줘야 합니다. 이는 v**의 서브트리만 제외하고** 전부 지워 줘야 된다는 말과 동치이며, 구간 쿼리로 환산한다면 전체 구간에서 특정 구간만 제외하고 0으로 설정하는 쿼리가 됩니다.

사실상 전체 구간을 두 개로 쪼개서 두 번 쿼리를 수행하면 됩니다. $[0, v)$와 $[v + \text{size}(v), N]$을 각각 모두 0으로 설정해 줍시다. (2단계 해결)

마지막으로 방향이 있는 간선의 방향을 지워 주는 쿼리입니다(3단계). 이 쿼리를 해결할 수 있다면 방향을 바꾸는 쿼리도 해결할 수 있게 됩니다.

방향이 있는 간선을 지우는 쿼리를, (방향이 없는 간선에 방향성을 부여하는 쿼리)를 실행 취소하는 작업이라고 생각해볼 수 있을까요?

초기에 모든 간선이 방향성이 없었다고 생각하고, 방향이 없는 간선에 방향성을 부여하는 쿼리가 **살아 있는 기간**을 고려해 봅시다. 예를 들어 3과 4를 잇는 간선이

- ✓ 초기에는 $3 \to 4$
- ✓ 4번째 쿼리에서 $3 \leftrightarrow 4$로 바뀜
- ✓ 6번째 쿼리에서 $3 \leftarrow 4$로 바뀜
- ✓ 7번째 쿼리에서 $3 \to 4$로 바뀜

의 과정으로 바뀌었다고 생각해 봅시다.

그러면,

- ✓ $3 \rightarrow 4$로 방향성을 부여하는 쿼리는 $[0, 3]$, $[7, Q]$에서 유효
- ✓ $3 \leftarrow 4$로 방향성을 부여하는 쿼리는 $[6, 6]$에서 유효

합니다.

이 접근을 골자로 여러 풀이가 존재합니다.

a) 선분의 합집합 길이를 활용한 방법.

BOI 2001 – Mars Maps (BOJ #3392 – 화성 지도)

주어지는 쿼리는 기본적으로 1이 칠해져 있는 전체 구간에 0을 칠하고, 1의 개수를 세는 문제입니다. 바꿔 생각하면 전체 구간의 길이는 고정이므로, 0의 개수를 세 주는 것으로도 문제를 해결할 수 있습니다.

이는 여러 선분의 합집합 길이를 동적으로 구하는 문제로 환원됩니다. 마찬가지로 $\mathcal{O}\left(Q \log Q \log N\right)$ 입니다.

b) 오프라인 동적 연결성 판정 문제와 유사한 방법.

CERC 2018 – The Bridge on the River Kawaii (BOJ #16695)

레이지 세그먼트 트리를 퍼시스턴트하게 만들면 쿼리를 $\mathcal{O}(1)$에 취소할 수 있습니다. 이 점에 집중했다면 쿼리들을 관리하는 세그먼트 트리를 만들고, 거기에 DFS하는 방법으로 각 쿼리의 답을 구할 수 있습니다.

사실 트리의 연산 형태가 단순해서 굳이 레이지 프로퍼게이션을 사용하지 않아도 됩니다. 직접 생각해보세요!

여담으로 첫 슬라이드에서 언급한 7급 공채 문제의 정답은 ④번이라고 합니다.

F. 교통량 분석

`dp`, `data_structure`, `tree`, `hld`

출제진 의도 – **Hard**

- 프리즈 전까지 제출 2번, 정답 ?명 (정답률 ?%)
- 처음 푼 사람: **?**, ?분
- 문제 아이디어: `ho94949`, `cs71107`
- 문제 세팅: `cs71107`

- 문제를 요약하면 다음과 같습니다.
  - 트리의 i 번째 간선은 x_i 와 y_i 를 연결하고, 음이 아닌 정수 A_i 가 쓰여있다.
  - 트리의 각 간선에 대해 $W_{x_i} + W_{y_i} \leq A_i$ 를 만족하도록
  - 트리의 정점에 음이 아닌 정수 W_j 를 적당히 배치할 때
  - $\sum W_j$ 의 최댓값을 구하여라.
- 쿼리는 나중에 생각하고, 주어진 트리에서 답을 구하는 방법을 생각해봅시다.

- 조금 생뚱맞지만, 트리의 minimum edge cover를 생각해봅시다.
- minimum edge cover는 모든 정점이 적어도 한 개의 간선과 인접하도록 트리의 간선 중 일부를 적절히 선택할 때, 그 가중치 합을 최소로 하는 간선들의 집합을 의미합니다.
- 이때, “minimum edge cover의 가중치와 $\sum W_j$ 의 최댓값이 같다”라는 사실이 성립합니다.

- $\sum W_j$ 가 minimum edge cover의 가중치 이하임은 쉽게 증명할 수 있습니다.
- 리프 정점부터 그리디하게 수를 배정하면 $\sum W_j$ 를 minimum edge cover의 가중치와 동일하게 만들 수 있습니다.
- 엄밀한 증명은 minimum edge cover의 형태와 몇 가지 케이스 분류를 통해 가능합니다.

- ✓ 쿼리가 주어지지 않는다면 Tree DP를 이용해 minimum edge cover를 구할 수 있습니다.
- ✓ 아래 두 가지 값을 관리하면 됩니다.
  1. v를 루트로 하는 서브 트리에서 minimum edge cover 가중치
  2. v를 루트로 하는 서브 트리에서 v를 제외한 정점들의 minimum edge cover 가중치
- ✓ 하지만 간선의 가중치를 바꾸는 쿼리가 주어지기 때문에 Tree DP를 동적으로 관리해야 합니다.
- ✓ 이러한 유형은 Heavy Light Decomposition으로 해결할 수 있음이 잘 알려져 있습니다.

- 정점은 체인에 속한 heavy edge에 의해 덮이거나, 해당 정점과 인접한 light edge에 의해 덮일 수 있습니다.
- 즉, 체인에 속한 정점을 처리할 때, 정점이 light edge에 의해 덮였을 때의 가중치와 덮이지 않았을 때의 가중치가 필요합니다.
- 그러므로 어떤 체인을 계산하기 전에, 해당 체인에 속한 정점과 인접한 light edge 아래에 있는 체인은 계산이 끝난 상태가 되어야 합니다.

- ✓ 세그먼트 트리를 이용해 체인을 어떻게 관리할지 알아봅시다.
- ✓ 세그먼트 트리의 각 정점은 체인의 l, $l+1$, $\cdots$, r 번째 정점에서의 아래 4가지 정보를 관리합니다.
  1. l, r 번 정점이 모두 덮이지 않고, $l+1$ 번 정점부터 $r-1$ 번 정점까지 모두 덮였을 때의 가중치
  2. l 번 정점은 덮이지 않고, $l+1$ 번 정점부터 r 번 정점까지 모두 덮였을 때의 가중치
  3. r 번 정점은 덮이지 않고, l 번 정점부터 $r-1$ 번 정점까지 모두 덮였을 때의 가중치
  4. l 번 정점부터 r 번 정점까지 모두 덮였을 때의 가중치
- ✓ $[l_a, r_a]$ 구간과 $[l_b, r_b]$ 구간은 r_a 번 정점과 l_b 번 정점을 잇는 heavy edge의 정보를 이용해 합칠 수 있습니다.

- Tree DP 과정에서 아래 두 가지 값을 알면 답을 구할 수 있습니다.
  1. v를 루트로 하는 서브 트리에서 minimum edge cover 가중치
  2. v를 루트로 하는 서브 트리에서 v를 제외한 정점들의 minimum edge cover 가중치
- 루트 정점의 결과가 문제의 정답이므로 체인의 가장 위에 있는 정점에 대해서만 관리해도 되고
- 이는 세그먼트 트리의 루트 정점에서 관리하는 값 중 2, 4번째 값에 대응됩니다.
- 세그먼트 트리를 이용해 관리하므로, 값이 변경되더라도 $O(\log N)$ 시간에 갱신할 수 있습니다.

- heavy edge를 모두 처리했으니, 이제 light edge로 연결된 정점을 처리해야 합니다.
- 정점이 light edge에 의해 덮였을 때의 가중치와 덮이지 않았을 때의 가중치를 구해야 합니다.
- light edge로 연결된 자식 정점은 자신이 속한 체인에서 가장 위에 있는 정점이므로
- 세그먼트 트리의 루트 정점과 light edge의 정보를 이용해 계산할 수 있습니다.
- 한 정점에서 light edge는 최대 $N-2$개 존재할 수 있는데, 세그먼트 트리를 이용해 light edge를 관리하면 $O(\log N)$에 갱신을 할 수 있습니다.

- 각 체인의 세그먼트 트리는 $O(N)$에 만들 수 있습니다.
- 간선의 가중치가 바뀔 때 영향을 받는 체인은 최대 $O(\log N)$ 개입니다.
- 각 체인을 갱신하는 것은 $O(\log N)$에 가능하므로 쿼리를 $O(\log^2 N)$에 처리할 수 있습니다.
- 그러므로 전체 시간 복잡도는 $O(N + Q \log^2 N)$ 입니다.

G. 구불구불 경주 트랙

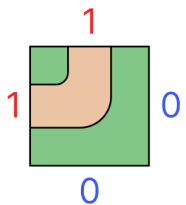
`bipartite_matching`

출제진 의도 – **Hard**

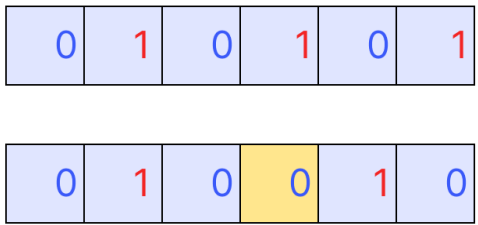
- 프리즈 전까지 제출 3번, 정답 ?명 (정답률 ?%)
- 처음 푼 사람: **?**, ?분
- 문제 아이디어: queued_q
- 문제 세팅: queued_q

- 다오가 타일을 먼저 놓고 나면 디지니는 항상 적절한 타일로 나머지 칸을 채울 수 있으므로, 디지니가 놓을 타일을 무시한 채 최대한 많은 타일을 놓으면 됩니다.
- 디지니의 구역을 빈 칸이라고 부르고, 디지니에게 공용 칸을 양보하는 것을 해당 칸을 비운다고 표현하겠습니다.
- 각 칸에 타일을 배정하는 문제가 아니라 각 타일과 타일 사이 선분에 색깔을 배정하는 문제로 생각합니다. 길이 연결되어 있지 않으면 색깔 0, 연결되어 있으면 색깔 1을 배정하겠습니다.
- 격자판 테두리와 다오가 이미 놓은 타일들의 경계에 색깔을 배정하고 시작합니다.

- 어떤 칸에 ㄱ자 타일이 놓였다는 것은 양 옆 세로 선분의 색깔이 서로 다르고, 위아래 가로 선분의 색깔이 다름을 의미합니다.
- 가로로 연속한 칸들 (이하 가로줄)을 다오의 타일로 채운다면 사이사이 선분들은 0, 1이 번갈아서 나타납니다.

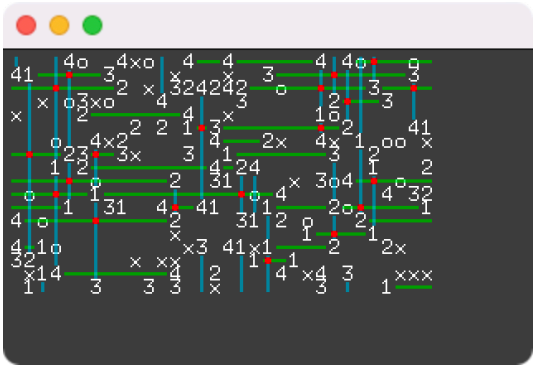


- 만약 가로줄 양 끝 선분의 값이 이미 정해져 있고, 홀짝이 안 맞아서 중간의 선분들을 칠할 수 없는 경우, 해당 가로줄에서 적어도 한 칸은 비워 두어야 합니다.
- 만약 홀짝이 안 맞는 가로/세로줄 중 전부 다오의 구역으로 이루어진 것이 있다면 경주 트랙을 완성하는 것이 불가능합니다. 다오의 구역은 비울 수 없기 때문입니다.



- 홀짝이 안 맞는 가로줄과 세로줄이 교차할 경우 교차 지점을 비우는 것이 최소한의 칸을 비우는 전략입니다. 가로줄과 세로줄을 따로 생각하면 두 칸을 비워야 하지만, 교차점을 비우면 한 칸만 비워도 되기 때문입니다.
- 단, 가로줄과 세로줄이 다오의 구역에서 교차한다면 해당 칸은 비울 수 없으므로 교차하지 않는 것으로 간주합니다.
- 각각의 홀짝이 안 맞는 가로줄과 세로줄을 정점으로, 교차 여부를 간선으로 생각하면 이분그래프를 만들 수 있습니다. 이분 매칭을 통해 비워야 하는 교차 지점의 최대 개수를 찾을 수 있고, 교차 지점을 비우지 않은 나머지 가로줄과 세로줄은 아무 위치나 비워 주면 됩니다.
- 전체 격자판에서 빈 칸과 비워야 하는 칸의 수를 빼 주면 답이 나옵니다.

- 랜덤 데이터 하나를 시각화한 그림 (빨간 점이 최대 이분 매칭)



H. 공정한 동전

`statistics`, `cht`, `divide_and_conquer`, `parametric_search`

출제진 의도 – **Challenging**

- 프리즈 전까지 제출 2번, 정답 ?명 (정답률 ?%)
- 처음 푼 사람: **?**, ?분
- 문제 아이디어: ho94949
- 문제 세팅: ho94949

- 구간으로 가능한 p-value 중에 가장 작은 값을 구하는 문제입니다.

- p-value를 구하는 함수인 $p(N, K)$는 다음과 같이 계산할 수 있습니다:
- $X \sim \mathcal{B}(N, 1/2)$인 Random Variable에 대해 $P(X = i) = \frac{\binom{N}{i}}{2^N}$이고,

$$p(N,K) = \begin{cases} 1 & N = 2K \\ 1 - \left[\sum_{i=\min(N-K,K)+1}^{\max(N-K,K)-1} P(X=i) \right] & N \neq 2K \end{cases}$$

- $P(X = i)$는 C++등에서 지원하는 `std::lgamma` 함수를 이용하면 다음과 같이 계산할 수 있습니다:
  - `exp(lgamma(N+1)-lgamma(i+1)-lgamma(N-i+1)-M_LN2*N)`

- 여기서, $X \sim \mathcal{B}(N, 1/2)$ 는 $\dot{X} \sim \mathcal{N}(N/2, N/4)$ 로 근사할 수 있습니다.
  - N 이 $\frac{0.05}{\epsilon}$ 정도에서 오차가 ϵ 이하가 됩니다.
- $P(X < K)$ 를 계산 할 때, $P(X < K - 0.5)$ 로 계산하는 연속성보정을 사용합니다.
  - 연속성보정을 사용하지 않으면 N 이 $\left(\frac{0.4}{\epsilon}\right)^2$ 정도가 되어야 오차가 ϵ 이하가 됩니다.
  - 이는 연속성보정이 CGF의 Taylor series의 1차항을 보정해주기 때문입니다.
- $P(X < K) = P(X < K - 0.5) \dot{\sim} P(\dot{X} < K - 0.5) = P(Z\sqrt{N/4} + N/2 < K - 0.5)$ 입니다. 여기서 $Z \sim \mathcal{N}(0, 1)$ 입니다.
- 식을 정리하면, $p(N, K) \approx P\left(|Z| > \frac{|2K - N| - 1}{\sqrt{N}}\right)$ 이고, 이는 $N \gg \frac{0.05}{\epsilon}$ 의 구간을 계산할 때 사용됩니다.

- $C = 2022, V = 100 \gg \frac{0.05}{C\epsilon}$ 라고 합시다.
- 길이가 V 미만인 구간에 대해서는 $\mathcal{O}(NV + CV^2)$ 정도의 시간으로, $|N/2 - K|$ 가 최대인 K 를 모든 N 에 대해 구한 후, $p(N, K)$ 를 계산해서 최솟값을 구해줍니다.
- 길이가 V 이상인 구간에 대해서는, $\frac{|2K - N| - 1}{\sqrt{N}}$ 의 최댓값 z 를 구한 이후에,
  $2 - 2\Phi(z) = \operatorname{erfc}\left(\frac{z}{\sqrt{2}}\right)$ 가 근사적인 답이 됩니다.

- 누적합 $S_x = \sum_{i=1}^{x}(A_i - C/2)$ 라고 합시다.
- i 이상 j 미만의 단계에 대한 $\frac{|2K-N|-1}{\sqrt{N}}$ 의 값은, $\frac{2|S_j - S_i|-1}{\sqrt{C(j-i)}}$ 이 됩니다.
- 우리는 이 값의 최댓값을 파라메트릭 서치와 분할정복으로 구해 나갈 것입니다.

- 우리는 $i, j \in [L, R+V]$ 이고, $j - i \geq V$ 인 (i, j) 쌍에 관심이 있습니다.
- $L = R$ 이면, 해당하는 (i, j) 는 $(L, R+V)$ 로 유일합니다.
- 아닌 경우 $L < M < R$ 을 만족하는 M 을 잡아서, $i, j \in [L, M+V]$ 이고, $i, j \in [M+1, R+V]$ 에 해당하는 답을 구했다고 해봅시다.
- 우리가 고려해야 할 나머지 경우는, $i \leq M$ 이고, $j \geq M+1+V$ 인 경우입니다.

- 연속성보정을 위해 사용한 -1은, 절댓값과 조합했을 때 생각보다 번거롭습니다.
- 우리는 다음과 같은 방식으로 절댓값 부호를 없애려고 합니다.
  - $i \leq M$ 이고 $j \geq M+1+V$ 일 때는, S_i와 S_j 사이에 $S_{M+1}, \cdots, S_{M+V}$ 모든 수가 들어가는 경우만 고려하면 됩니다.
    - 이 경우가 아니라면, 일반성을 잃지 않고, $S_i < S_j < S_k$ 이고 $i < k < j$인 k가 존재한다고 합시다.
    - $[i, j)$를 사용하는 것 보다, $[i, k)$를 사용하는 것이 전체 동전을 던진 횟수도 적으며, 중앙값과 차이도 크므로, p-value가 더 작습니다.

- 이제, $S_i < \min(S_{M+1}, \cdots, S_{M+V}) \leq \max(S_{M+1}, \cdots, S_{M+V}) < S_j$ 인 경우와, $S_i > \max(S_{M+1}, \cdots, S_{M+V}) \geq \min(S_{M+1}, \cdots, S_{M+V}) > S_j$ 인 경우가 있습니다.
- 일반성을 잃지 않고, 첫째 경우만 고려합니다. 이제 절댓값 기호를 없앨 수 있습니다.
- 또한, 여기서 파라메트릭 서치를 이용합니다.
- $\frac{2(S_j - S_i) - 1}{\sqrt{C(j-i)}} \geq z$ 인 쌍이 존재하는지 여부를 확인 해주면 됩니다.

- 양변을 제곱하면 $\frac{(2(S_j - S_i) - 1)^2}{C(j-i)} \geq z^2$ 이 됩니다.
- $D = Cz^2$ 을 포함해 식을 정리하면, $-4(2S_i + 1)S_j + ((2S_i + 1)^2 + Di) \geq Dj - 4S_j^2$ 인 (i, j) 쌍이 존재하는지를 묻는 문제입니다.
- 이는 S_j 에 대한 일차식 꼴이므로, 기울기가 $-4(2S_i + 1)$ 이고, y 절편이 $(2S_i + 1)^2 + Di$ 인 선분의 convex hull을 구해주고, $x = S_j$ 일 때 답을 $Dj - 4S_j^2$ 과 비교하는 방식으로 문제를 해결할 수 있습니다.

- ✓ 분할정복을 하면서 정렬이 되어 있는 $S_L, \cdots, S_R$ 과 $S_{L+V}, \cdots, S_{R+V}$ 을 관리해 준다면
  - Convex hull은 스택을 이용해서 $\mathcal{O}(R-L)$ 시간에 구할 수 있습니다.
  - $x = S_j$ 일 때의 답은 2-pointer를 이용해서 $\mathcal{O}(R-L)$ 시간에 구할 수 있습니다.
  - 두 배열을 병합해서 정렬하는데 $\mathcal{O}(R-L)$ 시간이 듭니다.
- ✓ 어떤 $[L, R+V]$ 구간에 대한 답을 구하는 데에는, $T(R-L) = 2T\left(\frac{R-L}{2}\right) + \mathcal{O}(R-L)$ 시간이 듭니다.
- ✓ $T(R-L) = \mathcal{O}((R-L)\log(R-L))$ 입니다.

- ✓ 파라메트릭 서치를 사용했기 때문에, 총 시간은 $\mathcal{O}(N \log N \log(1/\epsilon))$ 시간이 듭니다.
- ✓ 문제를 총 $\mathcal{O}\left(\frac{1}{C} \cdot \left(\frac{N}{\epsilon} + \frac{1}{\epsilon^2}\right) + N \log N \log\left(\frac{1}{\epsilon}\right)\right)$ 시간에 해결할 수 있습니다.