IamCoder 2024 입부시험 풀이

쉬웠죠?

by

2024 나코더 입부 출제진

문제		의도한 난이도	출제자
A	매우 어려운 문제	**Easy**	한재민
B	모기잡이	**Easy**	윤승하
C	Split the SSHS 3	**Medium**	이서환
D	출제는 지루해!!	**Medium**	윤승하
E	도망친게 아니라, 빛이 드는 곳으로 갔을 뿐이야	**Hard**	한재민
F	Grand Escape	**Hard**	이유찬
G	Antifreeze	**Challenging**	이유찬
H	Love is war	**Challenging**	이유찬

A. 매우 어려운 문제

필요한 개념: 없음
예상 난이도: **Bronze I**

- 제출 78회, 정답 27명
- 처음 푼 사람: 변재우, 3분 33초
- 출제자: 한재민
- 윌슨 정리를 이용하여 잘 푸셨나요?

부분문제 1 (10점)

- ✓ $N = 3, M = 5$
- ✓ 출제 의도: 팩토리얼 연산을 이해하고 있는가, 입출력 함수를 알고 있는가?
- ✓ 직접 계산해보면 결과가 1로, 단순히 1을 출력해주면 됩니다.

부분문제 2 (10점)

- ✓ $N \leq 10000, M \leq 1000$
- ✓ 출제 의도: for 문을 알고 있는가?
- ✓ 직접 for문을 이용하여 1 부터 N 을 곱하고, M 으로 나눠주는 과정을 반복하면 됩니다.

부분문제 3 (10점)

- ✓ N, M은 소수, $N \leq 10^6$, $M \leq 10^6$
- ✓ 출제 의도: long long 자료형을 알고 있는가?
- ✓ 부분문제 2와 거의 같습니다.
- ✓ 문제의 노트에서 언급하였듯, long long 자료형을 이용하면 풀 수 있습니다.

부분문제 4 (70점)

- ✓ 추가 제약 조건 없음.
- ✓ 출제 의도: 주어진 연산의 특성을 찾을 수 있는가?
- ✓ $M \leq N$ 일 때는 1 부터 N 까지의 곱에 M 이 존재하므로 0 을 출력하면 됩니다.
- ✓ $M > N$ 일 때는 부분문제 3과 동일하게 해도 시간 초과가 나지 않습니다.
- ✓ $O(\min(N, M))$ 내에 풀 수 있습니다.

B. 모기잡이

필요한 개념: `bruteforce` , `implementation`
예상 난이도: **Silver I**

- ✓ 제출 215회, 정답 14명
- ✓ 처음 푼 사람: 변재우, 12분 4초
- ✓ 출제자: 윤승하
- ✓ 재삼쌤과 상수쌤께는 비밀로 해주세요. 솔직히 경곽 최고의 대머리는 재삼쌤

B. 모기잡이

I'm

- ✓ 완전탐색 + 구현 + STL
- ✓ next_permutation() 을 배워봐요

- 재민이부터 해결해봅시다
- 모기를 잡을 순서를 정해준다면, 재민이가 다음 모기로 넘어가는데 걸리는 시간은 택시거리입니다.
- 가능한 모든 모기 순서를 돌리기 때문에 $O(K \times K!)$ 입니다.
- 이렇게 하면 같은 자리에 모기가 여러마리 있는 케이스도 한번에 처리됩니다.
- 백트래킹을 해서 구현해도 되지만, STL 함수를 사용하면 간단합니다.

- next_ permutation() 함수는 배열에서 다음 순열을 호출합니다.
- 다음 순열이 없다면 false를 리턴합니다.
- 맨 처음 벡터에 1부터 K를 순서대로 대입합니다.
- do_ while(next_ permutation(v.begin() , v.end())
- 를 하면 2줄만에 모든 순열을 도는 코드를 짤 수 있습니다.

- ✓ 유찬이는 어떻게 할까요?
- ✓ $N \times M$ 위의 모든 격자칸에 대해서 전기장을 형성해보면 됩니다.
- ✓ $O(N \times M \times K)$ 에 해결할 수 있습니다.

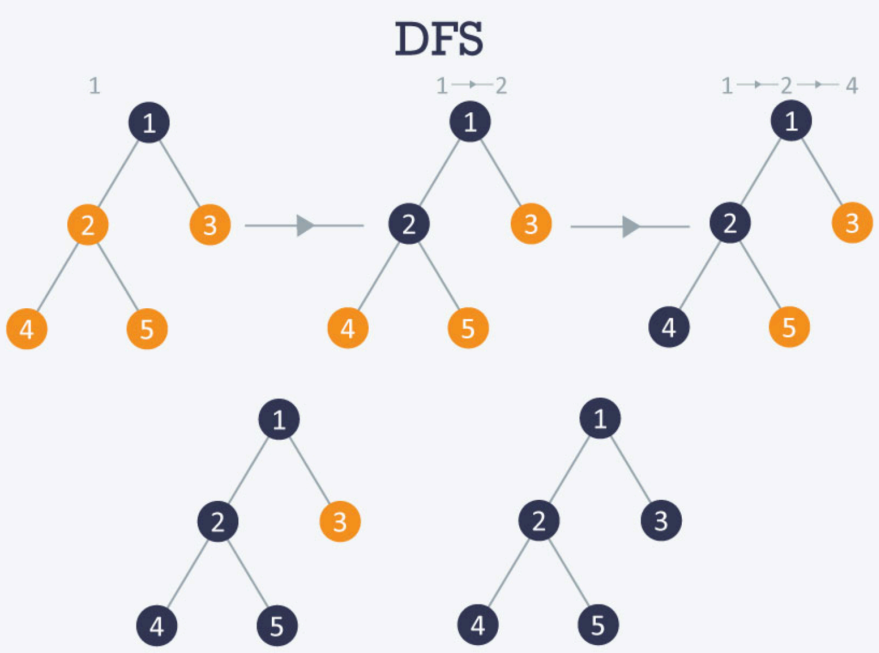
C. Split the SSHS 3

필요한 개념: `dfs`, `dp`, `bruteforce`
예상 난이도: **GOLD IV**

- ✓ 제출 75회, 정답 11명
- ✓ 처음 푼 사람: 변재우, 15분 38초 (죄송합니다.)
- ✓ 출제자: 이서환
- ✓ 죄송합니다
- ✓ 트리 몰라도 풀만하지 않았나요?

필요한 사전 지식 : DFS

- ✓ 깊이 우선 탐색
- ✓ 한 방향으로 끝까지 파고들고, 다시 돌아와서 가장 가까운 갈림길에서 반복
- ✓ 그림으로 이해하기!


DFS
1
1→2
1→2→4
1
2
3
4
5

I'm

필요한 사전 지식 : DP

- ✓ 같은 상황을 여러 번 만나는 경우, 그 상황에 대한 값을 정의
- ✓ 상황에 대한 상태 정의가 중요함!
- ✓ 여러 번 계산할 필요가 없어서 시간 단축
- ✓ EX) $F_n = F_{n-1} - F_{n-2}$
  F_{n-1} 을 구할 때 F_{n-2} 필요 ⇒ 저장해놓으면 한 번만 구해도 됨

I'm

부분문제1(17점)

- ✓ $N \leq 5000$
- ✓ dfs + bruteforce

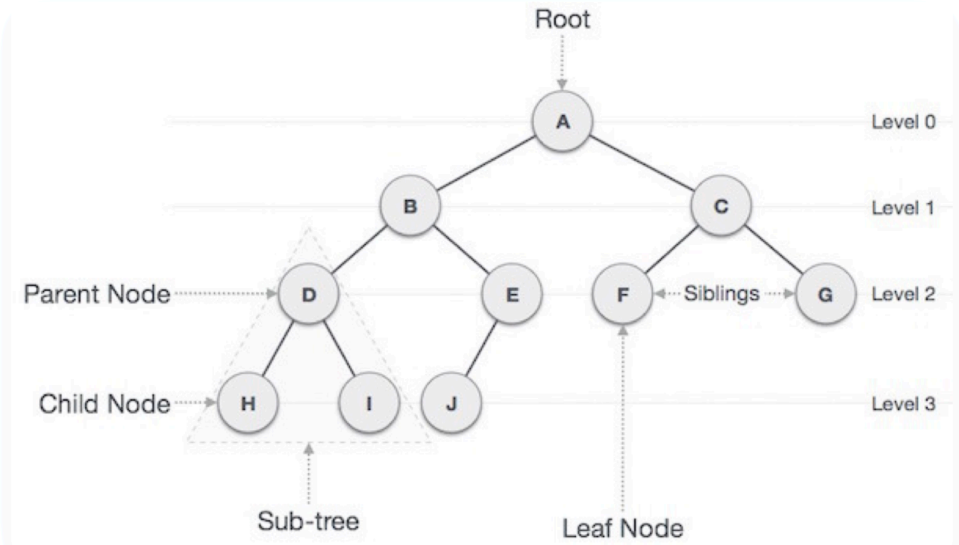
I'm

부분문제2(17점)

- ✓ prefix sum
- ✓ $S_n = S_{n-1} + A_n$
- ✓ 1번 k번, $k+1$번 N번 으로 나눠진 숲들의 가중치 차이는 $(S_{tot} - S_n) - S_n$
- ✓ 각 위치마다 S_n 을 구해주면서 $S_{tot} - 2S_n$ 최솟값 구하기

용어 설명

- ✓ 트리 : 특이한 형태의 그래프, 주어진 나무들과 나무줄기들이 만드는 그래프
- ✓ 노드 : 대나무
- ✓ 간선 : 나무 줄기
- ✓ 깊이 : 0부터 시작
- ✓ 루트 노드 : 깊이가 0인 노드(오직 하나!)
- ✓ 서브트리 : 임의의 간선을 끊었을 때 생성되는 트리


Root
A
Level 0
B
C
Level 1
Parent Node
D
E
F
Siblings
G
Level 2
Child Node
H
I
J
Level 3
Sub-tree
Leaf Node

I'm

부분문제 3(66점)

- ✓ 트리 dp
- ✓ $S_n : n$을 루트 노드로 하는 서브트리들의 가중치 합
- ✓ $|S_{tot} - 2S_n|$ 가능한 n을 모두 해보자!
- ✓ $O(n)$

D. 출제는 지루해!!

필요한 개념: `sorting` , `binary _ search`
예상 난이도: **Gold III-I**

- ✓ 제출 89회, 정답 5명
- ✓ 처음 푼 사람: 문승현, 73분
- ✓ 출제자: 윤승하
- ✓ 출제자를 2주동안 고통받게 한 문제
- ✓ 난이도 조절하느라 힘들었음

- ✓ 선분들을 정렬하고 이분탐색할거에요
- ✓ lower_bound()와 upper_bound()를 배워봐요

- $x = a$에서 시작해 $x = b$에서 끝나는 선분을 (a, b)라고 표기합시다.
- (a, b) 선분을 4종류로 분류해봅시다.
- Type1 : $a \leq S \quad and \quad b \geq E$
- Type2 : $a \leq S \quad and \quad b < E$
- Type3 : $S < a \quad and \quad a \leq E \quad and \quad b \geq E$
- Type4 : $S < a \quad and \quad b < E$

- Type1 선분이 하나라도 있다면 자명히 답은 0입니다.
- 이제 2개의 선분으로 덮는 경우를 생각해볼까요?
- Type2와 Type3 선분을 이용할 수 있습니다.
- $v1$: Type2 선분의 끝점을 저장한 벡터
- $v2$: Type3 선분의 시작점을 저장한 벡터
- $v1$과 $v2$는 각각 오름차순으로 정렬합니다.

- ✓ $x = b$에서 끝나는 Type2 선분이 있다고 해봅시다.
- ✓ 2개의 선분으로 덮는 최적의 경우는 $x = a$에서 시작하는 선분 중 $a \leq b$를 만족하는 a가 최대인 선분입니다.
- ✓ 이러한 a를 빨리 찾는 것은 $v2$ 벡터에서 이분탐색을 통해 $O(logN)$에 찾을 수 있습니다.
- ✓ STL을 이용하면 간단하게 구현할 수 있습니다.
- ✓ upper_bound() 함수는 자신보다 큰 최초의 원소가 등장하는 위치를 찾아줍니다. 이를 idx라고 합시다.
- ✓ (idx-1)은 찾고자 하는 값보다 작거나 같은 최초의 원소가 등장하는 위치입니다. 이러한 값이 없을 때의 예외처리를 해주어야 합니다.

- 이제 선분 3개로 덮는 경우를 생각해봅시다.
- 선분 3개로 덮으려면 Type2, Type3, Type4 선분을 1개씩 이용해야 합니다.
- Type4 선분 기준으로 찾아주면 편합니다. (a, b) 인 Type4 선분이 있다고 합시다.
- 우리가 찾아야 할 것은, $x = a$ 보다 늦게(같거나) 끝나는 Type2 선분과 $x = b$ 보다 일찍(같거나) 끝나는 Type3 선분을 찾아야 합니다.
- 이는 v1에서 lower _bound() 연산과 v2에서 upper _bound() 연산을 활용하면 $O(logN)$ 복잡도에 찾을 수 있습니다.

- 3개 선분이 모두 겹치는 구간은 어떻게 처리할까요?
- 3개 선분이 모두 겹치는 구간은 문제의 정의상 3번 계산해야 합니다.
- 하지만 잘 생각해보면, 3개 겹치는 구간은 항상 2개 겹치는 구간으로 바꿀 수 있고, 이렇게 하는게 이득임을 알 수 있습니다.
- 즉, 3개 선분이 모두 겹쳐서 덮는 경우는 최적해가 될 수 없습니다.
- 따라서 2개 선분을 덮는 경우에서 이미 처리가 되었음을 알 수 있습니다.

E. 도망친게 아니라, 빛이 드는 곳으로 갔을 뿐이야

필요한 개념: 없음
예상 난이도: **Platinum III**

- ✓ 제출 67회, 정답 0명, 최고점 80점
- ✓ 최고점에 가장 빨리 도달한 사람: 조은호, 154분
- ✓ 출제자: 한재민
- ✓ 문제에서 힌트를 줬습니다. "도망치세요"
- ✓ 근데 솔직히 쉬웠다.

시작하기 전에

- ✓ 만약 문제가 버림이었다면, 여러분은 매우 쉽게 푸셨을 겁니다.
- ✓ 근데... 버림과 반올림은 구간의 크기가 같은데요?

부분문제 1 (2점)

- ✓ $p = 2, r = 0$
- ✓ 출제 의도: 예외 조건을 파악할 수 있는가?
- ✓ 프로그램이 끝나지 않는 유일한 상황입니다.
- ✓ 모든 수는 언젠가 1에 도달할 것이고, 다음 시행에는 0.5가 되므로 반올림 했을 때 다시 1이 됩니다.
- ✓ 따라서 q와 관계 없이 절대 끝나지 않습니다.
- ✓ 0을 출력하면 됩니다.

I'm

부분문제 2 (7점)

- ✓ $p^q \leq 10^6, r = 0$
- ✓ 출제 의도: 모든 경우를 탐색하는 코드를 짤 수 있는가?
- ✓ 1 부터 p^q 까지의 모든 수를 문제의 순서도대로 판단하면 됩니다.
- ✓ 부분문제 1의 경우는 예외처리 해줍니다.

부분문제 3 (16점)

- ✓ $p = 3, r = 0$
- ✓ 출제 의도: p가 제한 되어 있는 상황에서 규칙을 찾을 수 있는가?
- ✓ 뒤에서 정해는 설명할 것이니, 단순히 규칙을 관찰해봅시다.

E. 도망친게 아니라, 빛이 드는 곳으로 갔을 뿐이야

I'm

- 3 미만의 수 중 0 이 되는 수는 1 , 2 두개입니다.
- 3 이상 9 미만의 수 중 0 이 되는 수는 4, 5, 7 세 개입니다. (초항)
- 9 이상 27 미만의 수 중 1 회 시행만에 4 를 만들 수 있는 수는 11, 13 입니다.
- 5 를 만들 수 있는 수는 14, 16 입니다.
- 7 을 만들 수 있는 수는 20, 22 입니다.
- 따라서 총 6 개입니다.

- 같은 방법으로, 27 이상 81 미만의 수 중 0 이 되는 수는 12 개 입니다.
- **따라서, 거듭제곱 구간마다 2배씩 수의 개수가 늘어남을 알 수 있습니다.**
- 다음 부분 문제에서 그 이유를 설명합니다.

I'm

부분문제 4 (25점), 부분문제 5 (30점)

- ✓ $r = 0$
- ✓ 출제 의도: 일반적인 상황에서 규칙을 증명할 수 있는가?
- ✓ 이젠 위 규칙이 성립하는 이유를 모르면 예외 찾기가 까다로울 겁니다.
- ✓ 이유를 설명해보죠.

- ✓ $p = 10$ 일 때를 가정합시다.
- ✓ 1 회 시행 후 1729 가 되는 수는 몇개일까요?
- ✓ 17285 부터 17294 까지 10 개일 것입니다.
- ✓ 다만, 17290 은 제외하면 9 개입니다.

- ✓ **다시 관찰하면, 우리는 1회 시행 후 각 n 자리수에 대해, 그 수가 되는 수의 개수는 9개이며, 그 모든 수는 $n+1$ 자리임을 알 수 있습니다.**
- ✓ 이때, 우리는 거듭제곱 구간마다 수의 개수를 셀 것입니다.
- ✓ 즉, 1 이상 10^q 미만의 수는 사실 q 자리 미만의 모든 수를 물어보는 것입니다.
- ✓ 위의 규칙을 적용하면 n 자릿수에서 0 이 되는 수의 개수는 $n-1$ 자릿수에서 0 이 되는 수의 개수보다 9 배 많음을 알 수 있습니다.

- 예외는 어떨 때일까요?
- 만들어질 수가 $10\cdots0$ 일때가 예외입니다. 이 경우, $99\cdots95$ 부터 $99\cdots9$ 가 9개의 수에 포함되며, 이들은 n 자리입니다.
- 그러나 $r=0$ 일때, $10^k(k\geq1)$ 꼴은 다음 시행에서 10^{k-1} 이 되고 프로그램이 종료되며, 따라서 예외로써의 역할을 하지 못합니다.
- **따라서 $k=0$ 일 때 (한 자릿수) 만이 예외입니다.**
- $p=10$ 일 때 기준으로 설명하면 한자릿수는 1 부터 9 가 전부 0 이 되며, 두자릿수는 1 회 시행 후 1 이 될 수 있는 수가 11 부터 14 까지 4 개뿐입니다.
- 수형도를 직접 그려보시면 직관적으로 이해되실 겁니다.

- p가 다르면 어떡하냐고요?
- 직관적으로는 이해하시겠지만, 엄밀하게는 **동일한 논리를 p 진법에서 적용하면 됩니다.**
- 코드를 짜실 때는 짝수와 홀수를 구분해서 생각해보세요.

(참고) dp형식을 이용한 풀이

부분문제 4 (25점)

- ✓ 앞에 부분을 이해하셨다면 바로 부분문제 6으로 넘어가셔도 됩니다.
- ✓ 사실 이 풀이가 접근하기는 더 쉬웠을 것이라 생각합니다.
- ✓ 정보를 많이 하신 분이라면 더 익숙할 것이기 때문입니다.

- ✓ dp를 설정하겠습니다.
- ✓ 기본적으로 p^q 이하를 물어봤으니, p로 나눈 나머지인 x, p의 지수에 들어갈 q 이하의 y를 이용하여 2차원 dp를 만듭니다.
- ✓ dp[x][y]와 같이 설정하겠습니다.
- ✓ dp[x][y]는 p진법으로 나타내었을 때, 끝자리가 x이고, 자릿수가 y인 수 (중 조건을 만족하는 수)의 개수를 나타냅니다.

- ✓ p로 나눈 나머지가 0일 때는 어떨까요? 당연히 $r = 0$이 될 수 없을 것입니다.
- ✓ 따라서 dp[0][y]$= 0$입니다.
- ✓ 만약 p로 나누었을 때, 반올림이 없다면 어떨까요? 그 수는 분명 $y - 1$자리의 수 일것입니다.
- ✓ 따라서, x가 반올림이 되지 않는 나머지일 때,
  dp[x][y]$=$dp[0][y]$+$dp[1][y]$+\cdots+$dp[$p-1$][y]입니다.

E. 도망친게 아니라, 빛이 드는 곳으로 갔을 뿐이야

I'm

- 만약 p로 나누었을 때, 반올림이 있다면 어떨까요? 다음 자리를 기준으로 분석해봅시다.
- p진법 표현이 $\ldots.ax_{(p)}$라 합시다. 이 수는 후에 $\ldots.(a+1)_{(p)}$을 만듭니다.
- 따라서 $a=0$인 수의 개수는 dp[1][$y-1$], $a=1$인 수의 개수는 dp[2][$y-1$]$\cdots$와 같이 표현됩니다.
- 다만, $a=p-1$인 수는 $\ldots.p_{(p)}$가 되어 연쇄적인 반올림이 일어나며, 이때, 자리 올림이 일어날 수 있으므로 수의 개수는 dp[0][$y-1$]+10^{y-1} 가능 여부+10^{y-2} 가능 여부일 것입니다.
- 이렇게 설정하면, $O(pq)$ 내에 해결할 수 있습니다.

부분문제 5 (30점)

- ✓ 이 형태는 식정리만 해준다면 끝납니다.
- ✓ $O(p)$ 내에 해결할 수 있습니다.
- ✓ 지금 보니 배점이 너무 높네요.

(참고) 끝

부분문제 6 (10점), 부분문제 7 (10점)

- ✓ 추가 제약 조건 없음.
- ✓ 출제 의도: 동일한 논리를 확장하여 생각할 수 있는가?
- ✓ 앞의 논리를 그대로 적용하면 되나, 규칙성만 이용해서 푸셨다면 접근하기 매우 힘드셨을 겁니다.
- ✓ 그래도 착하게 배점 줄여놨어요.

- $p = 10$일 때, 1회 시행하여 1729로 끝내는 수는 무엇일까요?
- 그런 수는 17290 하나 뿐입니다.
- 따라서 r을 p진법으로 표현하고, 끝에 0을 붙인 수를 만드는 것이 목표가 됩니다.
- 이후 과정은 동일합니다.
- 즉, 17290을 만드는 수는 역시 9개의 여섯 자리 수일 것이며, 그 9개의 수는 다시 각각 일곱 자리인 9개의 수로부터 만들어질 것이며... 이를 반복하게 됩니다.
- 그럼 예외는 없을까요?

- ✓ 아쉽게도 있습니다.
- ✓ 그런데 사실상 부분문제 4, 5와 동일한 논리의 예외입니다.
- ✓ 예를 들어, 10000은 직접적인 예외가 됩니다.
- ✓ 왜냐하면, 10000은 결국 4개의 여섯 자리 수 뿐만 아니라, 5개의 다섯 자리 수로부터 만들어지기 때문입니다. (99995부터 99999)
- ✓ 결국, 이는 원래와 달리 다섯 자리 수가 6개가 되며, 여섯 자리 수의 개수도 조금 달라질 것입니다.
- ✓ 즉, r이 p^s 꼴인 경우는 예외 처리를 해줘야 합니다. (p진법으로 $10\cdots0$인 경우)

- ✓ 이 예외처리를 완벽하게 한다면 문제가 풀립니다.
- ✓ 주의하실 점은 p^q **미만** (이하 아님) 이라는 점입니다.
- ✓ 즉, 추가적인 예외처리를 하시면 안됩니다.
- ✓ 수고하셨습니다. (죄송합니다.)

F. Grand Escape

필요한 개념: `offline-query`, `sweeping`, `segment-tree`, `coordinate-decompression`
예상 난이도: **Platinum III**

- ✓ 제출 45회, 정답 5명
- ✓ 처음 푼 사람: 조은호, 89분
- ✓ 출제자: 이유찬
- ✓ RADWIMPS - Grand Escape (feat. Toko Miura)

문제 분석

- ✓ 세그먼트 트리, 스위핑, 좌표압축을 쓰는 전형적인 자료구조 문제였습니다.
- ✓ 예상대로, E번 문제에 비해 훨씬 많은 분들이 풀어주셨습니다.

부분문제 4 ($x \leq 50$)

- ✓ 이 문제에서 가장 핵심적인 관찰 2개를 해보겠습니다.
- ✓ 어떤 사람이 (Px, Py)에 있고, 그 사람이 닿는 벽의 y 좌표가 순서대로 $\{y_1, y_2, ..., y_n\}$ 이면, 그 사람의 답은 $Py + y_1 + y_2 + ... + y_n$ 입니다.
- ✓ 모든 사람에 대한 답을 입력 순서대로 답할 필요가 없습니다. 어떤 순서든 상관없습니다.

부분문제 4 ($x \leq 50$)

- ✓ 사람과 벽을 모두 y 좌표가 증가하는 순서대로 정렬합시다. 어떤 가로선이 $+y$ 방향으로 이동한다고 생각합시다.
- ✓ 또한 크기 50 (x 좌표의 범위) 인 배열 arr 을 하나 관리합니다.
- ✓ 어떤 벽 $(Lx1..Lx2, Ly)$ 에 닿으면, $arr[Lx1..Lx2]$ 구간에 Ly만큼 더합시다.
- ✓ 어떤 사람 (Px, Py) 에 닿으면, 그 사람의 답은 $arr[Px] + Py$ 입니다.
- ✓ 이렇게 하면 $O((N+M)log(N+M) + Nx + M)$ 시간에 문제를 해결할 수 있습니다.
- ✓ 이러한 알고리즘을 **스위핑** 알고리즘이라고 하고, 정말 많이 쓰입니다.

I'm

부분문제 5 ($x \leq 100,000$**)**

- **세그먼트 트리** : 배열의 특정 위치에 수를 더하는 **update 연산**, 배열의 특정 구간$[l..r]$의 합을 구하는 **query 연산**을 둘다 $O(logN)$에 시행할 수 있는 잘 알려진 자료구조입니다.
- **느리게 갱신되는 세그먼트 트리** : 배열의 특정 구간$[l..r]$에 수를 더하는 **update 연산**, 배열의 [l..r] 구간의 합을 구하는 **query 연산**을 둘다 $O(logN)$에 시행할 수 있는 잘 알려진 자료구조입니다.
- 느리게 갱신되는 세그먼트 트리를 쓰면 부분문제 4 풀이를 그대로 대입해 풀 수 있습니다.
- **느리게 갱신되는 세그먼트 트리를 쓰지 않고도 문제를 해결할 수 있습니다.** arr 대신 arr의 차이$(arr[i] - arr[i-1])$에 해당하는 배열 $darr$을 만듭시다. 이러한 방법을 **누적 합**이라고 합니다.
- $arr[l..r]$**+=**$x \leftrightarrow darr[l]$**+=**$x$**,** $darr[r+1]$**-=**x
- $arr[i] = darr[1] + darr[2] + ... + darr[i]$

만점 풀이

- ✓ 기존 풀이를 그대로 적용하면, 배열의 크기가 너무 커집니다. (x 좌표의 범위)
- ✓ 하지만 모든 x 좌표를 쓰지 않습니다. 실제로 쓰는 x 좌표는 $O(N+M)$ 개밖에 되지 않습니다 (sparse함).
- ✓ **좌표 압축**: 실제로 쓰는 좌표만 뽑아서 정렬시킨 뒤, 이분 탐색 등으로 압축된 좌표를 구할 수 있습니다.

G. Antifreeze

필요한 개념: `dp-tree`, `union-find`, `priority queue`, `smaller-to-larger`
예상 난이도: **Diamond IV**

- ✓ 제출 13회, 정답 0명, 최고점 45점
- ✓ 최고점에 가장 빨리 도달한 사람: 변재우, 85분
- ✓ 출제자: 이유찬
- ✓ 검정치마 - Antifreeze

문제 분석

- ✓ 자료구조와 테크닉을 상당히 많이 요구하지만, 전형적인 문제였습니다.
- ✓ **트리 dp에서 Smaller to Larger**를 하는 문제 유형입니다. 다이아 이상의 올림피아드 문제에서 자주 보이는 유형입니다.
- ✓ **Union-Find 자료구조**는 정말 중요합니다. 만점 풀이를 몰라도, Union-Find 를 왜 쓰는지 정도는 이해해주셨으면 합니다.

I'm

부분문제 4 (난방 기구의 개수 $\leq 1,000$)

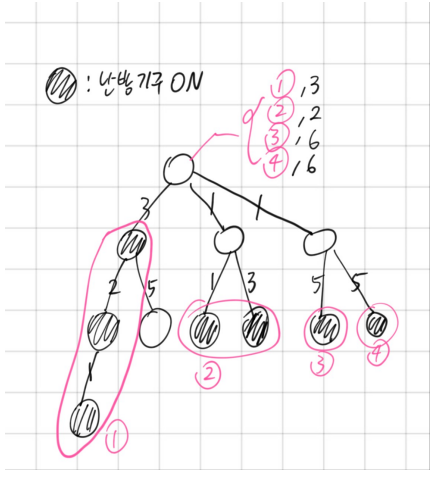
- ✓ 교실 A에서 B로 갈 수 있다면, 교실 B에서 A로도 갈 수 있습니다.
- ✓ 따라서 **"서로 갈 수 있는 교실의 집합"** 은 독립된 집합을 이룹니다. 더 쉽게 말하면, **교실을 서로 이동할 수 있는 그룹으로 나눌 수 있습니다.**
- ✓ **Union find (Disjoint Set Union)** : 독립된 집합을 다루는 자료구조
- ✓ Find(x) : x가 속해 있는 그룹을 구함
- ✓ Union(x,y) : x가 속해 있는 그룹과 y가 속해 있는 그룹을 합침
- ✓ 각각의 연산은 $O(logn)$ 또는 더 빠르게 할 수 있습니다.
- ✓ 만약 서로 갈 수 있는 교실의 그룹을 구하면, 같은 그룹에 속해있는지 확인함으로써 문제를 풀 수 있습니다.

부분문제 4 (난방 기구의 개수 $\leq 1,000$)

- ✓ 난방 기구가 켜져 있는 두 정점을 골라서, 그 정점 사이의 거리가 T 이하이면 Union-find 자료구조를 이용해 합쳐주는 연산을 함으로써 문제를 해결할 수 있습니다.
- ✓ **LCA(Least Common Ancestor) 알고리즘**을 이용하면 두 정점 사이의 거리를 $O(logN)$ 에 구할 수 있습니다. 이를 이용하면 부분문제 4를 $O(N^2logN)$ 에 풀 수 있습니다.
- ✓ 해당 풀이는 발전의 여지가 없으므로, 자세한 설명은 생략하고, 다른 풀이를 찾아봅시다

부분문제 4 (난방 기구의 개수 $\leq 1,000$)

- ✓ 트리 dp의 관점에서 생각해봅시다.
- ✓ 가장 먼저 임의의 정점을 루트로 하고, 정점의 깊이 d를 정의합시다. d는 어떤 정점으로부터 루트까지의 거리입니다.
- ✓ 어떤 정점에 대해서, 그 정점을 루트로 하는 서브트리에는 여러 그룹이 존재합니다. **각각의 그룹에서 가장 높은(d가 작은) 점의 d 값의** 집합 S 를 저장합니다. 그룹들은 별개의 union-find 형태로 관리합시다.
- ✓ 먼저 자신의 자손에 대해서 문제를 풀고, 자신의 자손의 집합을 합쳐준 뒤 그룹들을 새로 합쳐주고, 합쳐준 그룹에 대해서 중복되는 값을 빼서 다시 집합 S 를 구하는 식의 dp를 합니다.


: 난방기구 ON
1 ,3
2 ,2
3 ,6
4 ,6
q
3
1
1
2
5
1
3
5
5
2
3
4
1

I'm

부분문제 4 (난방 기구의 개수 $\leq 1,000$)

- ✓ 만약 현재 정점 v가 난방 기구라면, 어떤 i에 대해, $d_i - d_v \leq T$ (즉, $d_i \leq T + d_v$) 라면 정점 i를 정점 v와 합쳐줍니다. 합쳐준 뒤 S에서 d_i는 제거합니다. (v가 i보다 높기 때문)
- ✓ 만약 현재 정점 v가 난방 기구가 아니라면, 어떤 i, j에 대해, $d_i + d_j - 2d_v \leq T$ (즉, $d_i + d_j \leq T + 2d_v$) 라면 정점 i와 정점 j를 합쳐줍니다.
- ✓ 사실 i, j를 보는 것 대신 집합들의 d값 중 최솟값을 d_{min}이라고 하면, $d_i + d_{min}$를 보는 것으로도 충분합니다. $d_i + d_{min} \leq T + 2d_v$ 면 정점 i와 d_{min}에 해당하는 정점을 합쳐주고, S에서 d_i는 제거합니다. 이렇게 하면 선형 시간에 그룹들을 합쳐주고 S의 원소를 제거할 수 있습니다.
- ✓ dp에서 집합을 합치는 것은 당연히 선형 시간에 됩니다.
- ✓ 집합의 크기가 최대 $O(N)$이므로, $O(N^2)$에 문제를 해결할 수 있습니다.

만점 풀이

- ✓ 만점을 받기 위해서, 두 가지의 추가적인 최적화가 필요합니다.
- ✓ **Priority Queue** : 어떤 집합을 합칠 때 **이미 같은 집합에 있던 원소들은 다시 검사할 필요가 없습니다.** 이를 잘 이용하면 priority queue 를 이용해 부분문제 4의 시행을 할 수 있습니다.
- ✓ **Smaller to Larger** : 어떤 집합을 합칠 때, **작은 집합에서 큰 집합으로 합치면 총 시간복잡도가 $O(NlogN)$ 인 것이 보장됩니다.** 이는 어떤 원소에 대해, 그 원소가 포함된 집합의 크기가 최소 2배 이상 늘어나기 때문에 한 원소가 합쳐지는 횟수가 $O(logN)$ 으로 제한되기 때문입니다.
- ✓ 이것을 이용해서 $O(Nlog^2N)$ 에 문제를 해결할 수 있습니다.

H. Love is war

필요한 개념: `binary-search`, `tree-set`
예상 난이도: **Diamond IV**

- 제출 3회, 정답 0명, 최고점 20점
- 최고점에 가장 빨리 도달한 사람: 변재우, 151분
- 출제자: 이유찬
- 우정이와 아름이는 원래 누군가의 본명이였습니다

문제 분석

- ✓ 제가 만든 문제 중 가장 마음에 들었던 문제입니다. G번 문제와 반대로 많은 알고리즘이 쓰이지 않고, 코드도 상당히 짧습니다.
- ✓ 우리는 문제를 다음과 같이 변형할 것입니다.
- ✓ **두 수열 → 여러 구간(interval)들 → 2차원 평면(2d plane)**
- ✓ 서브테스크 풀이 없이, 만점 풀이로 넘어가겠습니다

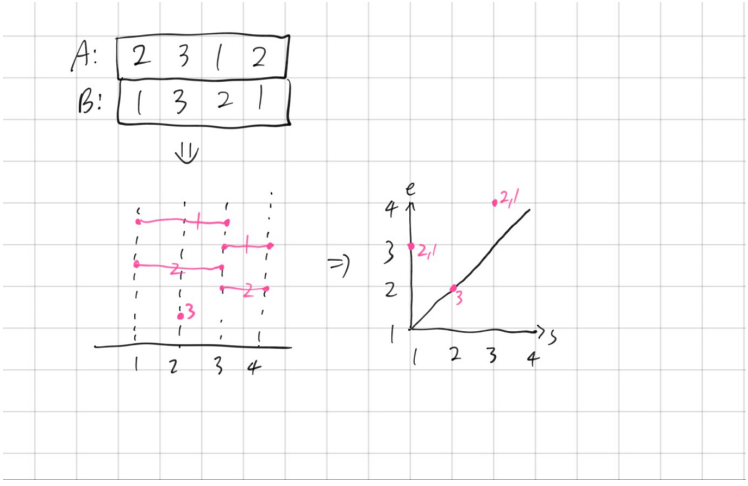
만점 풀이

- 수 하나에 대해서, 그 수가 $A[l..r] \cap B[l..r]$ 에 포함될 조건을 생각해봅시다.
- A 에서 그 수가 있는 위치를 $a_1, a_2, ..., a_i$, B 에서 그 수가 있는 위치를 $b_1, b_2, ..., b_i$ 라고 하고 이를 합쳐서 정렬하면, 인접한 a_i 와 b_i 를 구할 수 있습니다.
- 두 수열을 어떤 수가 써져있는 구간 $O(N)$ 개로 바꿀 수 있습니다.
- 구간들의 집합을 S 라고 합시다. S 의 각 원소는 (구간 $(l..r)$, 수 x) 입니다.
- 이걸 하면 뭐가 좋을까요?

만점 풀이

- ✓ 어떤 구간에 대해, 그 구간의 답은 S 중 그 구간에 포함된 구간에 적혀있는 수의 최댓값입니다.
- ✓ 어떻게 해야 이를 빠르게 계산할 수 있을까요?
- ✓ **구간을 2차원 점 하나에 대응해봅시다.** 구간은 (l,r)로 나타낼 수 있습니다. 이를 x,y좌표로 생각하여 그대로 2차원 평면 위에 올려봅시다.

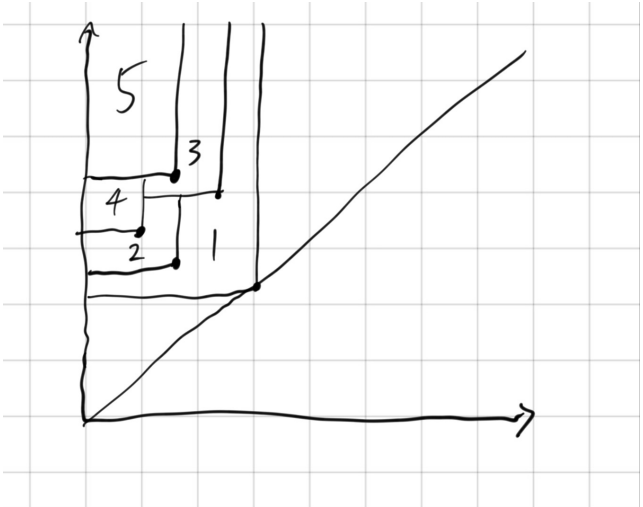
만점 풀이



만점 풀이

- 좌표평면의 $x \geq 1, y \leq N, x \geq y$ 영역의 삼각형의 각 좌표에 그 구간에 해당하는 답을 써봅시다.
- 어떤 점에 쓰인 수는 그 점보다 **오른쪽 아래**에 있는 S의 원소 중에 v의 최댓값입니다.
- 문제의 답은 모든 점에 쓰인 수의 합입니다.
- 그림으로 그리면 다음과 같은 모양이 됩니다.

만점 풀이



만점 풀이

- ✓ 작은 수는 큰 수에 영향을 주지 않으므로, 큰 수부터 작은 수까지 영역을 추가한다는 방식으로 문제를 해결합시다. (S를 v의 내림차순으로 정렬)
- ✓ 각 원소를 보면서 **왼쪽 위의 영역**을 색칠한다고 생각하고, 아직 색칠되지 않은 영역의 넓이를 구할 수 있다면 그 넓이에 v를 곱한 값이 새로 써진 수들의 합입니다.

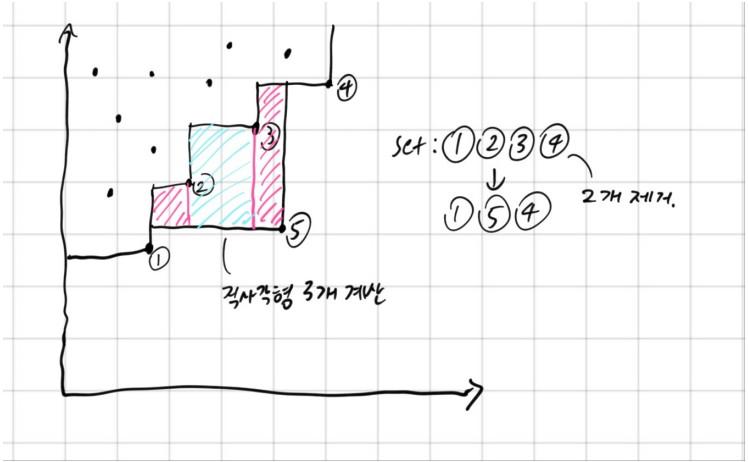
만점 풀이

- ✓ **std::set**은 원소의 삽입인 insert, 원소의 삭제인 erase, 그리고 어떤 원소가 존재하는지 찾는 find, 어떤 원소보다 큰 최소 원소를 찾는(이분 탐색) lower bound 를 모두 O(log N) 에 진행할 수 있는 자료구조입니다.
- ✓ std::set에 **x좌표가 증가하면서 y좌표가 증가하는 점**들을 저장합시다. (저는 이걸 increasing hull 이라고 부르는 데 실제로 쓰는 용어인진 모르겠습니다)

만점 풀이

- ✓ std::set의 lower bound 연산을 이용하여 최초로 자신보다 y 좌표가 큰 점을 구할 수 있고, 자신보다 x 좌표가 커질 때까지 현재 원소를 지우고 다음 원소로 넘김으로써 (set.erase()는 원소를 지운 뒤 다음 원소를 반환합니다) std::set의 단조성을 유지할 수 있습니다.
- ✓ 이것을 하는 동시에 새롭게 색칠되는 영역을 **직사각형 여러 개의 합**으로 구할 수 있습니다.

만점 풀이



Enter Caption

만점 풀이

- ✓ 직사각형 k 개의 넓이를 계산하면, set에선 $k-1$ 개의 원소가 빠집니다. 따라서 계산해야하는 총 직사각형 넓이는 $O(N)$ 개입니다.
- ✓ 따라서 (set의 연산을 이요하므로) 총 시간복잡도 $O(NlogN)$ 에 문제를 풀 수 있습니다.
- ✓ 이런 형태의 시간복잡도를 **amortized** 시간복잡도라고 합니다.

감사합니다

- ✓ 긴 풀이를 따라와주셔서 감사합니다.
- ✓ 2024 나는코더다 부원이 되신 것을 진심으로 환영합니다!