20.12.09 동아리 풀이

선린인터넷고등학교 소프트웨어과
30610 나정휘
https://JusticeHui.github.io

JusticeHui

문제 목록

- A. 트리플 소트 - 2020 UNIST 교내 대회 A번
- B. 화학 실험 - 2020 UNIST 교내 대회 C번
- C. CPU 벤치마킹 - 2020 UNIST 교내 대회 D번
- D. 대홍수 - 2020 UNIST 교내 대회 F번
- E. 쿼리와 쿼리 - 2019 연세대학교 교내 대회 K번
- F. 게임 개발자 영우 - 2020 중앙대학교 교내 대회 G번
- G. 야바위 - 2020 UNIST 교내 대회 G번
- H. Confuzzle - 2020 서강대학교 교내 대회 Div1 G번

JusticeHui

트리플 소트 (1)

- 홀수 인덱스에서 짝수 인덱스로 이동할 수 없다.
- 짝수 인덱스에서 홀수 인덱스로 이동할 수 없다.

- i%2 == A[i]%2인지 확인하면 된다.

- http://boj.kr/7df0c2e35f544a97926186587a81f0e6

JusticeHui

화학 실험 (1)

- 어떤 색이 (N+1)/2개보다 많이 있으면 불가능하다.
- 그렇지 않다면 항상 가능하다.

- 개수가 가장 많은 색깔부터
- 1 3 5 … 2 4 6 … 순서로 배치하면 된다.

- http://icpc.me/1545도 풀어보자.
- http://boj.kr/ca6e355be0004c5ea6eba2fa1418b84d

JusticeHui

CPU 벤치마킹 (1)

- 각 행마다 어떤 수가 적히는지 생각해보자.
  - 1행 : 아무것도 안 적힌다.
  - 2행 : A_1
  - 3행 : A_1*A_2 / A_2
  - 4행 : A_1*A_2*A_3 / A_2*A_3 / A_3

- S_i를 (i+1)번째 행에 적힌 수의 합이라고 하자.

JusticeHui

CPU 벤치마킹 (2)

- S_i를 (i+1)번째 행에 적힌 수의 합이라고 하면
  - S_i = (S_{i-1} + 1) * Ai 가 된다.

- 간단한 DP 문제로 바뀐다.

- http://boj.kr/04673bbb87374694864157 95d157e5d0

JusticeHui

대홍수 (1)

- L[i] = i번째 사람이 갈 수 있는 가장 왼쪽 지점
- R[i] = i번째 사람이 갈 수 있는 가장 오른쪽 지점

- L[i] ≤ L[i+1]이고, R[i-1] ≤ R[i] 인 것은 쉽게 알 수 있다.
- 투포인터를 이용해 L[i]와 R[i]를 O(N)에 구하자.
- 최대 높이는 세그먼트 트리로 구하면 된다.

- http://boj.kr/a6c860364e7f43248549d8a21a0dd532

JusticeHui

쿼리와 쿼리 (1)

- 문제에서 요구하는 것
  - 모든 [L_i, R_i] 구간을 합한 집합을 **S**라고 하면
  - 주어진 L_i와 R_i를 적절히 매칭해서 **S**의 최댓값을 최소로 만드는 것

- L_i ≤ L_j, R_i ≤ R_j 라면 [L_i, R_i], [L_j, R_j]로 매칭하는 것이 이득임을 보일 것이다.

JusticeHui

쿼리와 쿼리 (2)

- $L_i \le L_j$, $R_i \le R_j$ 일 때
- $[L_i, R_j]$, $[L_j, R_i]$를 매칭한 것을 생각해보자.

- 만약 $L_i > R_j \ ||\ L_j > R_i$라면 정답은 1e9이다.
- 그렇지 않다면 $[L_i, R_j]$가 $[L_j, R_i]$를 포함한다.

JusticeHui

쿼리와 쿼리 (3)

- 포함 관계를 풀어서 [L_i, R_i], [L_j, R_j]로 매칭해도 손해를 보지 않는다.
- 오히려 L_i > R_j || L_j > R_i인 상황을 풀어준다.

- 이 사실을 이용해 Exchange Argument로 최적임을 증명할 수 있다.

JusticeHui

쿼리와 쿼리 (4)

- 주어진 `L_i`와 `R_i`를 정렬한 다음, 차례대로 매칭하면 된다.

- http://boj.kr/7775ae8724c641ffa312dd43ff79d458

JusticeHui

게임 개발자 영우 (1)

- X값을 고정하면, 가능한 Y값은 0 혹은 1개이다.
- N 이하의 모든 자연수 X에 대해, 가능한 Y값을 구하자.

- 풀이부터 설명하자면,
  - Prefix Sum과 Parametric Search를 이용해 레벨 업 하는 지점을 찾아서
  - N번째 사냥으로 인해 레벨 업이 되는지 확인하면 된다.
  - N번째 사냥으로 레벨 업이 된다면 **그 레벨**이 Y값이다.

JusticeHui

게임 개발자 영우 (2)

- 시간 복잡도를 계산하자.
  - 매 사냥마다 경험치를 1 이상 얻기 때문에
  - X값을 고정하면 최대 N/X번 레벨 업 할 수 있다.

  - 모든 X에 대해 확인하므로
  - Parametric Search는 최대 O(N log N)번 수행한다.
  - 따라서 총 시간 복잡도는 O(N log^2 N)이다.

JusticeHui

게임 개발자 영우 (3)

- 매 사냥마다 얻는 경험치는 홀수이기 때문에 매번 경험치의 기우성이 바뀐다.

- 경험치가 0으로 초기화되어 다시 시작하는 시점에 따라서
- 뒷면의 경험치를 1515...와 5151...로 결정하면 된다.

- http://boj.kr/45096899a60a4902a55c7fec3a118489

JusticeHui

야바위 (1)

- 각 쿼리마다 구슬의 최종 위치만 알면 된다.
  - 오프라인으로 문제를 해결할 것이다.
  - S_i번째 컵(std::set)에 i번째 구슬을 넣고, 구슬을 이동시키자.

- 윤이가 본 N-1개의 동작은 지문에 나와있는대로 처리하면 된다.
- 이후 주어진 M개의 동작은 해당하는 구슬만 이동하면 된다.

JusticeHui

야바위 (2)

- 두 set을 합치는 것은 Small to Large를 사용하면, 각 구슬은 최대 O(log M)번 이동하게 된다.

- 쿼리를 정렬하는데 O((N+M) log (N+M))
- 쿼리를 처리하는데 O(M log^2 M)
- 시간 내에 문제를 풀 수 있다.

- http://boj.kr/3bb8ca848aa5433e88a7d575115dbbca

JusticeHui

Confuzzle (1)

- 풀이가 2개 존재한다.
  - **Solution 1**. Centroid Decomposition
  - **Solution 2**. Sqrt Decomposition

- Solution1은 다 안다고 믿고, Solution2만 설명한다.
  - Centroid 모르면 내 블로그 보자.
  - https://justicehui.github.io/hard-algorithm/2020/08/25/centroid/

JusticeHui

Confuzzle (2)

- O(N^2) 정도에 문제를 해결하는 풀이 두 개를 설명한다.
  - 색깔이 c인 모든 정점 쌍을 확인
    - 색깔이 c인 정점이 M개라면 O(M^2) 혹은 O(M^2 log N) 정도 걸린다.
  - Multi-Source BFS를 돌려서 가장 가까운 점 확인
    - 각 색깔마다 O(N) 짜리 루틴을 수행하므로, O(N*(색깔 개수)) 정도 걸린다.

- 두 풀이는 모두 최악의 경우 O(N^2) 정도 걸리지만
- 최악의 경우가 되는 조건이 **서로 다르다**.

JusticeHui

Confuzzle (3)

- 각 풀이가 빨라지는 조건을 살펴보자.
  - 모든 정점 쌍을 확인하는 풀이
    - 그 색깔을 갖고 있는 정점이 적을수록 이득
  - Multi-Source BFS
    - 색깔의 종류가 적을수록 이득
- 두 풀이를 섞자!

- 어떤 적당한 상수 K를 잡아서
  - 색깔이 c인 정점이 K개 이하라면 모든 정점 쌍을 확인하고
  - K개 초과라면 Multi-Source BFS를 할 것이다.

JusticeHui

Confuzzle (4)

- 모든 정점 쌍을 확인하는 풀이에서는
  - 각 정점은 최대 K개의 칸과 비교하게 된다.
  - O(log N)만에 LCA를 구한다면 O(NK log N)이 걸린다.
- Multi-Source BFS 풀이에서는
  - 정점이 K개 이상 있는 색은 최대 N/K가지이다.
  - O(N)짜리 BFS를 최대 O(N/K)번 수행한다.
- 총 시간 복잡도는 O(NK log N + N^2/K)이다.

JusticeHui

Confuzzle (5)

- O(NK log N + N^2/K)
  - K를 sqrt(N)으로 잡으면 O(N sqrt N log N)이 된다.
  - K를 sqrt(N/logN)으로 잡으면 O(N sqrt(N log N))이 된다. **~~십덕~~**

- LCA를 O(1)만에 구하는 방법이 있다. -> O(NK + N^2/K)
  - K를 sqrt(N)으로 잡으면 O(N sqrt N)이 된다.

JusticeHui

Confuzzle (6)

- O(1) LCA는 secmem에 잘 나와있다.
  - http://www.secmem.org/blog/2019/03/27/fast-LCA-with-sparsetable/
- O(N log N) with Centroid
  - http://boj.kr/e4db954b41c54226911477abfe018c3f
- O(N sqrt(N log N)) with O(log N) LCA
  - http://boj.kr/2679cb6504614d43add226fef1422c63
- O(N sqrt N) with O(1) LCA
  - http://boj.kr/39fa7b56f3624db1be5c035fd10f55de

JusticeHui