20.11.18 동아리 풀이

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

문제 목록

- A. 어려운 문제 - 2008 NWERC D번
- B. 넴모넴모 2020 - 2020 서울대 대회 Div.2 C번
- C. 요세푸스 문제 3
- D. Number Sets - 2008 Google Code Jam Round1B B번
- E. 순열 그래프 - 2013 Daejeon Regional I번
- F. 요세푸스 문제 2
- G. 소방차 - 2005 KOI 고등부 3번
- H. 편식 - 2015 크로아티아 올림피아드 4번

어려운 문제 (1)

- 모든 a, b에 대해 시도해보면 된다.
- TLE가 난다면 커팅을 잘 하자.
- http://boj.kr/aa28b74b50b84d1c8cb4e32a50f8c08d

넴모넴모 2020 (1)

- (x, y)가 주어질 때 아래 두 값을 빠르게 구하면 된다.
  - x번째 칸에서 제거되는 개수
  - y층에서 제거되는 개수

- x > A[y]라면 제거되는 y층에서 제거되지 않는다.
- 그렇지 않다면
  - x < A[z], y ≤ t ≤ N인 가장 큰 z에 대해, x번째 칸에서 (t-y+1)개 제거
  - y층에서 (A[y]-x+1)개 제거

넴모넴모 2020 (2)

- **x < A[z], y ≤ t ≤ N**인 가장 큰 z는 이분 탐색으로 구할 수 있다.

- 난 이분 탐색 코딩하기 싫어서 Merge Sort Tree 짬
  - y축에서 제거되는 개수는 (A[y]-x+1)로 하고
  - x번째 칸에서 제거되는 개수를 Merge Sort Tree로 계산

- http://boj.kr/19c832ca7db44ef995bf82a4b3e40ff4

요세푸스 문제 3 (1)

- f(N, K) = 입력이 N, K일 때 정답
- f(N, K) = (f(N-1, K) + K) mod N
  - 0-based다.
  - K번째 사람 없애고, 그 다음 사람을 0번으로 Renumbering
- 재귀로 짜면 메모리 터질 수도 있다.
- http://boj.kr/ea002341fe4b49f1b3ae2abe64de79ee

Number Sets (1)

- p | a & p | b 이면 p | b-a다. -> p는 100만 이하다.
  - x | y : y는 x로 나누어 떨어진다.
- p 이상의 모든 소인수에 대해, p의 배수를 Union해주면 된다.
- X = 1e6일 때, O(X log log X * log X)에 문제를 풀 수 있다.
  - N 이하의 모든 소수 p에 대해, N/p의 합은 O(N log log N)이다.
- http://boj.kr/3bebdf7e6fd1413eb693d442cc06bc17

순열 그래프 (1)

- 입력을 잘 가공해주면
  - x < y이면서 A[x] > A[y]인 (x, y) 개수를 구하는 문제로 바뀐다.

  - Inversion Counting 문제
    - Merge Sort를 이용한 풀이
    - Fenwick Tree/Segment Tree 등 자료구조 이용

- Merge Sort를 이용한 풀이
  - https://justicehui.github.io/ps/2019/04/23/BOJ1517/ 참고

- Fenwick Tree를 이용한 풀이?

순열 그래프 (2)

- x < y이면서 A[x] > A[y]인 수
  - 각 i에 대해, A[1] ~ A[i-1] 중 A[i]보다 큰 원소의 개수를 구해서 더하자.

```
for(int i=1; i<=N; i++){
    ans += sum(A[i]+1, MX);
    add(A[i], 1);
}
```

- http://boj.kr/ed890fead46847f38b2d846f9c79e01b

요세푸스 문제 2 (1)

- 임의의 자연수 k에 대해, k번째 원소를 구할 수 있으면 된다.
  - Segment Tree로 O(log N)만에 하는 방법
  - OS Rank로 (느린) O(log N)만에 하는 방법
  - Fenwick Tree와 Parametric Search로 O(log^2 N)만에 하는 방법

  - 위 2개는 AC, 아래는 TLE
  - AC를 받는 풀이 2개를 모두 소개한다.

요세푸스 문제 2 (2)

- Segment Tree를 이용해 O(log N)만에 하는 방법
- 코드 보고 이해하는 것이 편하다.

```
int tree[1 << 18];
const int sz = 1 << 17;
void add(int x, int v){
  x += sz; tree[x] += v;
  while(x /= 2) tree[x] = tree[x*2] + tree[x*2+1];
}
int kth(int k){
  int x = 1;
  while(x < sz){
    if(k <= tree[x*2]) x = x*2;      //k번째 원소가 왼쪽 서브트리에 있음
    else k -= tree[x*2], x = x*2+1; //오른쪽 서브트리에 있음
  }
  return x - sz;
}
```

요세푸스 문제 2 (3)

- OS Rank를 이용해 느린 O(log N)만에 하는 방법

- https://codeforces.com/blog/entry/11080 를 보자.
  - GCC / G++에서만 가능하다.
  - Visual Studio에서 안 된다고 짜증내지 말자.

- SegTree : http://boj.kr/f16205569c4a4837a55243d204b8458e
- OSRank : http://boj.kr/823ecad1b8024e7694a53a9e2dc491f6

소방차 (1)

- 매칭하는 것을 “구간” 이라고 생각하자.
- 정답이 어떤 형태인지, 혹은 특정한 형태로 강제할 수 있는지 생각해보자.

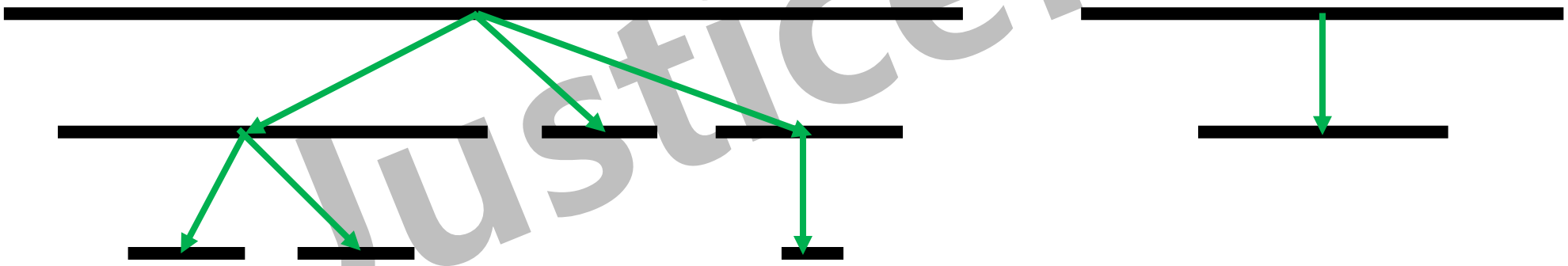
소방차 (2)

- ***Lemma 1.*** 구간들이 서로를 완전히 포함하거나 분리된 형태의 최적해가 존재한다.
- ***Proof.*** $a \le b \le c \le d$ 일 때 (a, c)와 (b, d)를 매칭하는 것이 최적이라고 하자. (a, d), (b, c)로 바꿔도 거리는 동일하다.

- ***Lemma 2.*** 최적해에 [l, r] 매칭이 존재한다면, 이 구간 안에 있는 소방차와 펌프는 모두 매칭되어 있다.
- ***Proof.*** 그렇지 않으면 적당히 바꿔서 거리를 더 줄일 수 있다.

소방차 (3)

- 구간의 포함 관계를 트리(혹은 포레스트)처럼 생각할 수 있다.
- 같은 depth에서는 모든 구간이 분리되어 있다.



소방차 (4)

- 모든 입력 좌표를 정렬하고
  - 펌프가 나오면 현재 depth에 넣고 depth++
  - 소방차가 나오면 --depth하고 그 depth에 넣음

- 같은 depth에서는 펌프와 소방차가 번갈아 나온다.
  - 원소가 짝수 개라면 순서대로 매칭해주면 된다.
  - 홀수 개라면 어떤 거 하나 빼고 매칭해야 한다.
    - 선형 시간에 계산할 수 있다.

- http://boj.kr/91b9e89374e34559b37f2477f06e1f89

편식 (1)

- 올리브의 Convex Hull 위에 있는 점만 생각해도 된다.

- 피자의 두 꼭짓점을 잇는 선분이 볼록 껍질과 닿지 않으면 된다.
  - 어떻게 빠르게 판별할까?
  - N^2개의 선분을 다 봐야 할까?

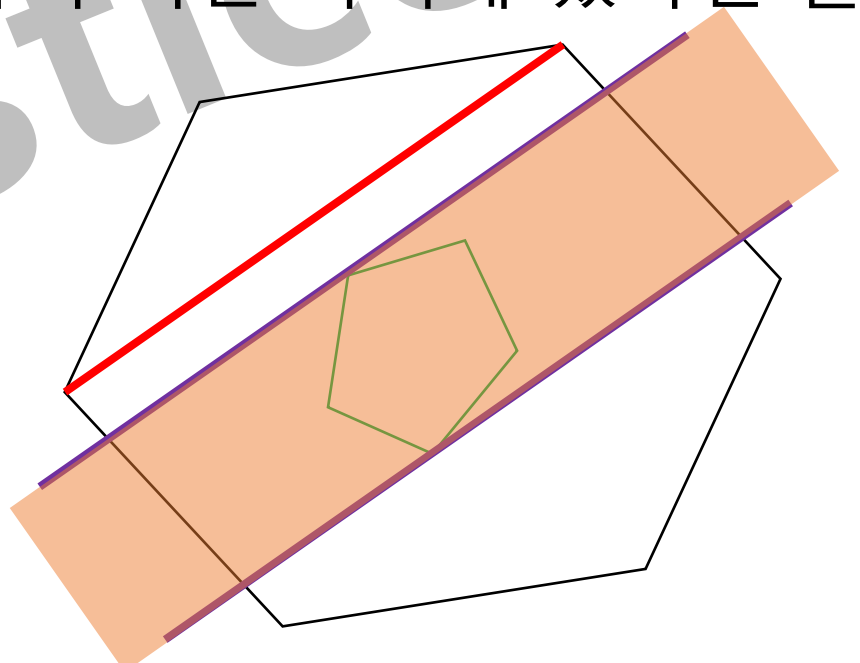
편식 (2)

- 필요한 개념
  - Convex Hull - 윗 껍질과 아랫 껍질 구하는 방법 (Monotone Chain)
  - 삼분 탐색
  - 투 포인터
  - 볼록 다각형의 접선 - 뒤에서 아주 간단하게 설명함
  - 신발끈 공식

편식 (3)

- 두 꼭짓점을 잇는 선분과 평행하면서 (빨간색)
- 올리브 볼록 껍질에 접하는 두 직선을 그려보자. (보라색)

- 빨간색 선분이 두 보라색 직선 사이에 있다면 불가능
- 그렇지 않으면 가능



편식 (4)

- 빨간색 선분이 두 보라색 직선 사이에 있다
  - 보라색 직선 y절편 ≤ 빨간색 선분 y절편 ≤ 다른 보라색 직선 y절편
  - 보라색 선분 y절편의 최대/최솟값을 구하면 된다.
  - 삼분 탐색으로 구할 수 있다.
    - Convex Hull을 윗 껍질과 아랫 껍질로 분리한 뒤, y절편으로 삼분 탐색하면 된다.
- 어떤 선분으로 자를 수 있는지 판별하는 것은 O(log M)에 할 수 있다.

편식 (5)

- 잘 생각해보면, 굳이 O(N^2)개의 선분을 모두 볼 필요가 없다.

- 투 포인터 느낌으로
  - i번째 점과 i+1, i+2, i+3, … , j번째 점을 연결할 수 있다고 하자.
  - i+1번째 점은 당연히 i+2, i+3, … , j번째 점과 연결할 수 있고
    - i+2 ~ j-1번째 점과 연결하는 것은 최대 넓이가 될 수 없다.
  - 그러므로 i+1번째 점은 j번째 점부터 시작하면 된다.

- O(N)개의 선분만 확인하면 된다.
- 넓이는 신발끈 공식 이용하면 상수 시간에 관리할 수 있다.

편식 (6)

- 올리브의 볼록 껍질을 구하는데 O(M log M)
- 판별해야 하는 선분의 개수 O(N)
  - 각 선분이 자를 수 있는 선분인지 판별하는데 O(log M)
  - 각 선분마다 넓이의 변화를 관리하는데 O(1)

- O((N+M) log M)에 문제를 풀 수 있다.

- http://boj.kr/2626bd351a2941c787e3c8b7e1d1de0e