Sogang ICPC Team

2021-1 중급 스터디 2회차

임지환
raararaara@gmail.com
2021.4.4

목차

- 강사 소개
- 도입
    - 강의 소개
    - About sqrt decomposition
- 유형
    - on array
    - Batch processing (with Mo’s)
    - on case
    - 기준값 설정
- 정리

- 컴퓨터공학과 14학번
- Kakao Corp. Reco team intern (2021.3 ~)
- Handle - raararaara
- 2020 ICPC 본선 (Team BlackWeasel)

도입

- 강의 소개
- about sqrt decomposition

- 이 강의를 임하는 자세

1. (힘들지만) '할만하다' 라는 마음가짐

2. (따라오기 어렵다면) 목적을 '아는 domain 확장하기'로 잡기

3. (문제는 안풀어도 좋으니) **끝까지 완주하기**

- Square root Decomposition (평방 분할)

- 연산, 또는 구조를 $O(\sqrt{N})$을 기준으로 분해
- $O(\log n) \leq O(\sqrt{N}) \leq O(N)$
- 자료구조(Data structure) 의 관점으로, 방법론적(Method) 관점으로 접근 가능
- 쿼리 문제를 해결할 때 주로 사용

- Time Complexity의 관점

- 쿼리 문제의 경우, $O(\sqrt{N})$또는 $O(\sqrt{Q})$의 경우가 대부분으로, $N, Q \leq 10^6$라면 사용을 고려해보자.
- 상수(constant) 나눗셈이 더 빠르다. $\sqrt{N}$을 고집하지 말고 $\sqrt{N}$ like한 적당한 상수를 잡자.

- sqrt heuristics

- time complexity 상에서 $\sqrt{N}(\text{or close})$가 붙는 문제들

유형

- on Array
- Batch processing (with Mo’s)
- on Case
- 기준값 설정

- Range Queries

\- L R : [L, R] 구간에 대한 연산(sum || min || max etc..)

을 굳이 $O(\sqrt{N})$에 구해보자.

- Range Sum

array																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	9	2	2	3	3	7	2	0	3	6	8	5	4	7	7	5

- Range Sum

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
array	9	2	2	3	3	7	2	0	3	6	8	5	4	7	7	5

- Range Sum

$\sqrt{N}$

<table>
<tr><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td></tr>
<tr><td>2</td><td>3</td><td colspan="4">12</td><td colspan="4">22</td><td>4</td></tr>
</table>

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
array	9	2	2	3	3	7	2	0	3	6	8	5	4	7	7	5

- Range Sum

구간 개수 ≤ $\sqrt{N}$

<table>
<tr><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td></tr>
<tr><td>2</td><td>3</td><td colspan="4">12</td><td colspan="4">22</td><td>4</td></tr>
</table>

array	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	9	2	2	3	3	7	2	0	3	6	8	5	4	7	7	5

- Range Sum

$|l_{end}| \le \sqrt{N}$

$|r_{end}| \le \sqrt{N}$

2	3	4	5	6	7	8	9	10	11	12
2	3	12				22				4

array	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	9	2	2	3	3	7	2	0	3	6	8	5	4	7	7	5

- Range Sum

bucket index of $i : \frac{i}{\sqrt{N}}$

bucket

0	1	2	3
16	12	22	23

array

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
9	2	2	3	3	7	2	0	3	6	8	5	4	7	7	5

for $s = \sqrt{N}$,
p buckets,

$$\sum_{i=l}^{r} a[i] = \sum_{i=l}^{(k+1)\cdot s-1} a[i] + \sum_{i=k+1}^{p-1} b[i] + \sum_{i=p\cdot s}^{r} a[i] = O(\sqrt{N}) \text{ per query}$$

- Update (point)

bucket

0	1	2	3
16	12	22	23

array

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
9	2	2	3	3	7	2	0	3	6	8	5	4	7	7	5

- Update (point)

bucket

0	1	2	3
16	12	22	23

array

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
9	2	2	3	7	7	2	0	3	6	8	5	4	7	7	5

- Update (point)

bucket	0	1	2	3
	16	16	22	23

array	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	9	2	2	3	7	7	2	0	3	6	8	5	4	7	7	5

$$a[i] + v \quad \Rightarrow \quad b\left[\frac{i}{\sqrt{N}}\right] + v$$

Problem 1

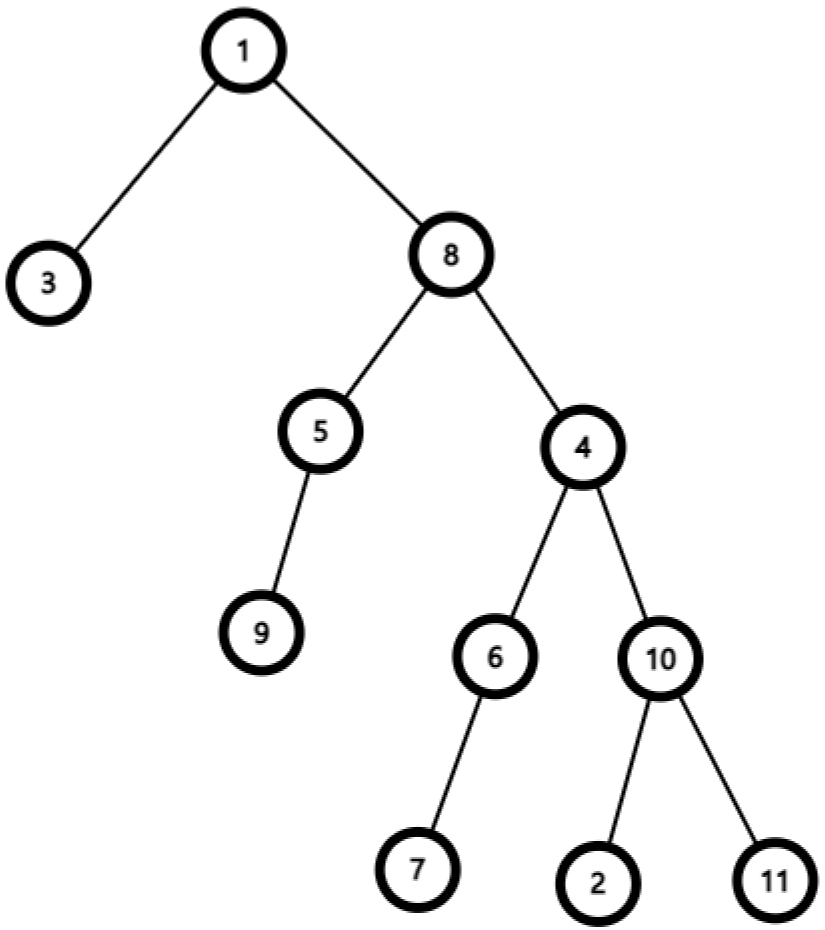
- 노드 $N(1 \leq N \leq 100{,}000)$개로 구성된, 루트가 1인 트리
- 각 노드마다 최초 가중치는 0
- $M(1 \leq M \leq 100{,}000)$개의 쿼리
  - $1\ i\ w$: i번 노드 포함, i아래에 있는 모든 노드의 가중치에 w만큼을 더함. $(|w| \leq 10{,}000)$
  - $2\ i$: i번 노드의 현재 가중치 출력

Problem 1

- segtree + lazy를 쓸 수 있습니다.
- lazy를 안써도 풀 수 있습니다.
- 굳이 sqrt decomposition으로 풀어보겠습니다.

Problem 1

1) dfs ordering



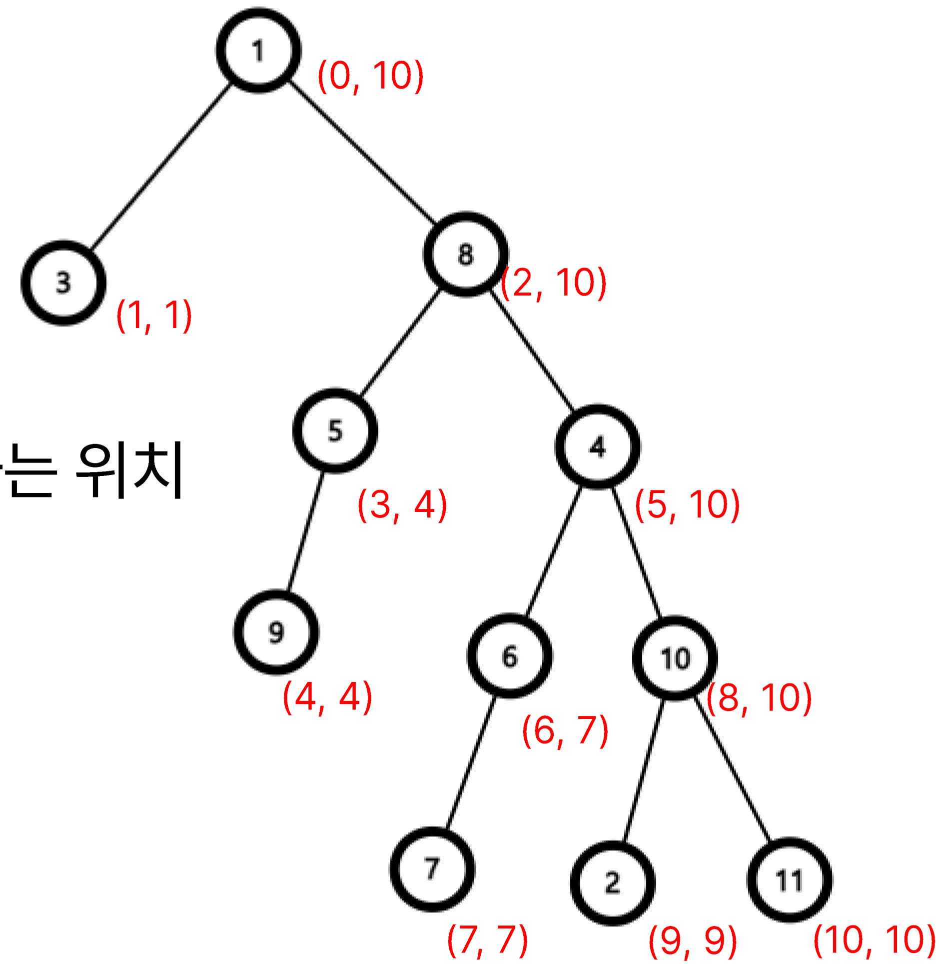
order	0	1	2	3	4	5	6	7	8	9	10
i	1	3	8	5	9	4	6	7	10	2	11

Problem 1

2) labeling as:

$l[i]$: index i가 order 상에서 처음으로 등장하는 위치

$r[i]$: index i의 자손들 중 가장 높은 order

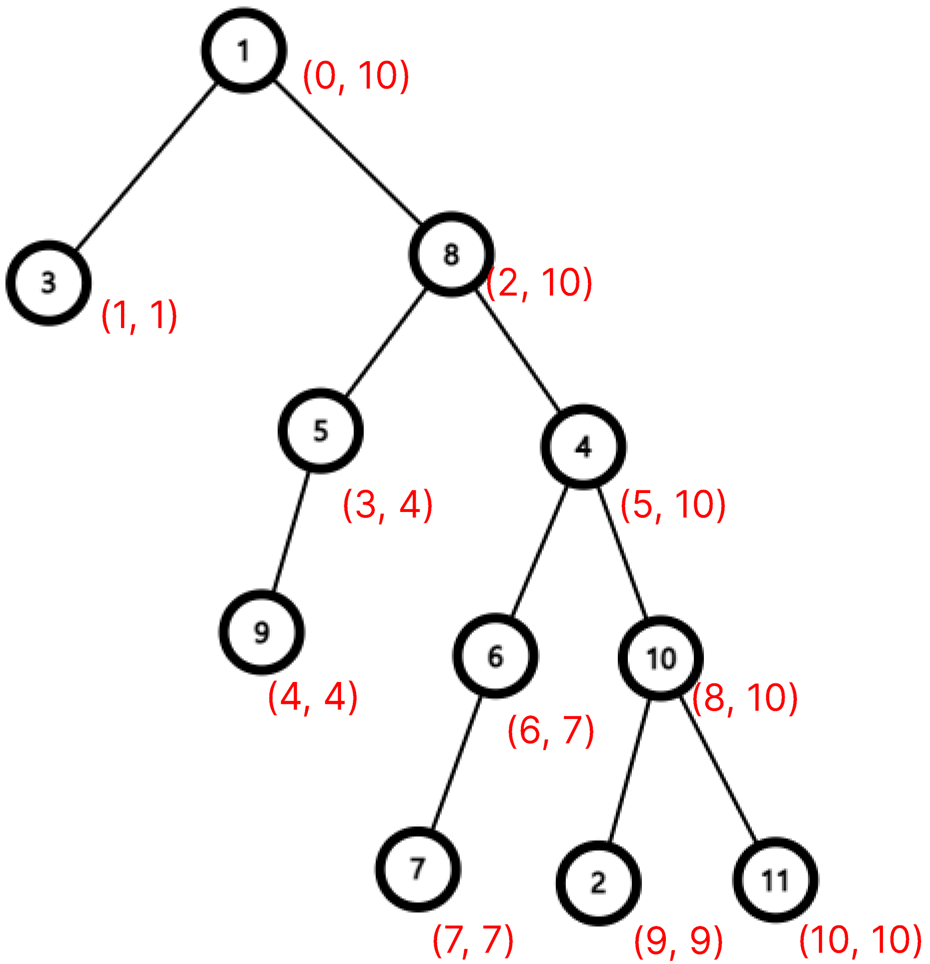


order	0	1	2	3	4	5	6	7	8	9	10
i	1	3	8	5	9	4	6	7	10	2	11

Problem 1

3) 쿼리 처리

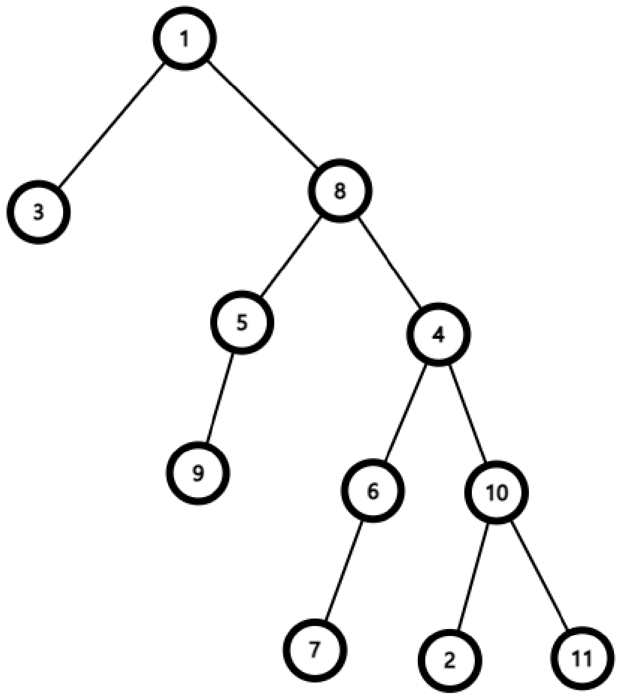
- $1\ i\ w : l[i] \sim r[i]$ 각각에 w만큼 덧셈
- $2\ i : l[i]$에 해당하는 값



order	0	1	2	3	4	5	6	7	8	9	10
i	1	3	8	5	9	4	6	7	10	2	11

Problem 1

3) 쿼리 처리



- $1\ i\ w : l[i]{\sim}r[i]$ 각각에 w만큼 덧셈
  -> update(l[i], r[i], w) … $O(\sqrt{N})$
- $2\ i : l[i]$에 해당하는 값
  -> $v = b\left[\frac{l[i]}{\sqrt{N}}\right] + a[l[i]]$ … $O(1)$

i	1	3	8	5	9	4	6	7	10	2	11
$l[i]$	0	1	2	3	4	5	6	7	8	9	10
$r[i]$	10	1	10	4	4	10	7	7	10	9	10

- Update (range) with point query

bucket

0	1	2	3
0	0	0	0

array

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
9	2	2	3	3	7	2	0	3	6	8	5	4	7	7	5

- Update (range) with point query

bucket

0	1	2	3
0	0	0	0

array

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
9	2	2	3	3	7	2	0	3	6	8	5	4	7	7	5

+3 for all

- Update (range) with point query

bucket	0	1	2	3
	0	3	3	0

array	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	9	2	5	6	3	7	2	0	3	6	8	5	7	7	7	5

+3 for all

Batch processing

- batch : people or things dealt with as a group or at the same time

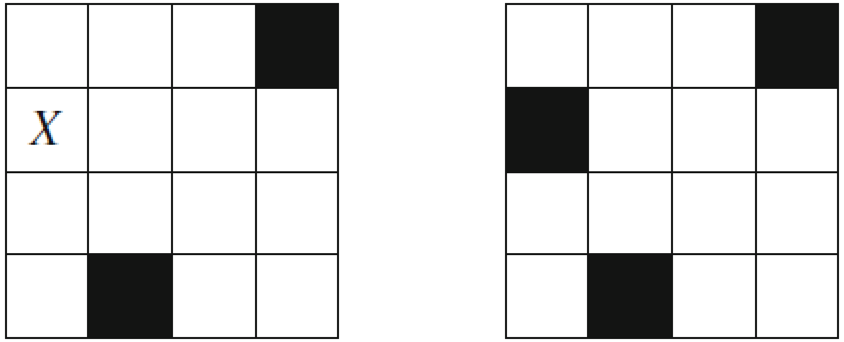
- 정확히는, k개 쿼리끼리 묶어서 처리

- k개의 쿼리마다 초기화
  - 쿼리값 반영 & 전처리

- batch 내에서의 순서는 여전히 중요

Problem 2

- $N \times N$ 격자에 한 칸만 검은색
- 총 $N^2 - 1$번의 쿼리
  - $x\ y$: (x, y)로부터 가장 가까운 검은 칸까지의 맨해튼 거리
- 쿼리 처리 후 해당 칸을 검은색으로 칠하기



Problem 2

- Naïve approach – 그냥 구해보기

1) 검은 칸 목록을 관리하여 일일이 거리 계산하기

$$\sum_{k=1}^{N^2} k = O(N^4)$$

Problem 2

- Naïve approach – 그냥 구해보기

1) 검은 칸 목록을 관리하여 일일이 거리 계산하기
2) 매 쿼리마다 BFS 수행

$$= O(N^4)$$

Problem 2

- Better approach – 전처리

1) k개의 쿼리들(Batch)에 대해서, 각 쿼리에 대해 검은 칸까지의 최소 거리 전처리
   -> $O(N^2)$ by BFS
2) Batch에서 생기게 되는 검은칸의 목록 또한 관리
   -> 각 쿼리의 결과: min(전처리한 값, batch 내에서 현재 이전의 검은 칸들까지의 거리)
   -> $O(k)$ by brute forcing
3) 매 Batch마다 1)~2) 반복
   -> $O(N^2) \times \frac{N^2}{k} + O(N^2k)$

Problem 2

- Better approach – 전처리

if $k = \sqrt{N^2} = N,$

$$O(N^2) \times \frac{N^2}{k} + O(N^2 k) = O(N^3)$$

- Introduction

- static array에서의 query by Offline

- query의 그룹화 by ‘sqrt-like’ indices

Problem 3

- 길이 $N(1 \le N \le 100{,}000)$인 수열
- $M(1 \le M \le 100{,}000)$개의 쿼리
  - $l\ r : a_l, a_{l+1}, \dots, a_r$에 존재하는 서로 다른 수의 개수

Problem 3

- Naïve approach – 일일이 세어보기

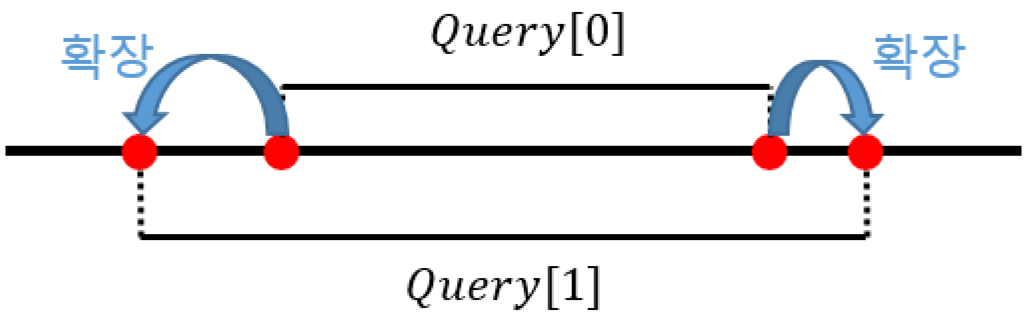
- 수의 uniqueness 판단
  - `std::set`
  - `cnt[i]` := value i가 등장한 횟수, unique count++ when cnt[i] = 0 -> 1

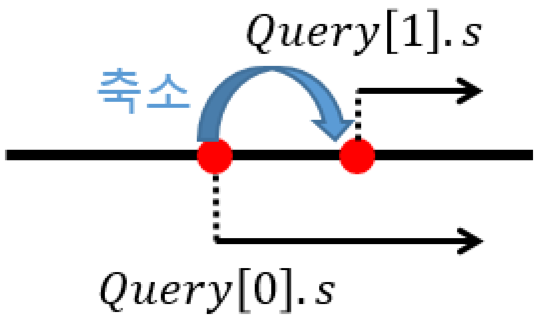
- but..$O(MN)$

Problem 3

- Better approach – 이전 구간 정보 이용하기

\- cnt[i]가 0->1 또는 1->0인 경우 쿼리값 변화





\- $O(|s_i - s_{i-1}| + |e_i - e_{i-1}|)$

Problem 3

- Mo’s – 쿼리 정렬 + 이전 구간 정보 이용하기

- 쿼리들 간 순서가 중요한가? -> No

- sort as:

1) 적절하게 겹치는 부분이 있도록
2) 겹치는 부분에 의한 구간이동이 적어지도록

쿼리의 시점이 유사한(같은 bucket에 들어가는) 것끼리 묶어서(=batch) 정렬!

Problem 3

- target: reduce $O(|s_i - s_{i-1}| + |e_i - e_{i-1}|)$

- case $O(|s_i - s_{i+1}|)$

1) $\frac{s_i}{\sqrt{N}} = \frac{s_{i+1}}{\sqrt{N}}$ 인 경우: 한 bucket내에 있으므로 $|s_i - s_{i+1}| \le \sqrt{N}$ -> $O(Q\sqrt{N})$
2) $\frac{s_i}{\sqrt{N}} \neq \frac{s_{i+1}}{\sqrt{N}}$ 인 경우: 서로 다른 $\sqrt{N}$개의 bucket에 대하여 -> $O(N\sqrt{N})$
   - 인접한 블록으로 이동하는 경우: $|s_i - s_{i+1}| \le 2\sqrt{N}$, 최대 $\sqrt{N}$번 발생
   - $|s_i - s_{i+1}| \le N - 1$: 구간이 늘어날수록 경우의 수가 감소

Problem 3

- target: reduce $O(|s_i - s_{i-1}| + |e_i - e_{i-1}|)$

- case $O(|e_i - e_{i+1}|)$

1) $e_i, e_{i+1}, \ldots, e_j$가 같은 batch($\frac{s_k}{\sqrt{N}}$ all same for $i \le k \le j$) 인 경우 -> $O(N\sqrt{N})$

   $\because \sum|e_i - e_{i+1}| \le N$, $\sqrt{N}$단위의 batch

2) $\frac{s_i}{\sqrt{N}} \ne \frac{s_{i+1}}{\sqrt{N}}$ 인 경우: 최대 $\sqrt{N}$번 등장, $|e_i - e_{i-1}| \le N$ -> $O(N\sqrt{N})$

Problem 3

- Mo's – 쿼리 정렬 + 이전 구간 정보 이용하기

- sort as: $\left\{\frac{s_i}{\sqrt{N}}, e_i\right\}$ 기준

- cnt[i]가 0->1 또는 1->0인 경우 쿼리값 변화
- 단 첫 쿼리는 직접 구하자.

Problem 4

- $N \times N$크기의 격자
- 같은 값이 들어있는 두 칸 사이의 최소 맨해튼 거리

Problem 4

- solution 1: 값이 v인 모든 칸에 대해 brute forcing

값이 v인 칸 k개에 대하여

$$O(k^2)$$

- k가 커질 경우(모두 같아질 경우): $\leq O(N^4)$

Problem 4

- solution 2: 값이 v인 모든 칸에 대해 multi-source BFS 수행

서로 다른 값의 개수 p에 대하여

$$O(p \times N^2)$$

- p가 커질 경우(모두 다른 경우): $\leq O(N^4)$

Problem 4

- 두 solution 합치기: 같은 값을 갖는 정점의 개수(k=$\sqrt{N^2}$)에 따른 case work

- 개수가 적은 경우: brute forcing -> $O\left(N^2 \times \frac{N^2}{N}\right) = O(N^3)$
- 개수가 많은 경우: multi-source BFS -> $O(N^3)$
  $k \geq N$인 케이스가 최대 N번 등장

Problem 5

- N개의 돌, 둘이서 돌게임
- 번갈아가며 돌을 가져가되, 직전 턴 사람이 k개를 가져갔다면 다음 턴 사람은 k 또는 k+1을 가져가야 한다. 돌을 가져갈 수 없다면 진다.
- 둘다 최선을 다할 때 누가 이길까?

- N이 크다면?($\leq 10^5$)
- 단발성 문제가 아니라 쿼리 문제라면?

Problem 5

- dp(i,j,t) := i개의 돌이 남아있고 직전 턴에 j개를 가져갔을 때 사람 t가 이길 수 있는가?

dp(i,j,t) = 1 – min(dp(n-k,k,t^1), dp(n-k-1,k+1,t^1))

but $i \le N, j \le N, t = 0, 1$
상태공간이 너무 크다..

Problem 5

- $i \leq N, j \leq N, t = 0, 1$에서 정말 $j \leq N$일까?

마지막으로 draw하는 개수를 k라 할 때

$$\sum_{i=1}^{k} i = \frac{k(k+1)}{2} \leq total\ draw \leq N$$

$$\Downarrow$$

$$k^2 \leq 2N, k \leq \sqrt{2N}$$

정리

- summary
- problem set

Summary

Problem set

- BOJ2042 – 구간 합 구하기
- BOJ16404 – 주식회사 승범이네
- BOJ13547 – 수열과 쿼리 5
- CF797E – Array Queries
- BOJ21090 – Trade
- BOJ 8462 – 배열의 힘
- BOJ 20297 - Confuzzle
- BOJ 13545 – 수열과 쿼리 0

- JOI2017 – Bitaro's Party
- POI2017 – Shipping containers, solution
- APIO2019 – Bridges, solution
- IOI2011 – Elephants, solution
- BOJ13518 – 트리와 쿼리 9, solution
- BOJ16124 – 나는 행복합니다