#6주차 고급반

Line Sweeping

선수내용: 이진 자료구조(Segment Tree, BIT(Fenwick Tree) etc.)

Lazy Propagation, 전구 끄기 문제(순서 강제 테크닉)

알면 좋은 내용: 좌표압축

T. 장형준

목표

- 생각하는 방법을 익힌다.
- 구현은 덤.

개념

Line Sweeping

‘생각’에 집중해주세요.

문제 상황

- 개념을 설명하기 위한 예시입니다.
  - 이 문제 자체가 중요한 것이 아닙니다.

- Closest pair of points problem
  - N개의 점이 주어졌을 때, 가장 가까운 두 점을 구하는 문제
  - Naive하게 O(N ^ 2). (: 모든 점 쌍을 비교해본다.)

개선 0 – 시간 복잡도를 생각해보기

- 점들을 다 보기는 해야하니 O(N)은 무조건 든다.
- 하지만 O(N) (: N^2 / N) 만큼의 시간복잡도가 더 생긴다.
- 더 줄일 수는 없을까?

개선 1 – 순서를 강제한다.

- 참고) noj.am/14927 (전구 끄기)

- 순서를 강제해서 생각을 간단하게 해보자.

- 점들에게 줄 수 있는 순서가 뭐가 있을까?
  - 입력 순서
  - x좌표
  - y좌표
  - 중앙으로부터 거리 etc.

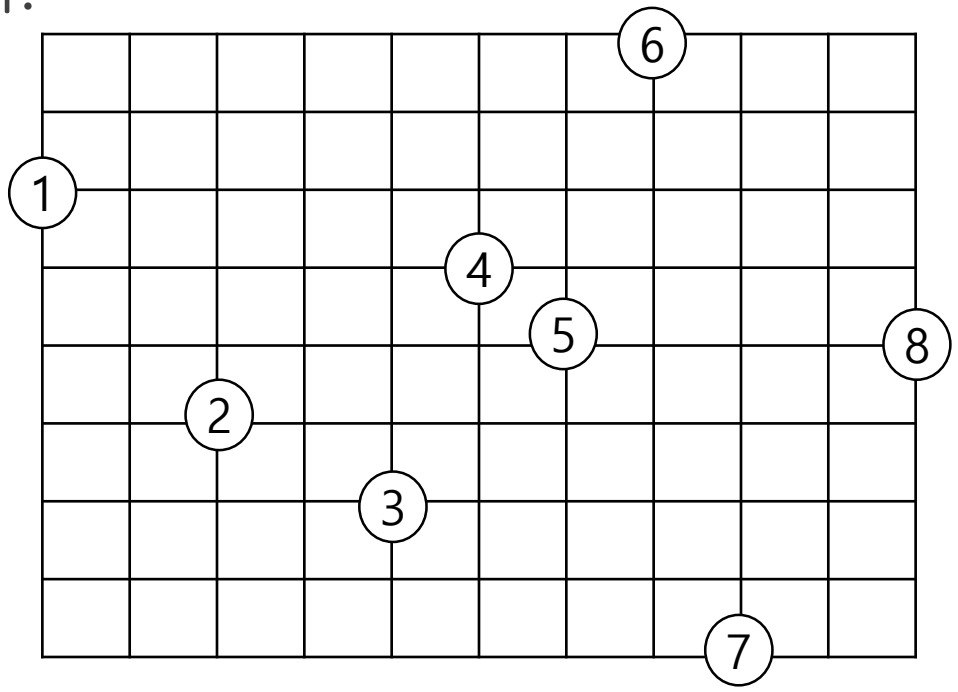
개선 2 – 효율적인 순서 찾기

- 입력 순
  - 비효율적
  - 지금 케이스를 입력순으로 정렬해서 이득을 얻을 수 있다고 가정하자.
  - 지금 케이스를 무작위로 섞은 케이스도 입력될 수 있다.
  - 모든 케이스가 이득을 얻을 수 있다 => 순서를 강제한 의미가 없다.

- 좌표 순
  - x, y 좌표
  - 입력을 무작위로 섞어도 동일한 기준으로 정렬된다. (일관된 순서)
  - 감이 온다. 다음 슬라이드에서 더 생각해보자.

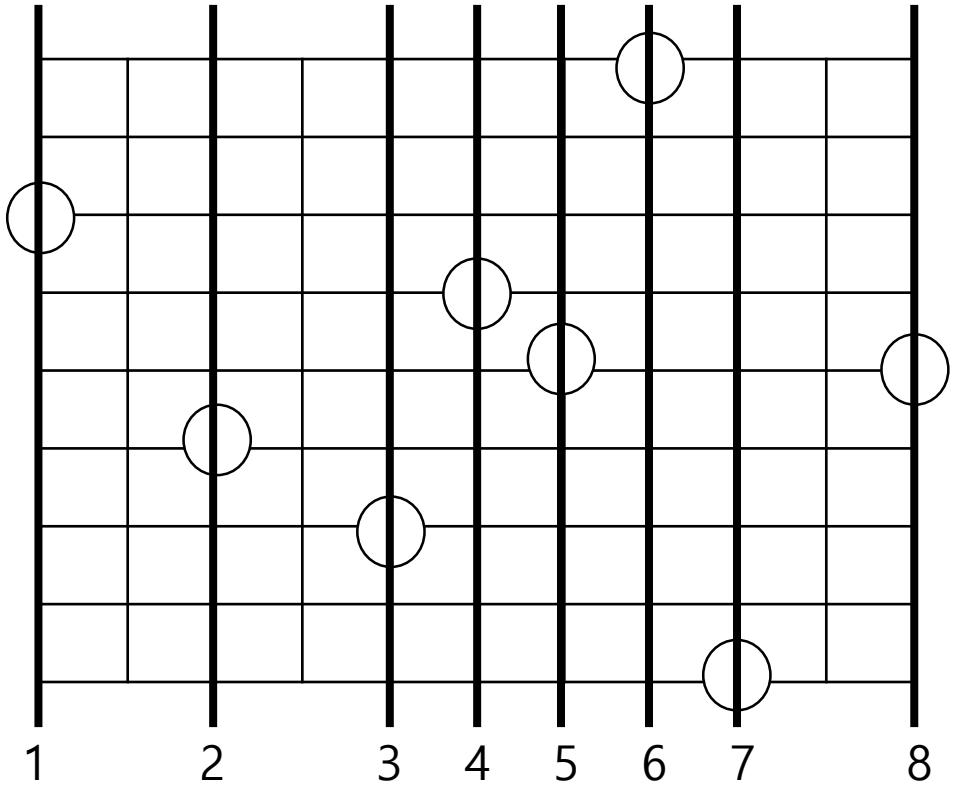
개선 3 – 좌표순으로 처리한다면?

- 다음과 같이 처리된다.
  - 임의의 예시입니다.



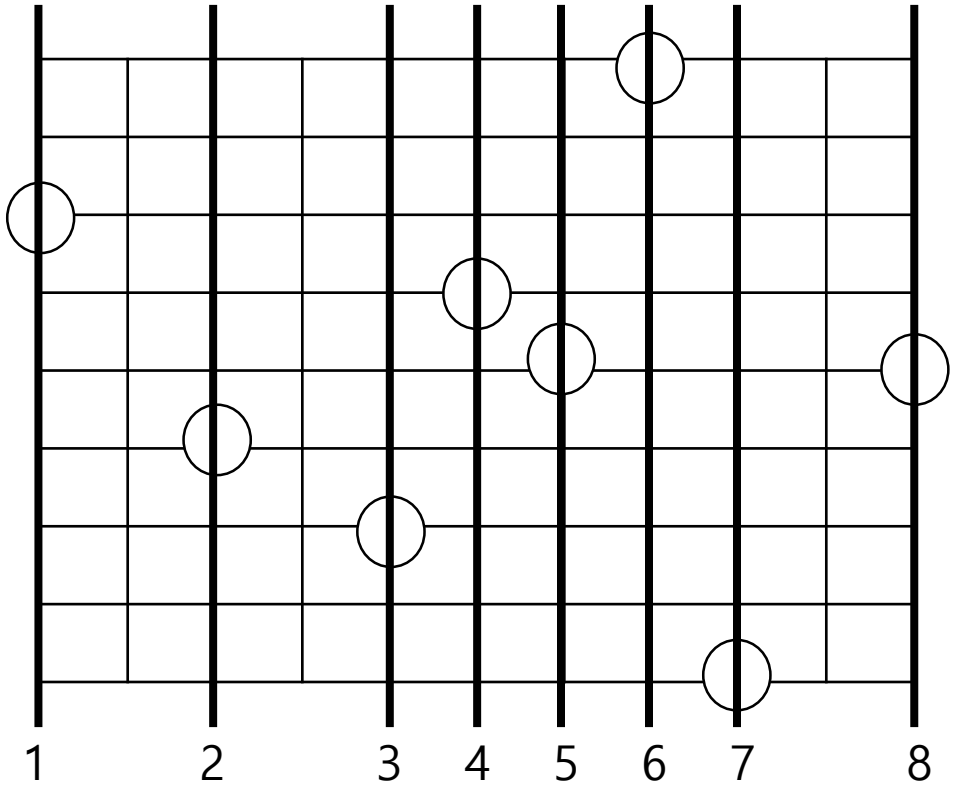
개선 3 – 좌표순으로 처리한다면?

- 이렇게 처리된다고 생각할 수도 있다.



이름의 유래 – Line Sweeping

- 순서를 강제하고 보니 선(Line)으로 쓸고 가는(Sweeping) 것 같다.



과연 효율적일 수 있는가

- 선(Line)은 이진 자료구조를 사용해서 lgN만에 처리할 수 있습니다.

- 는 조금 쉬었다가.

- 이것보다 직관적인 다른 예시로 알아볼 거예요.
  - Closest pair는 개념 설명용
  - 이게 어떻게 효율적인지 감 잡는 건 또 다른 예시
  - 구현은 두 예시 모두 해볼게요.

제발 쉬어요

어떻게 개선되는가?

Line Sweeping

이해까지 한 달음에 달려갑니다.
그 전에 좀만 더 쉬어요.

문제상황 – 직사각형의 면적 구하기

- 참고) noj.am/3392 화성 지도

- 직사각형의 합집합의 넓이를 구한다.
  - N개의 직사각형이 주어진다. (N <= 1e4)
  - 직사각형은 좌표평면 위에 축과 평행하게 주어진다.
  - 직사각형의 합집합의 넓이를 구한다.
  - 직사각형의 크기는 크다.
  - 유사) noj.am/2563 색종이

문제 해결 방법 - 의식의 흐름

- 직접 색칠 – 직사각형이 커서 실패
- 좌표 압축 후 색칠 - 1e4^2 = 1억. 터진다.
- 좌표 압축 후 쿼드트리 || 2차원 세그 – 똑같이 터진다.
- I&E(포함 배제) – 역시 N이 커서 무용지물.

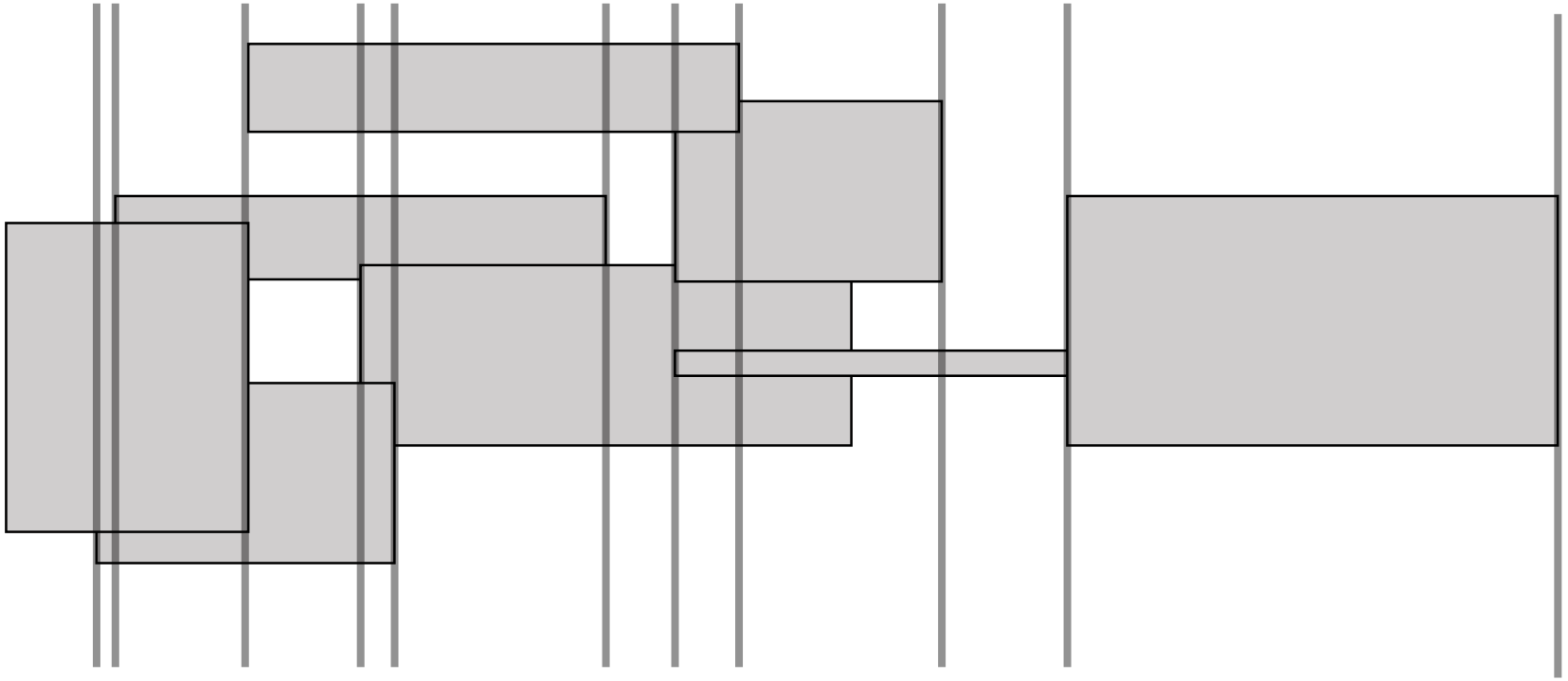
- 어떤 정보를 생각하지 않을 수 있을까?
- $S = \int y\,dx$. 면적만 구하면 되니까 y 이상의 정보는 필요하지 않은 걸?
  - ??!

문제 해결 방법 – 계산하지 않을 계획

- 시작과 끝이 같다면 직사각형이 몇 개가 있어도 문제 없다.
  - (가로길이) * (세로길이의 총 합)

문제 해결 방법 – 환원하기

- x 좌표 순으로 합집합 도형을 잘라보자.
- 직사각형의 끝 에서 세로길이의 총 합을 업데이트하는 문제가 된다.



문제 해결 방법 – 자료구조 적용하기

- 직사각형의 끝의 개수는 2N개이다.

- 선의 업데이트 – 좌표의 개수는 최대 30000이다.
  - 이진 자료구조를 사용하면 lgN으로 만들 수 있다.
  - 세그먼트 트리에 Lazy 사용하면 O(2N * 2lgN) = O(N lg N) !!!
  - 자세한 건 다음 슬라이드에서

해석 – 이게 뭔 소린가.

- 0. 가로 길이가 같다 && 세로 길이의 총 합을 안다

  => 넓이를 구할 수 있다.
- 1. 직사각형들의 왼쪽/오른쪽 x 좌표를 저장해 놓는다.
  - 왼쪽 경계면 추가, 오른쪽 경계면 삭제를 좌표와 묶어서 저장한다.
- 2. 그 좌표들을 정렬한다.
  - 추가/ 삭제도 같이 정렬된다.
- 3. 정렬된 배열을 돌면서 추가/삭제 업데이트를 해준다.
  - => 세로 길이의 총 합을 lg N 으로 구한다.
- 4. O(N^2) => O(N lg N) ??? Profit!

삐빅! 정상입니다.

구현을 보면 이해가 될지도 몰라요. 아니면 PPT를 반복해서 읽어보세요.
실전에서 쓸 수 있으려면 6개월에서 1년 걸리는 건 다반사!
기 죽지 말고 모두들 파이팅입니다요!
한번에 이해했으면 아마 이미 굇수일거야...
찬찬히 곱씹어가며 있으면 체화될 거에요.
일상생활에 응용해보는 것도 팁 (“어? 저거 라인스위핑으로 되겠는데?”)
주의 0: 혼모노 취급을 받을 수 있습니다.
주의 1: 주변에 혼모노라고 깔 친구도 없으면 혼자 웃다 슬퍼집니다.

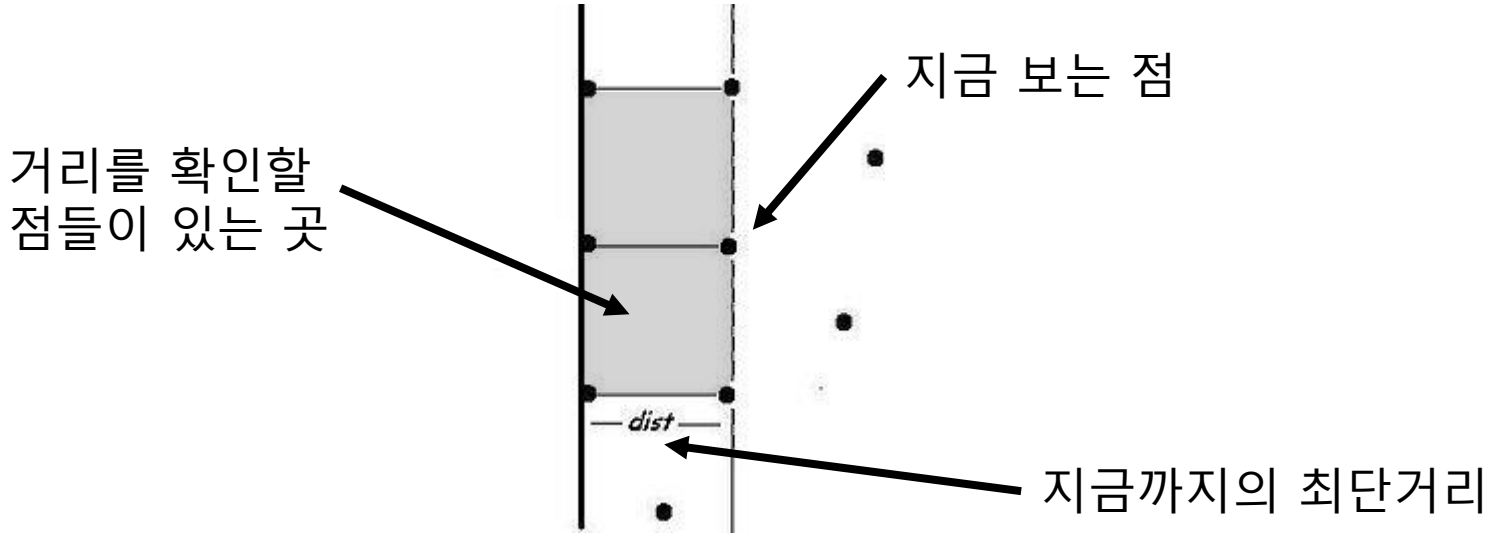
구현

Line Sweeping

Closest pair – 핵심

- 0. pair< x, y > 순으로 정렬한다.

- 1. 지금까지의 최단거리 * 최단거리 박스 안의 점만 본다.



Closest pair - 테크닉

- set을 사용한다.
- 슬라이딩 윈도우로 set에서 점을 뺀다.

Closest Pair - 코드

- BOJ 2261

```cpp
#include <bits/stdc++.h>
#define X second
#define Y first
using namespace std;
using pii = pair< int, int >;

const int N = 1e5 + 5;
const int MAX_D = 1e4 + 5;
pii d[N];

int getDistSq( pii a, pii b ) {
    int dx = a.X - b.X;
    int dy = a.Y - b.Y;
    return dx * dx + dy * dy;
}

int main()
{
    ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    int n; cin >> n; for( int i = 0; i != n; i++ ) cin >> d[i].X >> d[i].Y;
    sort( d, d + n, []( pii a, pii b ) { return a.X != b.X ? a.X < b.X : a.Y < b.Y; } );
    int p = 0, distSq = getDistSq( d[0], d[1] );
    set< pii > st; st.insert( d[0] ); st.insert( d[1] );
    for( int i = 2; i != n; i++ ) {
        pii& currDot = d[i];
        int dist = (int)ceil(sqrt(distSq));
        auto beg = st.lower_bound( {currDot.Y - dist, -MAX_D} );
        auto end = st.lower_bound( {currDot.Y + dist, MAX_D} );
        for( auto j = beg; j != end; j++ )
            distSq = min( distSq, getDistSq( currDot, *j ) );
        while( 1 ) {
            int dx = currDot.X - d[p].X;
            if( dx < dist ) break;
            st.erase( d[p++] );
        }
        st.insert( currDot );
    }
    cout << distSq;
    return 0;
}
```

직사각형들의 넓이 - 테크닉

- 사각형은 양쪽 끝이 있다.
  - 왼쪽 끝에서 더한 선분이 오른쪽 끝에서 빠진다.

- 겹친 사각형은 하나로 센다.
  - Lazy를 내려 줄 필요가 없다.
  - 위의 조건과 연계해서, Lazy != 0 만 검사하면 된다.

직사각형들의 넓이 - 코드

- BOJ 3392

```cpp
#include<bits/stdc++.h>
using namespace std;

const int N = 1e5 + 5;
struct Query{ int x, y1, y2, val; };
Query q[N];
int lazy[12*N], dt[12*N];
void update(int l,int r,int nowl,int nowr,int idx,int val){
    if( nowr < l || r < nowl ) return;
    if( l <= nowl && nowr <= r) {
        lazy[idx] += val;
        if( nowl == nowr ) {
            dt[idx] = !!lazy[idx];
            return;
        }
    }
    else{
        int m = ( nowl + nowr ) / 2;
        update( l, r, nowl, m, idx*2, val );
        update( l, r, m+1, nowr, idx*2+1, val );
    }
    if( lazy[idx] ) dt[idx] = (nowr - nowl + 1);
    else dt[idx] = dt[idx*2] + dt[idx*2+1];
}
int getSumY(){ return dt[1]; }
int main()
{
    int n; cin >> n;
    for(int i=0;i!=n;i++){
        int xs,ys,xe,ye; cin >> xs >> ys >> xe >> ye;
        q[ i*2 ]   = { xs, ys, ye-1,  1 };
        q[ i*2+1 ] = { xe, ys, ye-1, -1 };
    }
    sort( q, q+2*n, []( Query a, Query b ) { return a.x < b.x; } );
    long long int ans = 0;
    for( int i = 0; i != 2*n - 1; i++ ) {
        int dx = q[i+1].x - q[i].x;
        update( q[i].y1, q[i].y2, 0, 30000, 1, q[i].val );
        ans += 1ll * dx * getSumY();
    }
    cout << ans;
    return 0;
}
```

끝