#05-3. 가장 가까운 두 점

나정휘

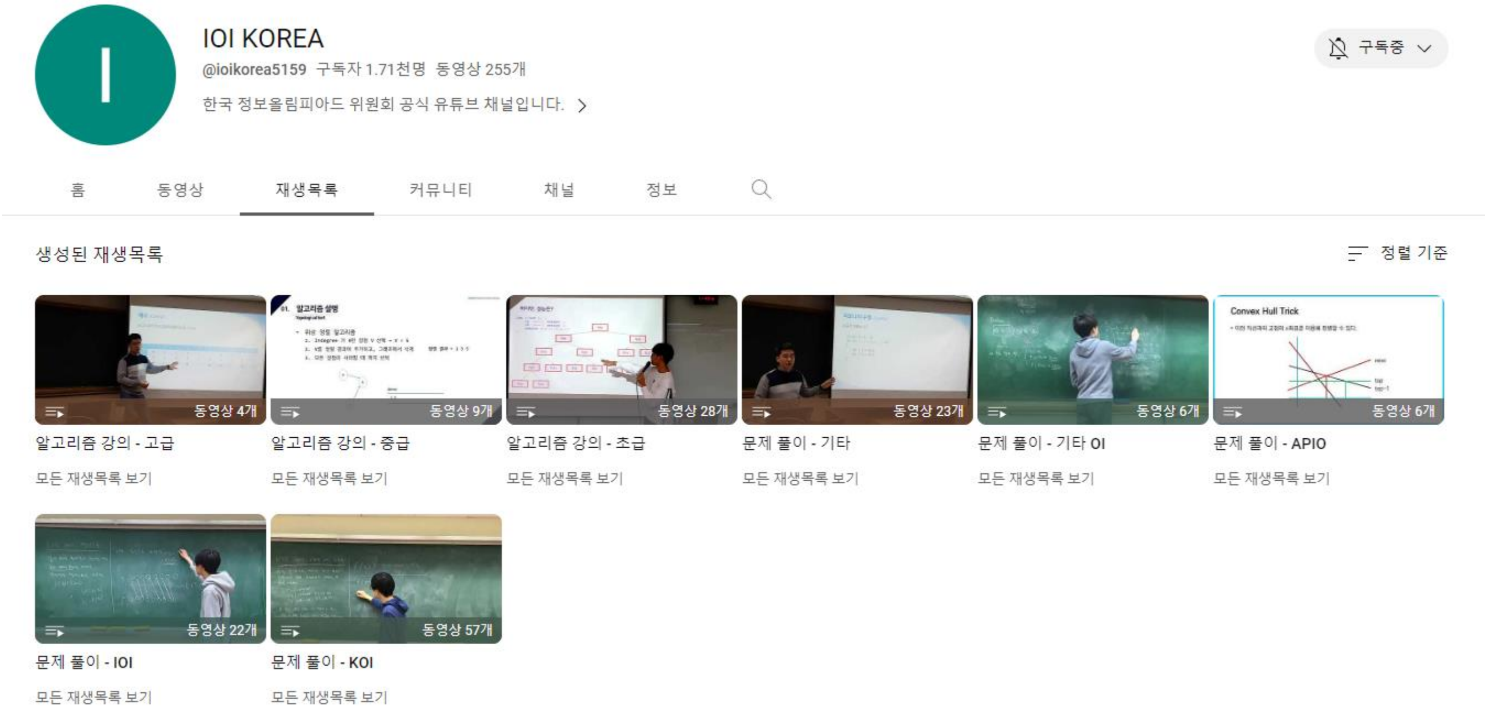
https://justiceHui.github.io/

동영상 강의

동일한 내용을 설명하는 영상이 있음

- 가장 가까운 두 점 (https://youtu.be/lv-KOgzQ-G8)
  - 스위핑 기법을 이용해 구하는 방법도 다룸

- 재귀 함수 말고도 다양한 영상이 있으니 많은 관심 부탁…
- 한국정보올림피아드위원회 공식 유튜브 채널임



목차

문제 소개

격자의 성질

$O(N \log^2 N)$ 분할 정복 알고리즘

$O(N \log N)$ 분할 정복 알고리즘

문제 소개

2차원 평면에 N개의 점이 주어지면 가장 가까운 두 점을 구하는 문제

- 이때 두 점 $(x_1, y_1), (x_2, y_2)$의 거리는 $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$으로 정의

- 단순하게 구현하면 $O(N^2)$
- 이번 슬라이드의 목표는 문제를 $O(N \log^2 N)$ 또는 $O(N \log N)$ 시간에 해결하는 것

격자의 성질

격자의 성질

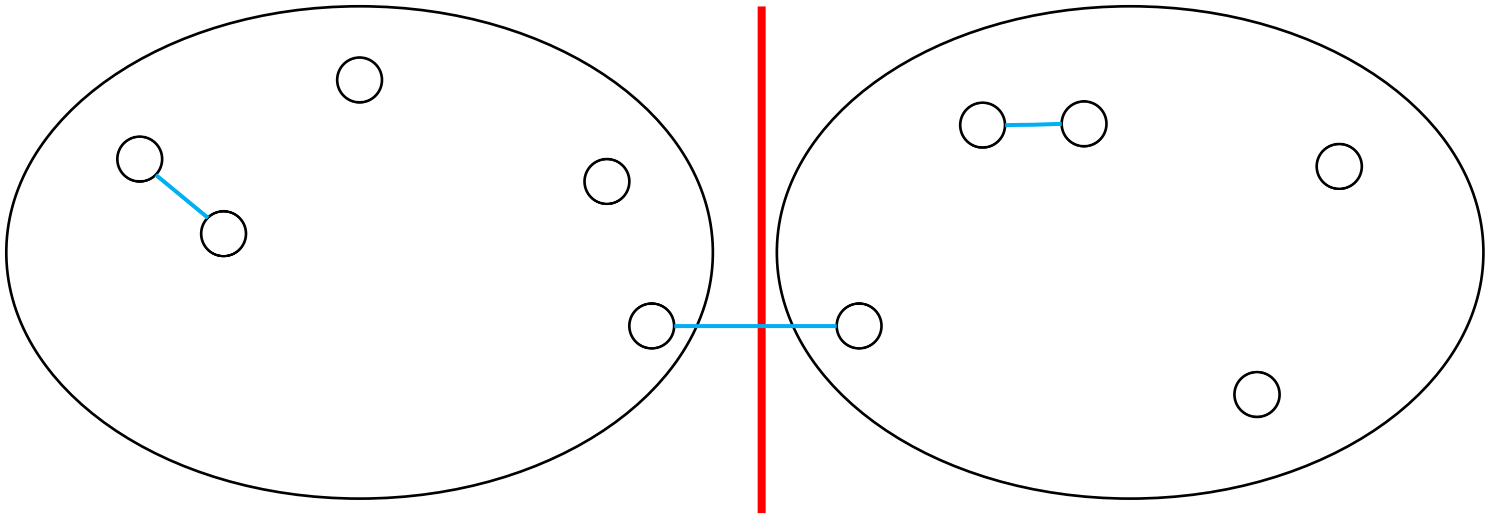
- N개의 점 중 가장 가까운 두 점의 거리를 δ라고 하자.
- 수직/수평선을 이용해 평면을 $\frac{\delta}{2} \times \frac{\delta}{2}$ 크기의 박스로 분할하면 아래 두 가지 사실을 알 수 있음
  - 각 박스에는 최대 1개의 점만 들어있음
    - 한 박스에서 점 2개를 최대한 멀리 찍어도 $\delta/\sqrt{2}$보다 멀어질 수 없음
    - 가장 가까운 두 점의 거리가 $\delta > \delta/\sqrt{2}$이므로 한 박스에 2개의 점이 존재할 수 없음
  - 두 점 p_1, p_2의 거리가 δ이면, 두 점이 속한 박스는 가로/세로 방향으로 각각 최대 2칸 떨어져 있음
    - 만약 두 점이 속한 박스가 가로 방향으로 3칸 이상 떨어져 있다면 $|p_2.x - p_1.x| > \delta$
    - $dist(p_1, p_2) \geq |p_2.x - p_1.x| > \delta$이므로 모순

분할 정복 알고리즘

알고리즘의 개요

- 점들을 x좌표 오름차순으로 정렬하자.
- 분할: 적당한 x_m에 대해, 직선 $x = x_m$을 기준으로 두 개의 부분 문제로 분할 후 각각 해결
- 정복: 직선 $x = x_m$을 기준으로 서로 반대편에 있는 점들의 거리를 고려

- 목표: $T(N) = 2T(N/2) + O(N \log N) \in O(N \log^2 N)$
  - $O(N \log^2 N)$ 알고리즘을 배우면 $O(N \log N)$은 쉬움



분할 정복 알고리즘

분할 단계

- 양쪽에 균등하게 점이 나눠지도록 x_m을 잡자.
  - x좌표 오름차순으로 정렬했을 때 중앙에 오는 점의 x좌표
- $T(N) = 2T(N/2) + f(N)$
- $f(N) = O(N \log N)$이 되도록 만들어야 함

정복 단계

- 두 부분 문제에서 계산한 최소 거리를 δ_s라고 하자.
  - 정복 단계에서는 $|x - x_m| < \delta_s$인 점만 고려해도 됨
  - 또한 $|y_1 - y_2| \geq \delta_s$인 두 점 $(x_1, y_1), (x_2, y_2)$는 고려하지 않아도 됨
- 두 조건을 모두 만족하는 점들의 쌍은 몇 개일까?

분할 정복 알고리즘

정복 단계

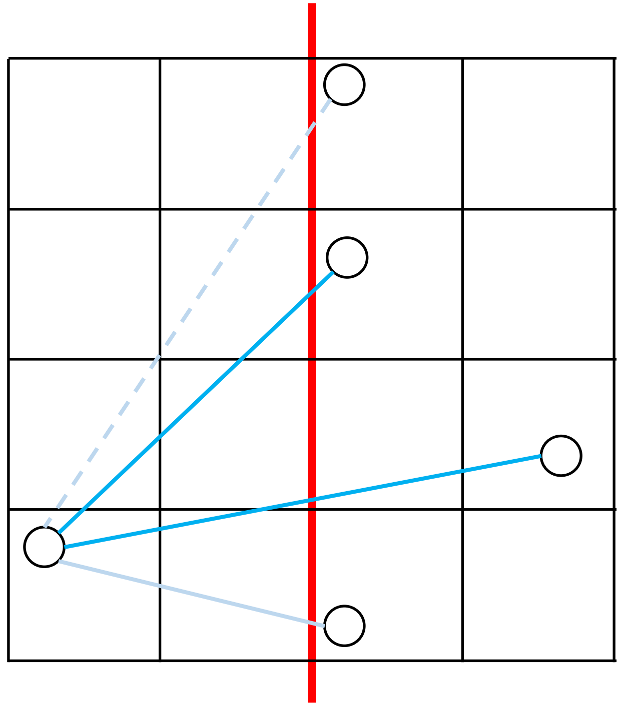
- x좌표가 x_m이하인 점들 중 가장 가까운 두 점의 거리를 δ_l
- x좌표가 x_m초과인 점들 중 가장 가까운 두 점의 거리를 δ_r이라고 하자.
- $\delta_s = \min(\delta_l, \delta_r)$

- 평면을 $\frac{\delta_s}{2} \times \frac{\delta_s}{2}$ 크기의 박스로 분할
  - x, y좌표 차이가 δ_s 이하인 점들을 보는 것은 가로/세로 방향으로 최대 2칸 떨어진 박스만 보는 것
  - 각 박스에는 최대 1개의 점이 있으므로 최대 16개의 점만 보면 됨
  - 따라서 확인해야 하는 쌍의 개수는 $O(N)$개

분할 정복 알고리즘

$O(N)$개의 쌍만 확인하는 방법

- $|x - x_m| < \delta_s$인 점을 모두 모아서 y좌표 오름차순으로 정렬한 다음
- 각각의 점 (x, y)를 차례대로 보면서
- $0 \le y' - y < \delta_s$를 만족하는 점 (x', y')만 확인하면 됨



분할 정복 알고리즘

분할 정복

- 전체 시간 복잡도는 $T(N) = 2T(N/2) + O(N\log N) \in O(N\log^2 N)$
  - y좌표 오름차순으로 정렬하는데 $O(N\log N)$ 소요

- $T(N) = 2T(N/2) + O(N) \in O(N\log N)$ 방법
  - $f(l,m)$과 $f(m,r)$이 종료되면$[\mathrm{l},\mathrm{m})$, $[m,r)$ 구간의 점은 각각 y좌표 오름차순으로 정렬된 상태
  - 합병 정렬의 방법을 이용해 두 구간의 점을 $O(N)$ 시간에 y좌표 오름차순으로 정렬할 수 있음

분할 정복 알고리즘

```cpp
#include <bits/stdc++.h>
#define x first
#define y second
using namespace std;
using ll = long long;
using Point = pair<ll, ll>;

inline ll Square(ll v){ return v * v; }
inline ll Dist(const Point &p1, const Point &p2){
    return Square(p2.x - p1.x) + Square(p2.y - p1.y);
}
struct CompareY{
    bool operator () (const Point &p1, const Point &p2) const {
        return tie(p1.y,p1.x) < tie(p2.y,p2.x);
    }
};

int N;
Point A[101010], B[101010];

ll DnC(int l, int r){
    if(l == r) return LLONG_MAX;
    int m = (l + r) / 2;
    ll xm = A[m].x, d = min(DnC(l, m), DnC(m+1, r));
    merge(A+l, A+m+1, A+m+1, A+r+1, B+l, CompareY());
    copy(B+l, B+r+1, A+l);

    vector<Point> P;
    for(int i=l; i<=r; i++) if(Square(A[i].x - xm) < d) P.push_back(A[i]);
    for(int i=0; i<P.size(); i++){
        for(int j=i-1; j>=0; j--){
            if(Square(P[i].y - P[j].y) < d) d = min(d, Dist(P[i], P[j]));
            else break;
        }
    }
    return d;
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N;
    for(int i=1; i<=N; i++) cin >> A[i].x >> A[i].y;
    sort(A+1, A+N+1);
    cout << DnC(1, N);
}
```

질문?