빙글빙글

나정휘

https://justiceHui.github.io/

목차

- 각도 정렬
- 불도저 트릭

각도 정렬

각도 정렬

- 각도 정렬
  - 2차원 평면에 N개의 점이 주어짐
  - x축의 양의 방향과 이루는 각도 오름차순으로 정렬
  - 당연히 atan2 같은 삼각함수 사용 불가

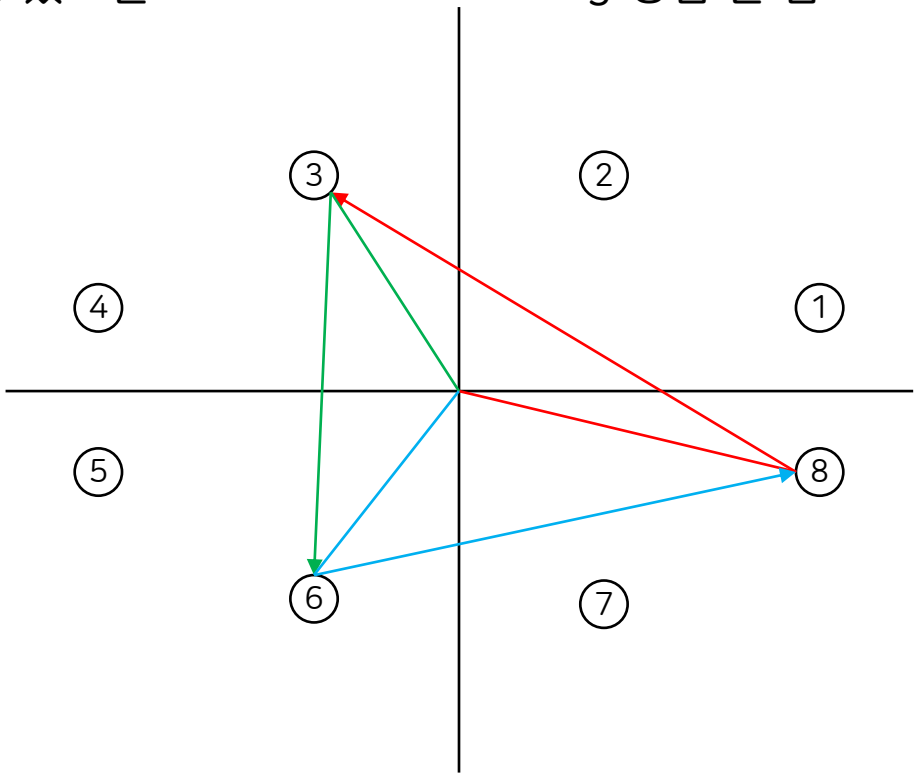
  - Graham Scan에서 사용한 정렬 방식을 그대로 사용하면 될까?

```
ll operator / (const Point &p1, const Point &p2){
    return p1.x * p2.y - p2.x * p1.y;
}

bool Compare(const Point &p1, const Point &p2){
    return p1 / p2 > 0;
}
```

각도 정렬

- 각도 정렬
  - Graham Scan의 정렬 방식의 한계
    - 모든 점이 y > 0을 만족하면 올바르게 정렬 가능
    - 하지만 y < 0인 점이 있으면 strict weak ordering 성립 안 함
      - 3 < 6 < 8 < 3



각도 정렬

- 각도 정렬
  - y = 0인 점이 없을 때의 해결 방법
    - 두 점의 y좌표 부호 먼저 비교
    - 부호가 다르면 y좌표가 양수인 점 먼저
    - 그렇지 않으면 return CCW({0,0}, a, b) > 0
  - y = 0인 점이 있을 때의 해결 방법
    - (0, 0) → (+, 0) → (x, +) → (-, 0) → (x, -) 순으로 정렬
  - 구현 코드 생략

각도 정렬

- BOJ 25051 천체 관측
  - 좌표가 정수인 2차원 점 N개 주어짐
  - 각 점은 가중치 S_i를 갖고 있음
  - P_i가 주어지면 원점을 중심으로 하는 중심각이 90도, 반지름이 $\sqrt{P_i}$인 부채꼴 중
  - 부채꼴에 포함되는 점들의 가중치 합의 최댓값을 구하는 문제

  - 풀이
    - N개의 점을 원점 기준으로 각도 정렬하고 거리가 $\sqrt{P_i}$ 초과인 점 제거
    - 부채꼴의 한쪽 각도를 고정한 뒤, 투 포인터 기법을 이용해 반대쪽 각도 결정
    - 가중치의 누적 합 배열을 이용해 각도 구간에 포함된 가중치의 합 계산

각도 정렬

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

struct Point{ ll x, y, v; };
ll Dot(const Point &p1, const Point &p2){ return p1.x * p2.x + p1.y * p2.y; }
ll Cross(const Point &p1, const Point &p2){ return p1.x * p2.y - p2.x * p1.y; }
ll Len(const Point &pt){ return Dot(pt, pt); }

ll N, Q, R=-9e18;
Point A[202020];
int Sign(ll v){ return (v > 0) - (v < 0); }

bool Compare(const Point &p1, const Point &p2){
    static constexpr int arr[9] = { 5, 4, 3, 6, -1, 2, 7, 0, 1 };
    int v1 = arr[Sign(p1.x)*3+Sign(p1.y)+4], v2 = arr[Sign(p2.x)*3+Sign(p2.y)+4];
    return v1 != v2 ? v1 < v2 : Cross(p1, p2) > 0;
}

void Solve(ll K){
    ll now = 0, mx = 0;
    for(int i=1, j=1; i<=N; i++){
        while(j < i+N && Dot(A[i], A[j]) >= 0 && Cross(A[i], A[j]) >= 0){
            if(Len(A[j]) <= K) now += A[j].v;
            mx = max(mx, now); j++;
        }
        if(Len(A[i]) <= K) now -= A[i].v;
    }
    R = max(R, mx - K);
}

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

질문?

불도저 트릭

불도저 트릭

- 정렬된 배열에서 할 수 있는 것
  - 이분 탐색, 투 포인터
  - A_i보다 작은 가장 큰 원소 → A_{i-1}, A_i보다 큰 가장 작은 원소 → A_{i+1}

- 2차원 평면의 점은 어떻게 정렬할 수 있을까?
  - x좌표 오름차순?
  - y좌표 오름차순?
  - y=x 직선에 사영한 위치 오름차순?
  - ...
  - 결국 적당한 직선에 사영한 위치에 대한 오름차순으로 정렬하게 됨

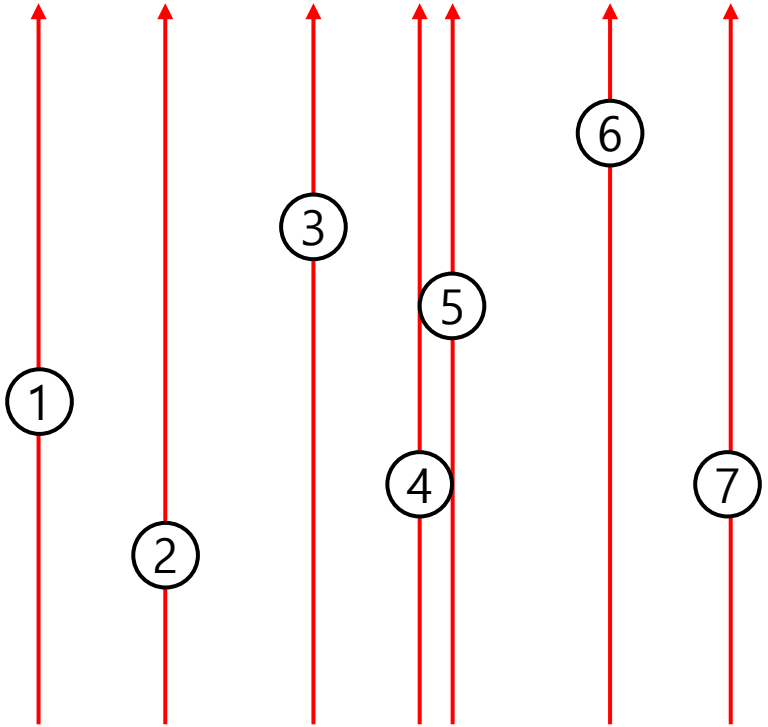
불도저 트릭

- 불도저 트릭의 용도
  - 축에 대해 정렬했을 때, 가능한 서로 다른 정렬 결과는 총 $O(N^2)$가지
  - 가능한 모든 정렬 결과를 $O(N^2 \log N)$에 순회할 수 있음

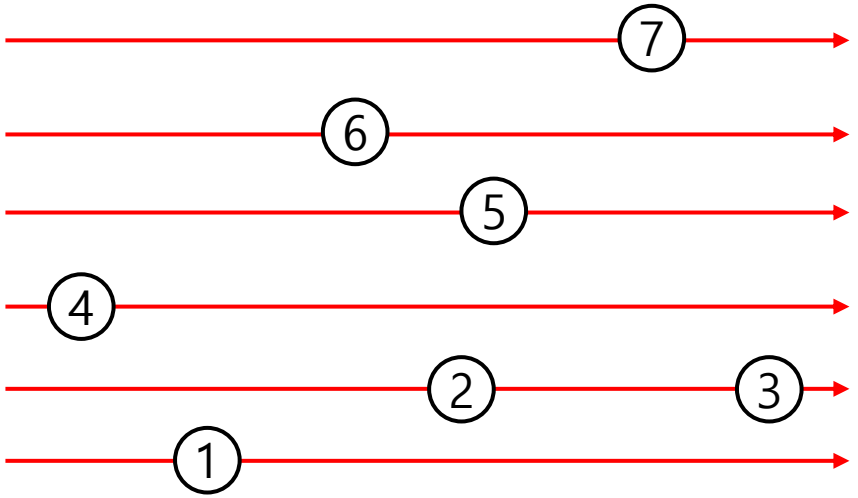
  - x축에 사영한 위치 오름차순으로 정렬
    - 기울기가 ∞인 직선이 왼쪽에서 오른쪽으로 이동하면서 만나는 순서대로 나열
  - y축에 사영한 위치 오름차순으로 정렬
    - 기울기가 0인 직선이 아래에서 위로 이동하면서 만나는 순서대로 나열
  - 기울기가 m인 직선에 사영한 위치 오름차순으로 정렬
    - 기울기가 -1/m인 직선이 한쪽 방향으로 이동하면서 만나는 순서대로 나열

불도저 트릭

- x 좌표 정렬: 기울기가 ∞인 직성 이동

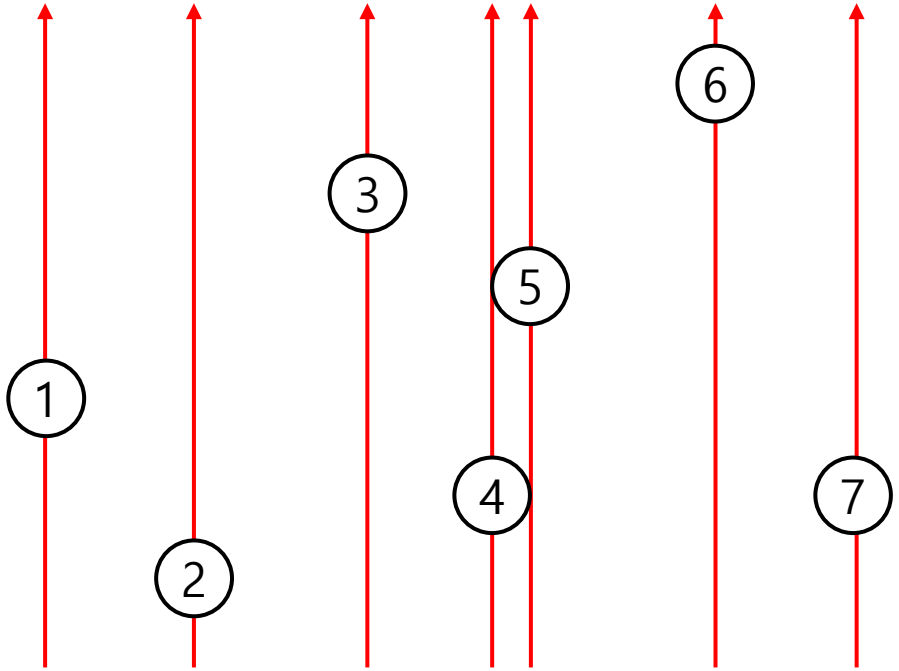


- y 좌표 정렬: 기울기가 0인 직선 이동



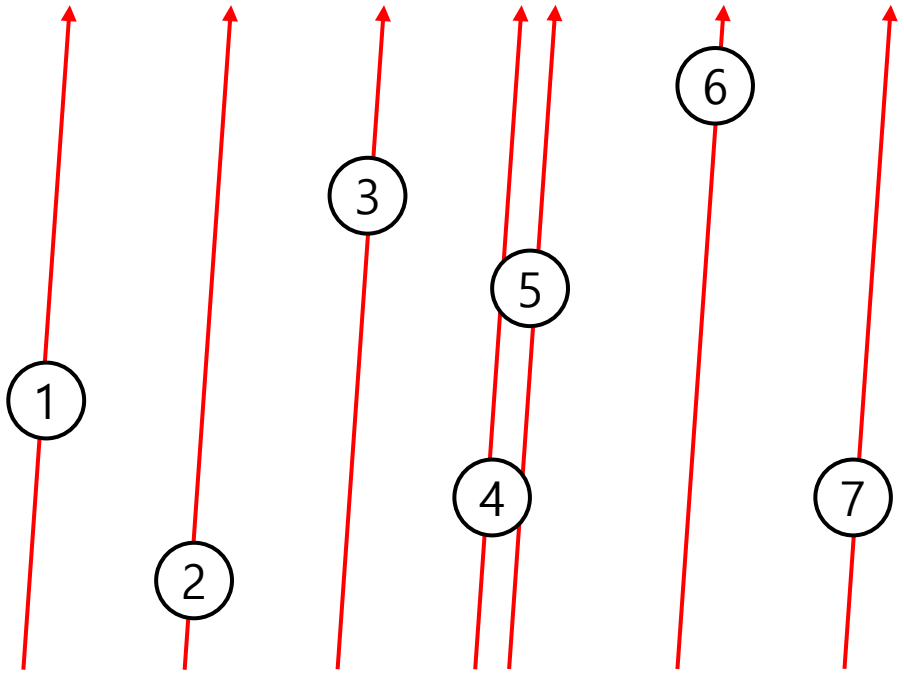
불도저 트릭

- 정렬 결과가 바뀌는 조건
  - 아주 조금 회전시키는 것으로는 정렬 결과가 바뀌지 않음



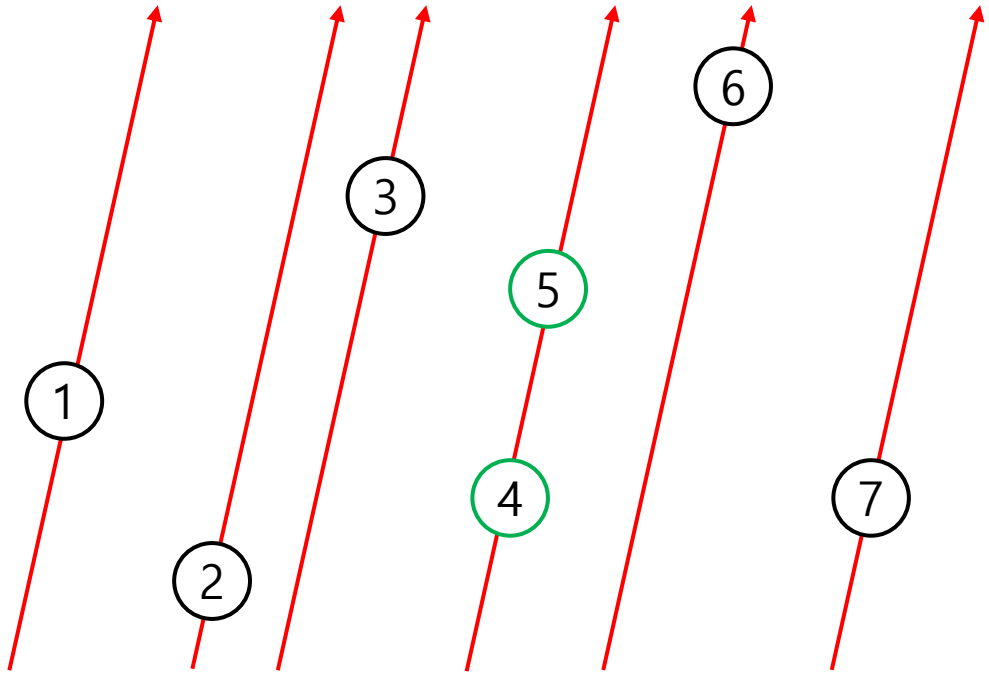
불도저 트릭

- 정렬 결과가 바뀌는 조건
  - 아주 조금 회전시키는 것으로는 정렬 결과가 바뀌지 않음



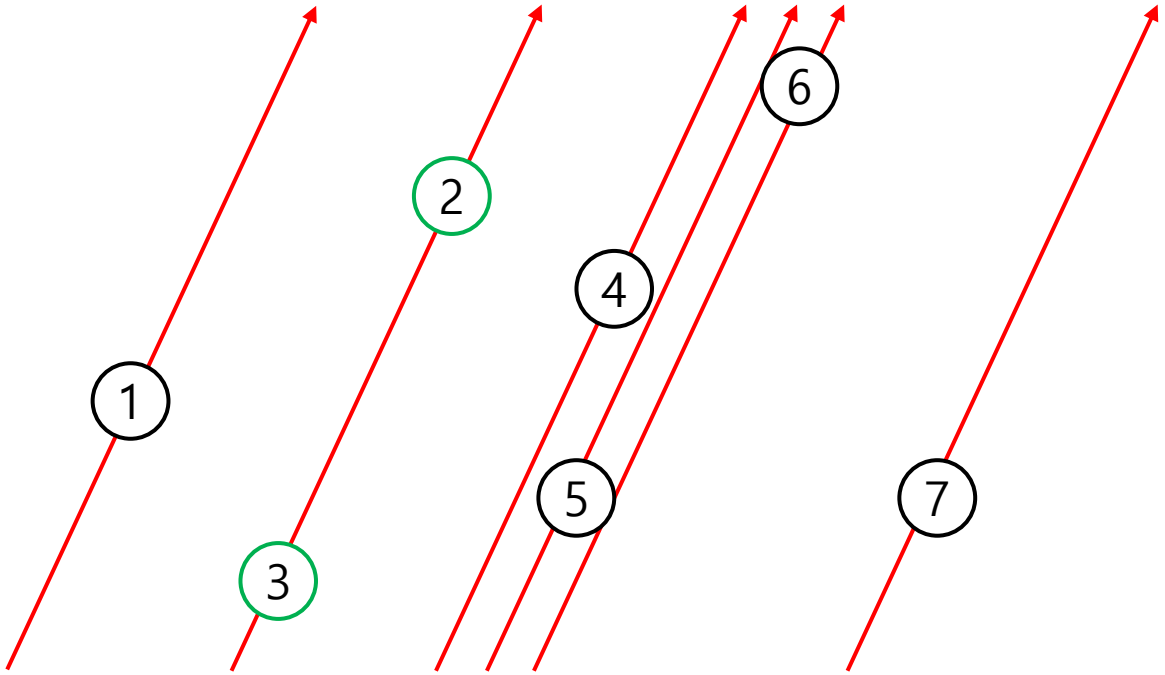
불도저 트릭

- 정렬 결과가 바뀌는 조건
  - 4번 점과 5번 점을 잇는 선분과 평행해지는 시점에 두 점의 우선 순위가 동등해짐
    - 반시계 방향으로 조금 돌리면 4번 점, 시계 방향으로 조금 돌리면 5번 점의 우선 순위가 높아짐



불도저 트릭

- 정렬 결과가 바뀌는 조건
  - 2번 점과 3번 점을 잇는 선분과 평행해지는 시점에 두 점의 우선 순위가 동등해짐



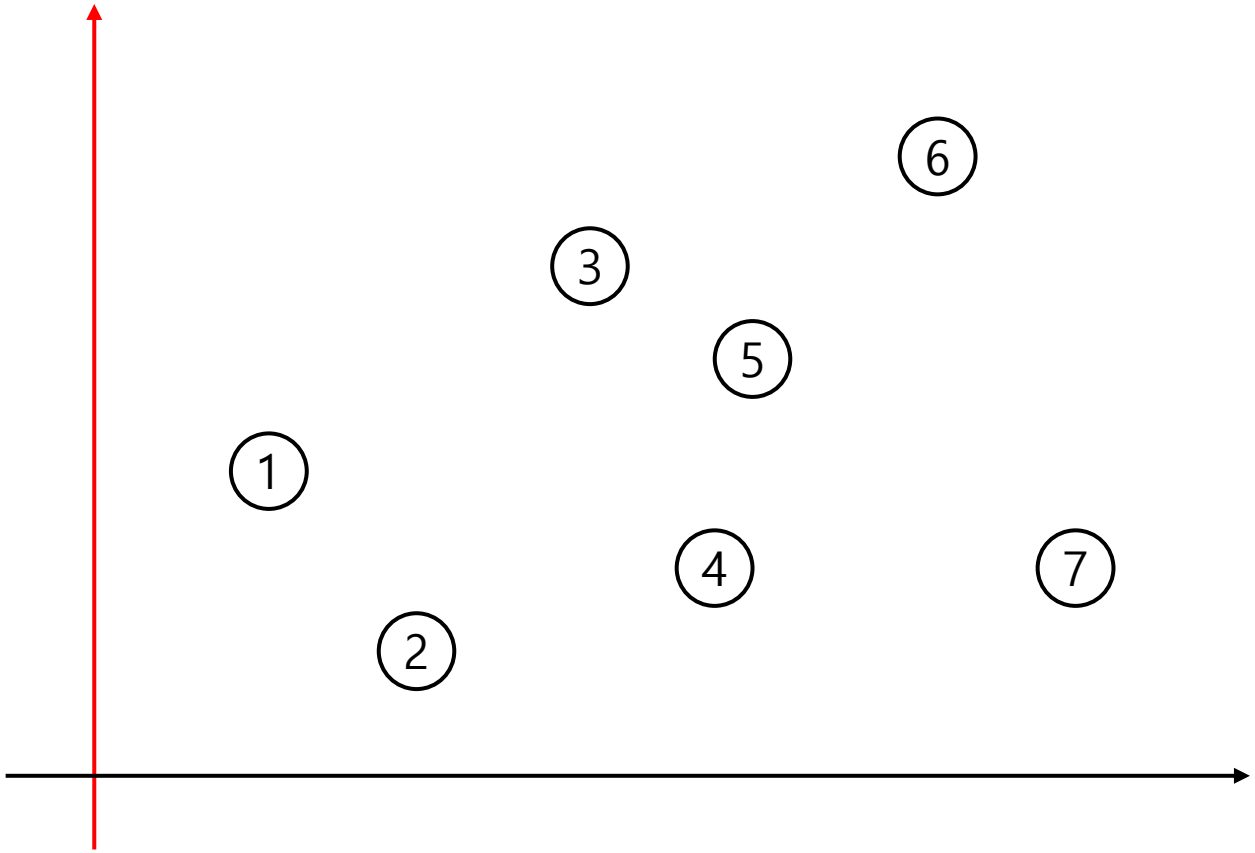
불도저 트릭

- 정렬 결과가 바뀌는 조건
  - 두 점을 잇는 선분과 평행해지는 시점에만 정렬 결과가 달라짐
  - 공선점이 없다면 인접한 두 점의 순서만 바뀜

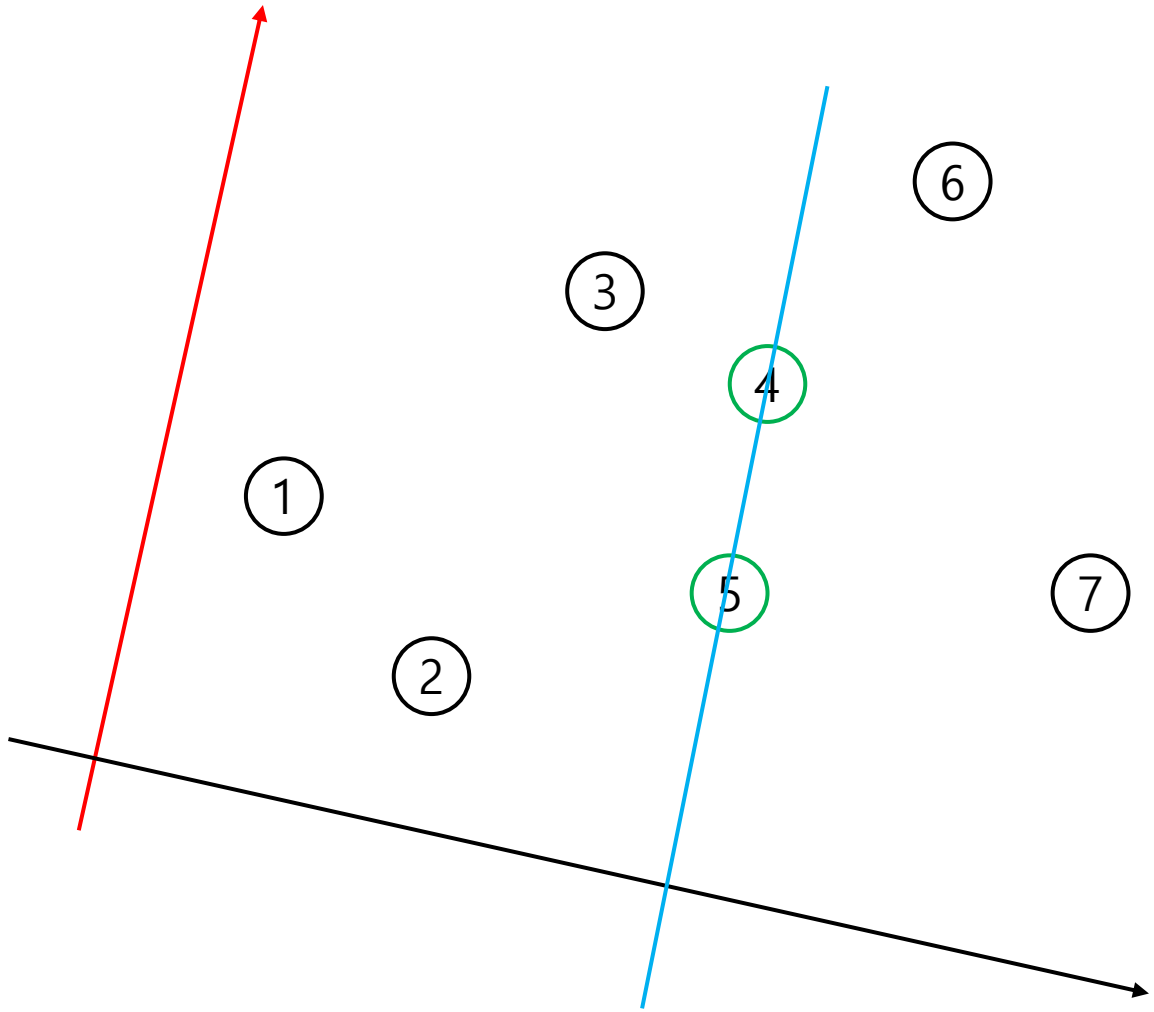
- 불도저 트릭
  - 두 점을 잇는 선분의 기울기만 고려해도 됨
  - 따라서 가능한 정렬 결과의 경우의 수는 $O(N^2)$
  - $O(N^2)$개의 선분을 모두 구해서 기울기 순서대로 순회
  - 두 점의 순서가 변경되는 상황에서 적당한 자료구조를 관리

질문?

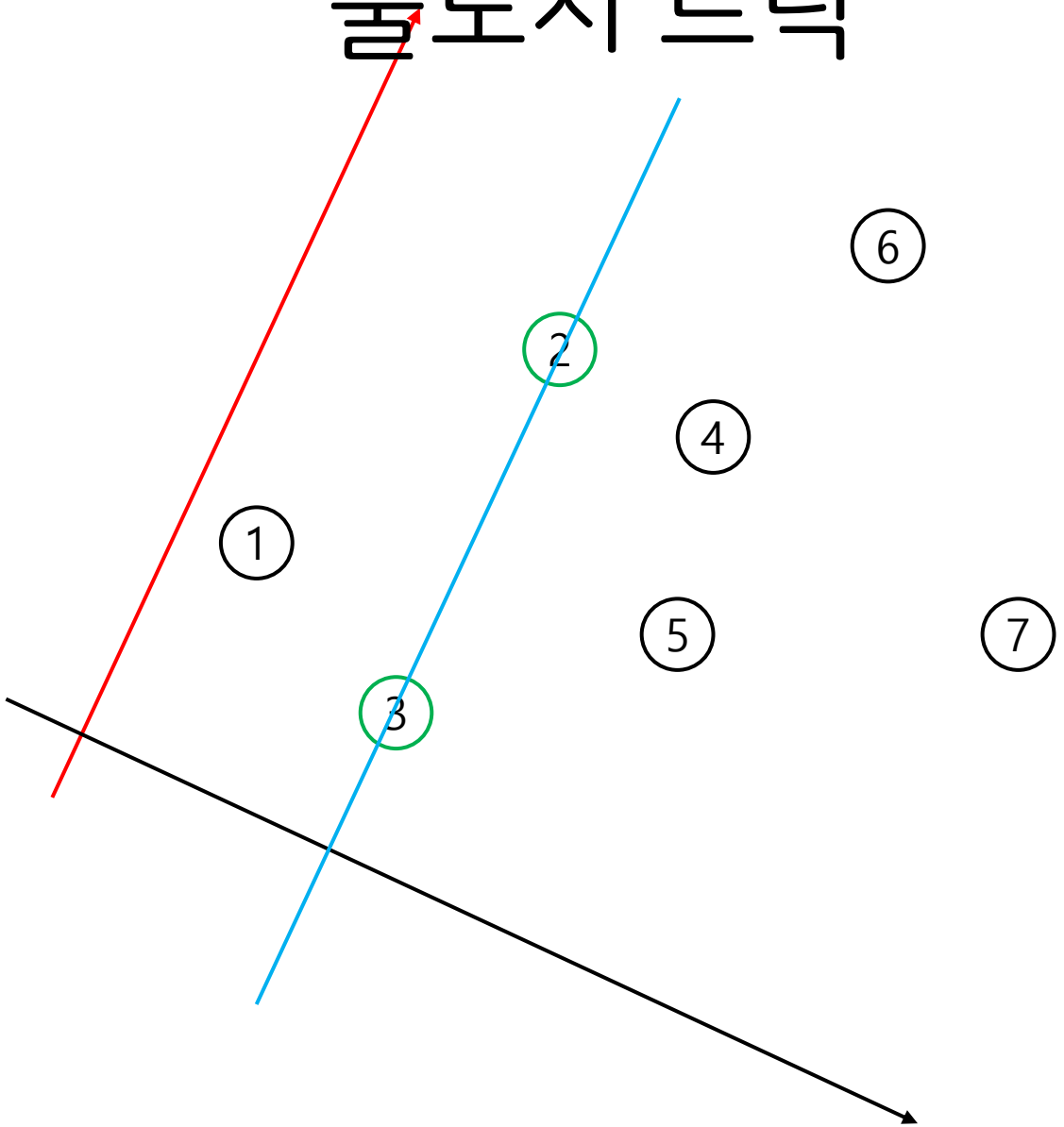
불도저 트릭



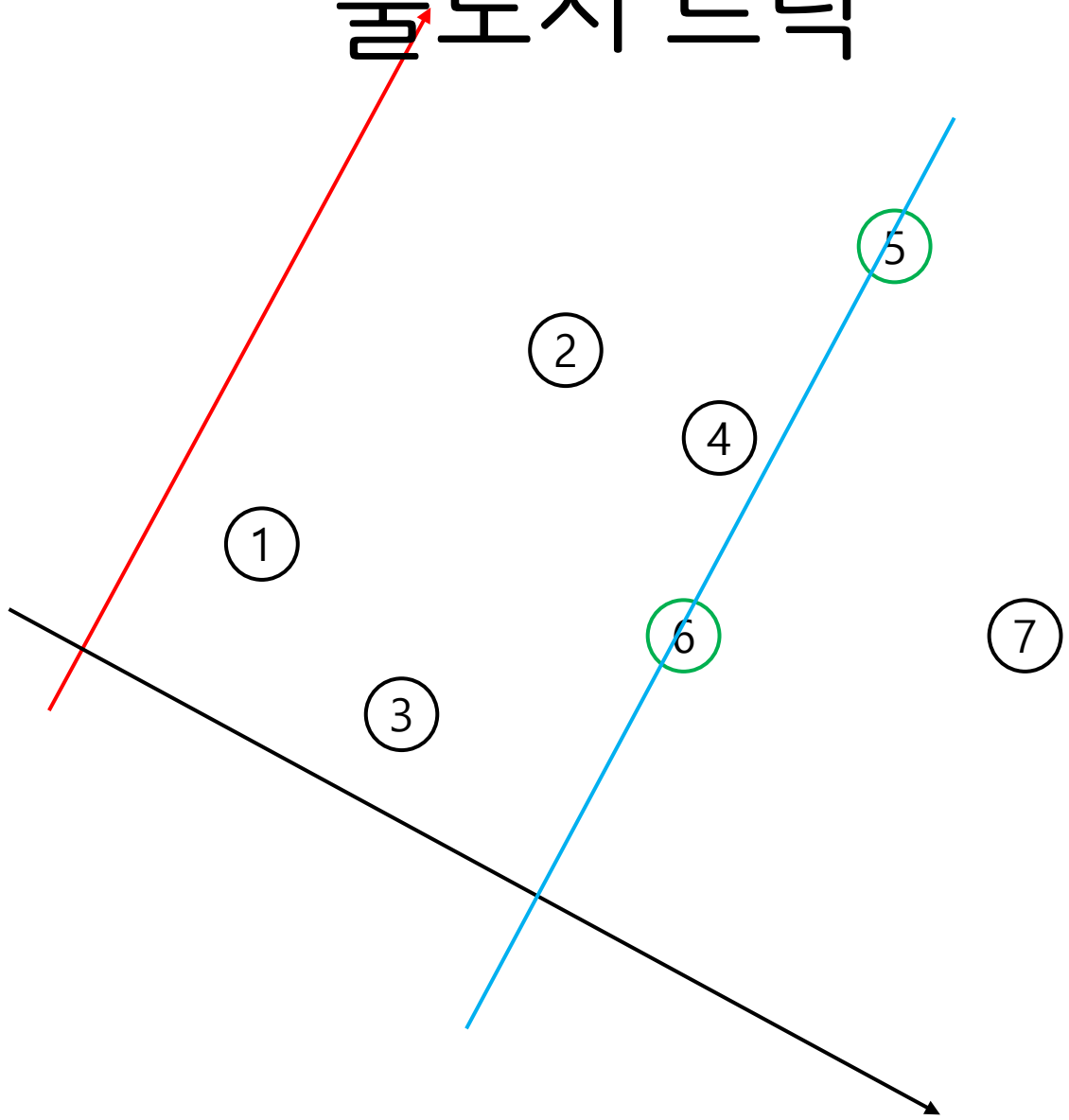
불도저 트릭



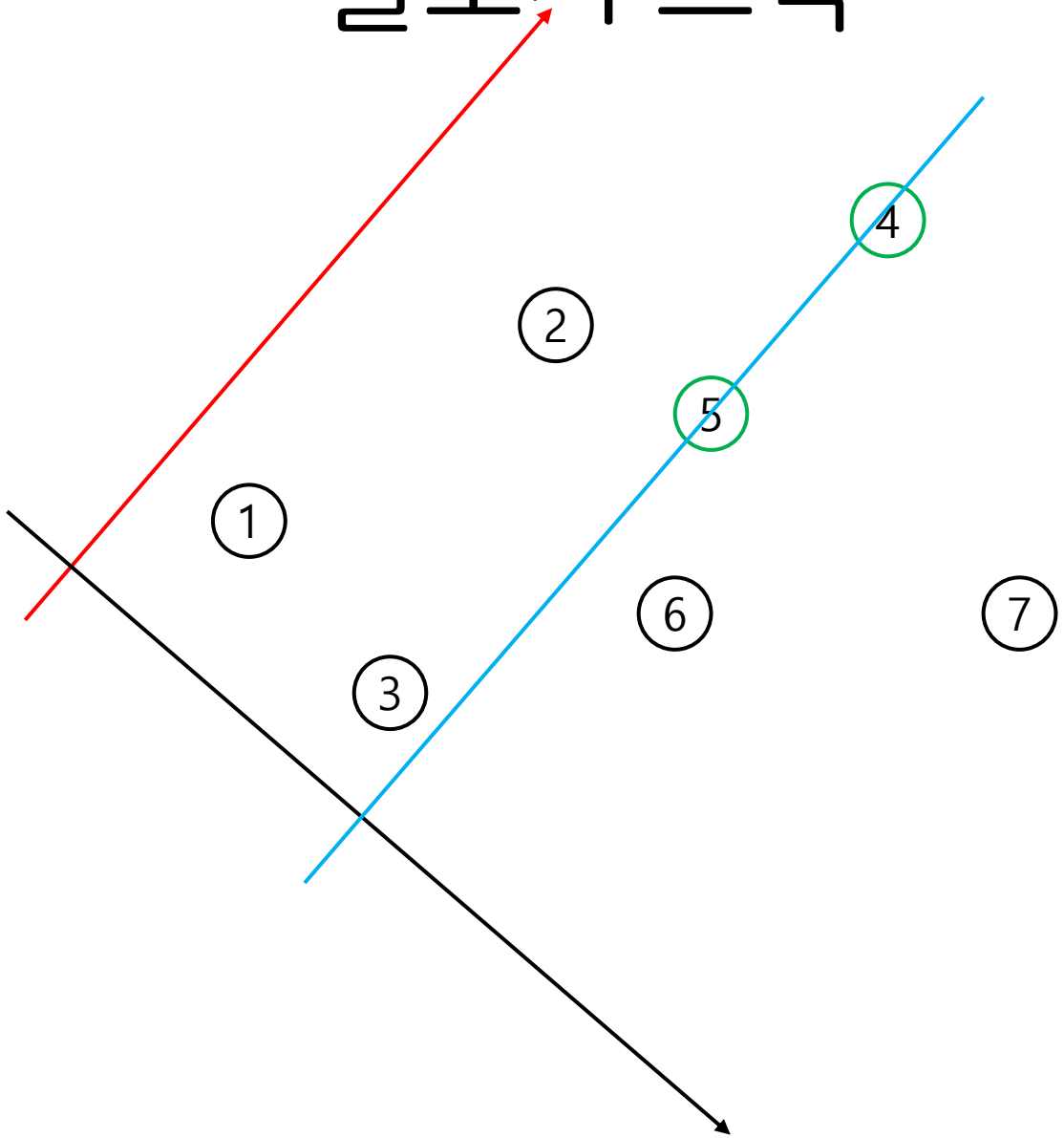
불도저 트릭



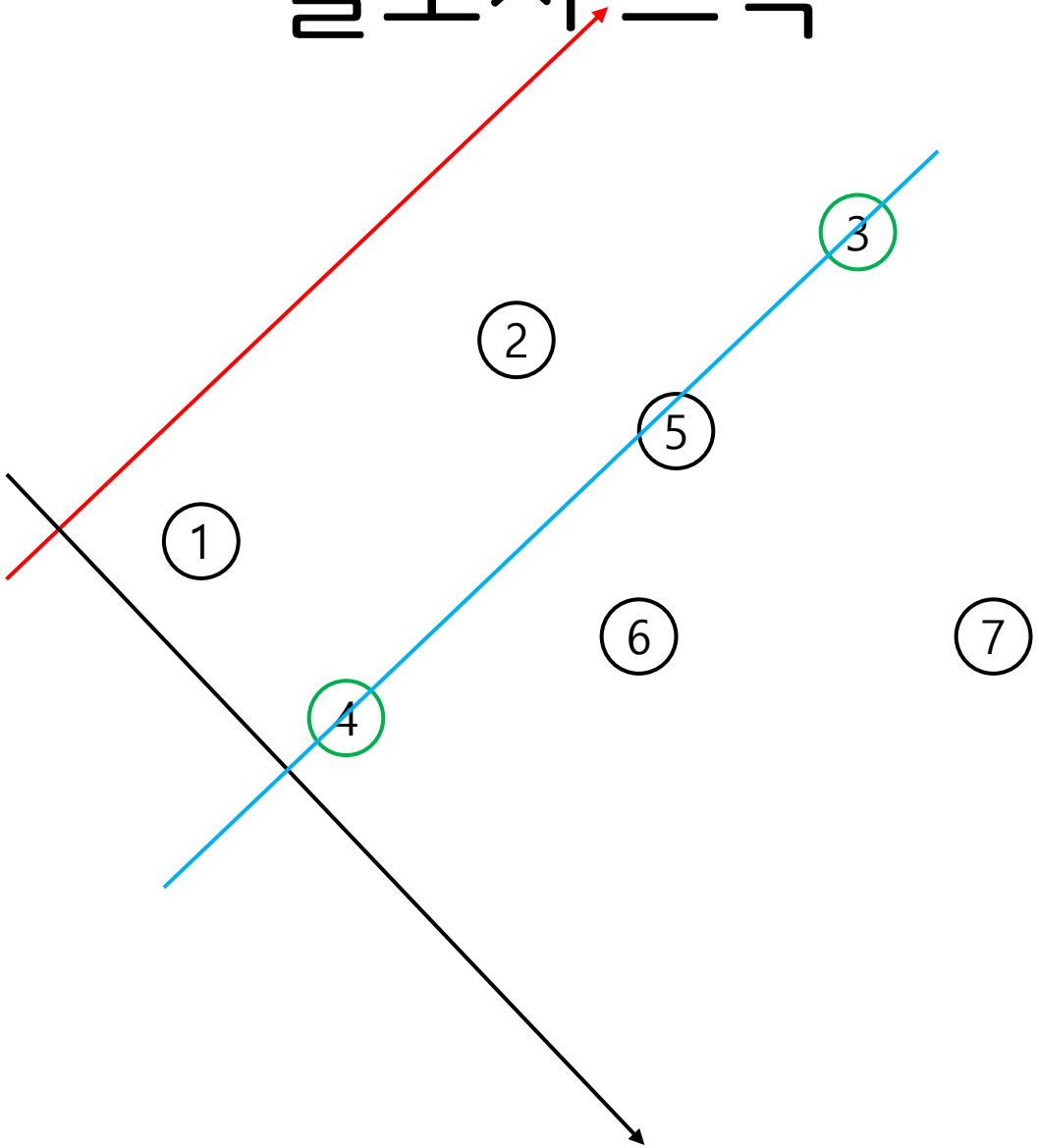
불도저 트릭



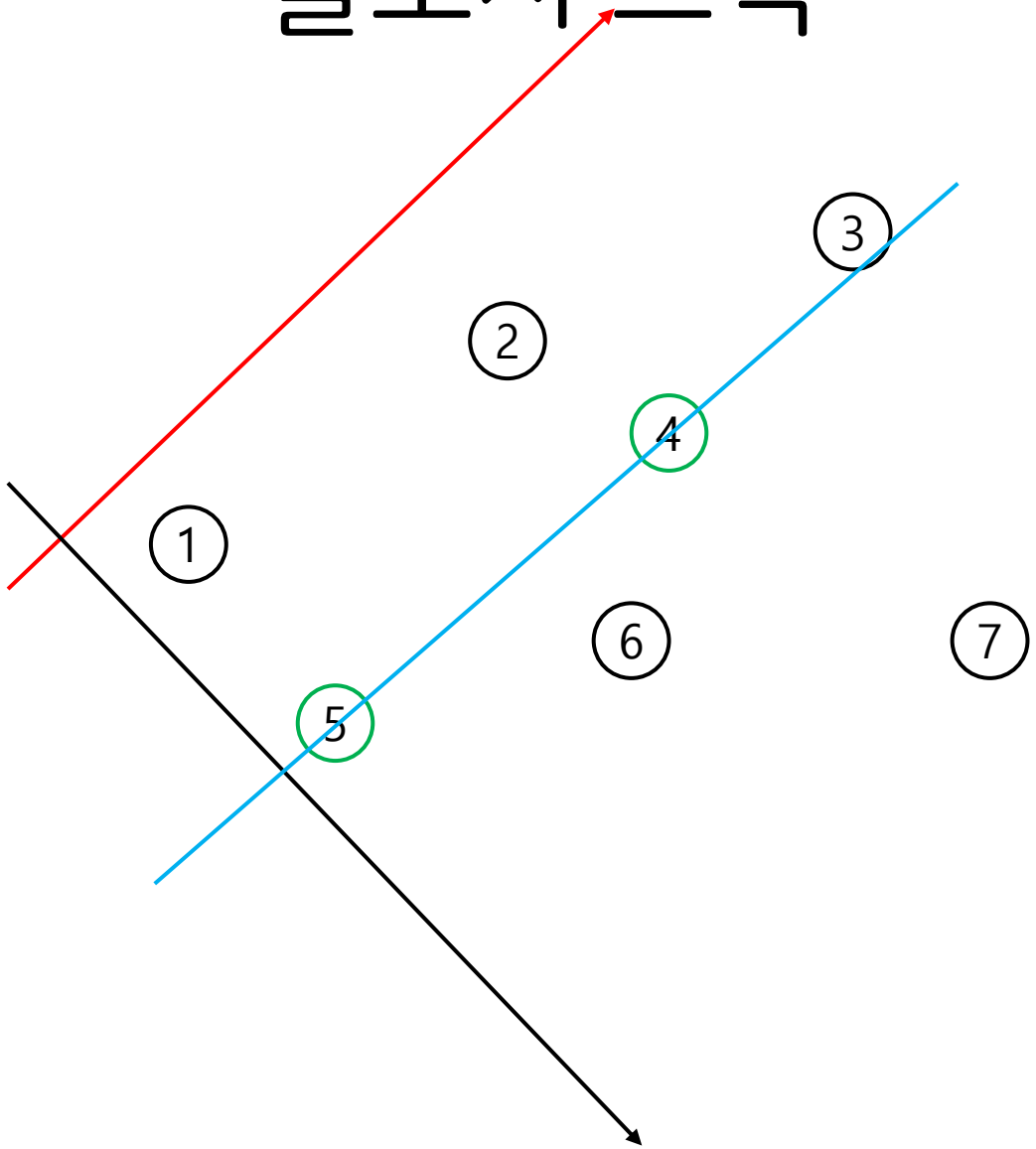
불도저 트릭



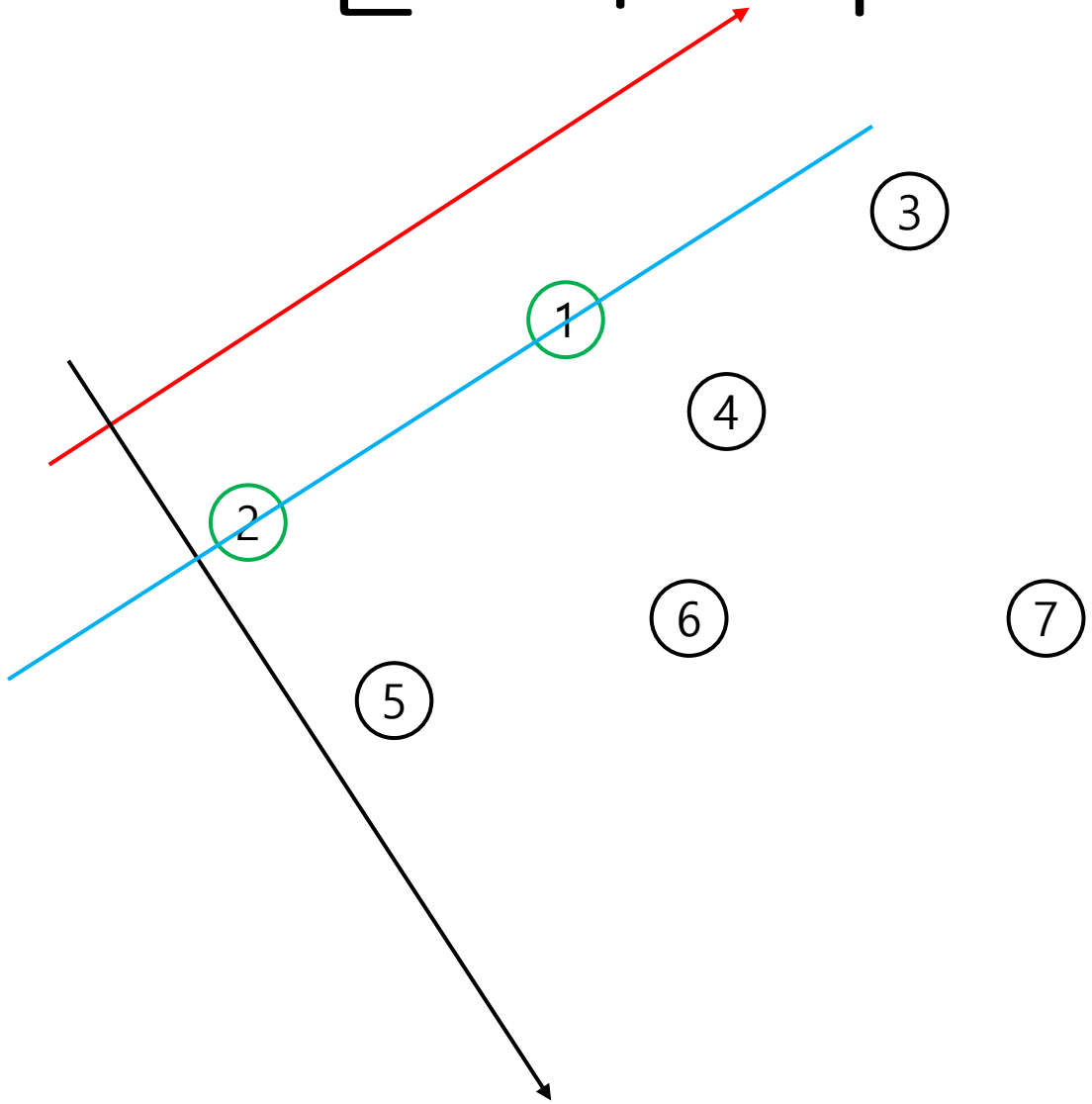
불도저 트릭



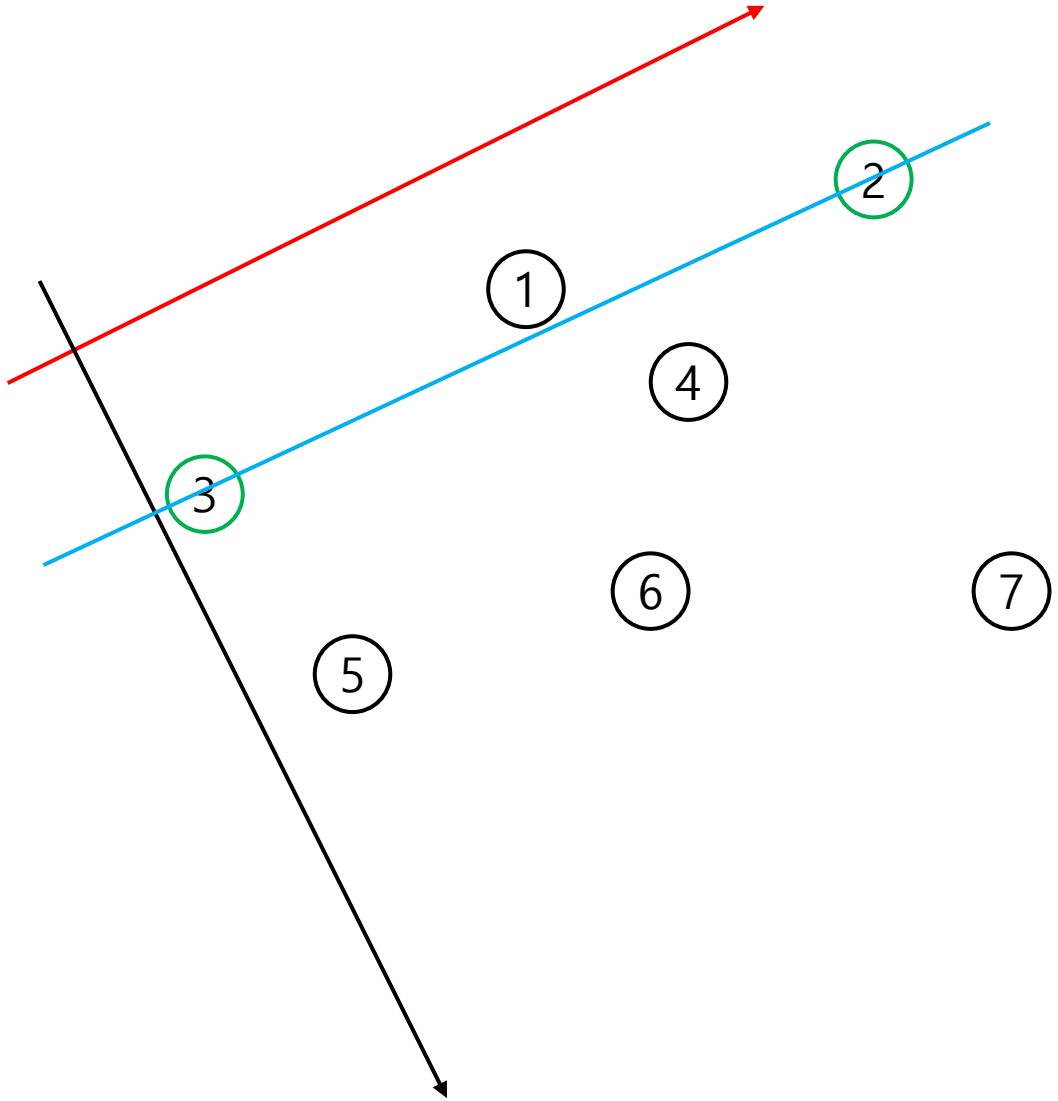
불도저 트릭



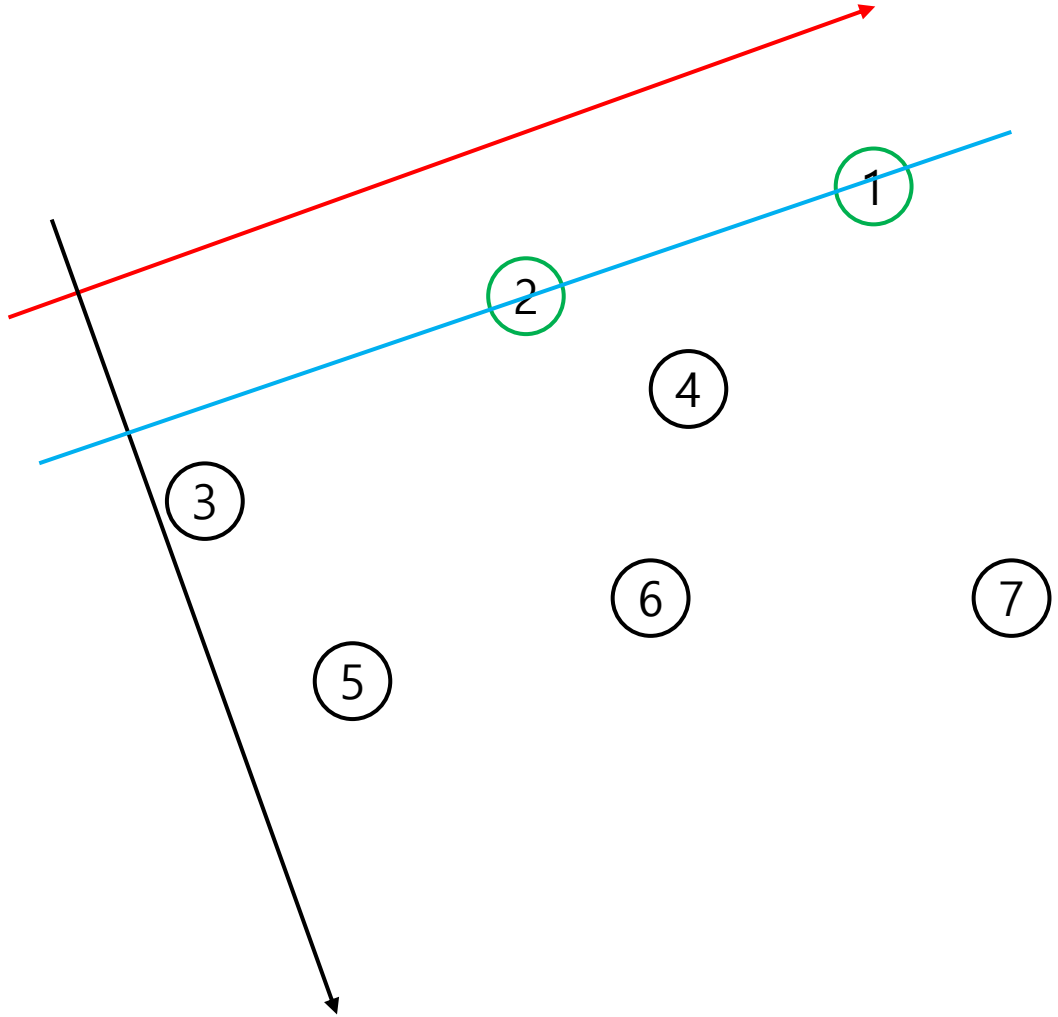
불도저 트릭



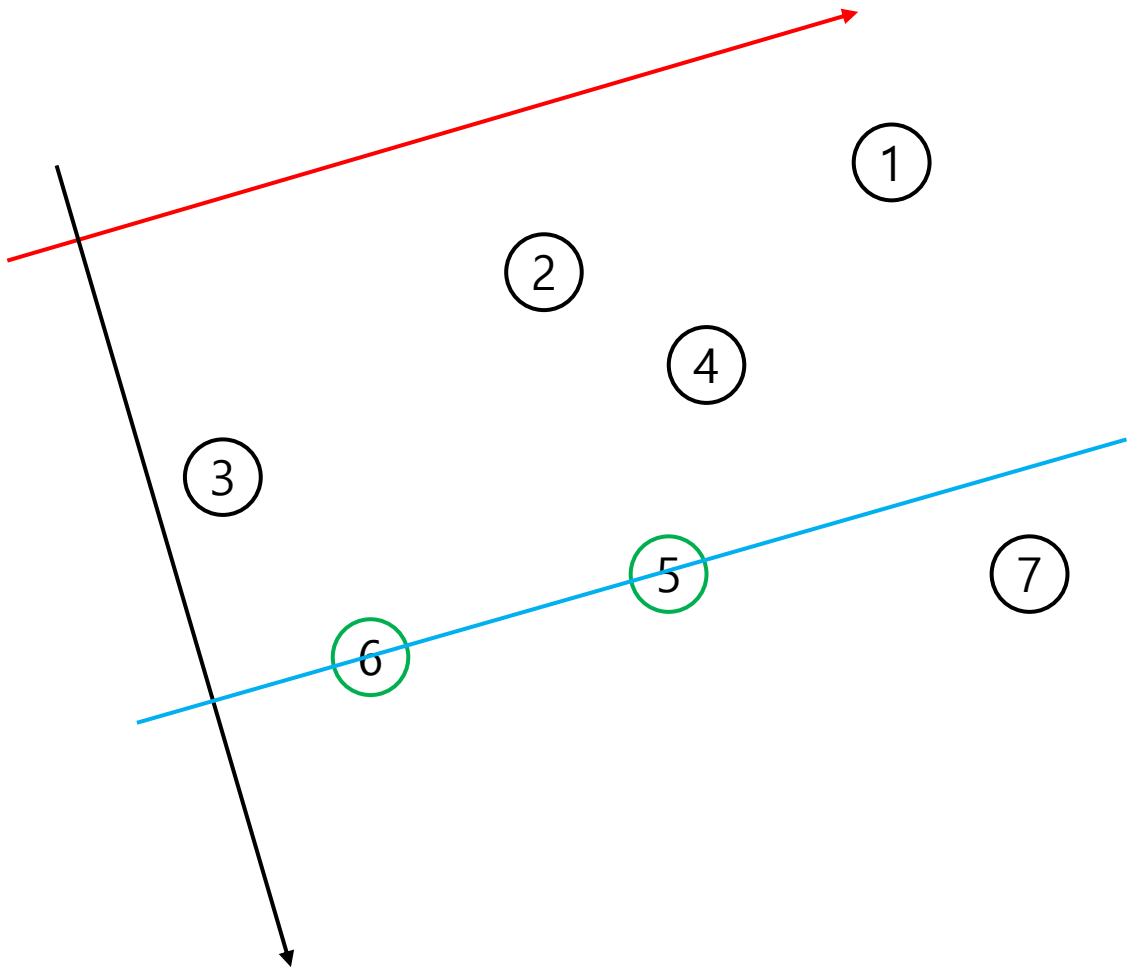
불도저 트릭



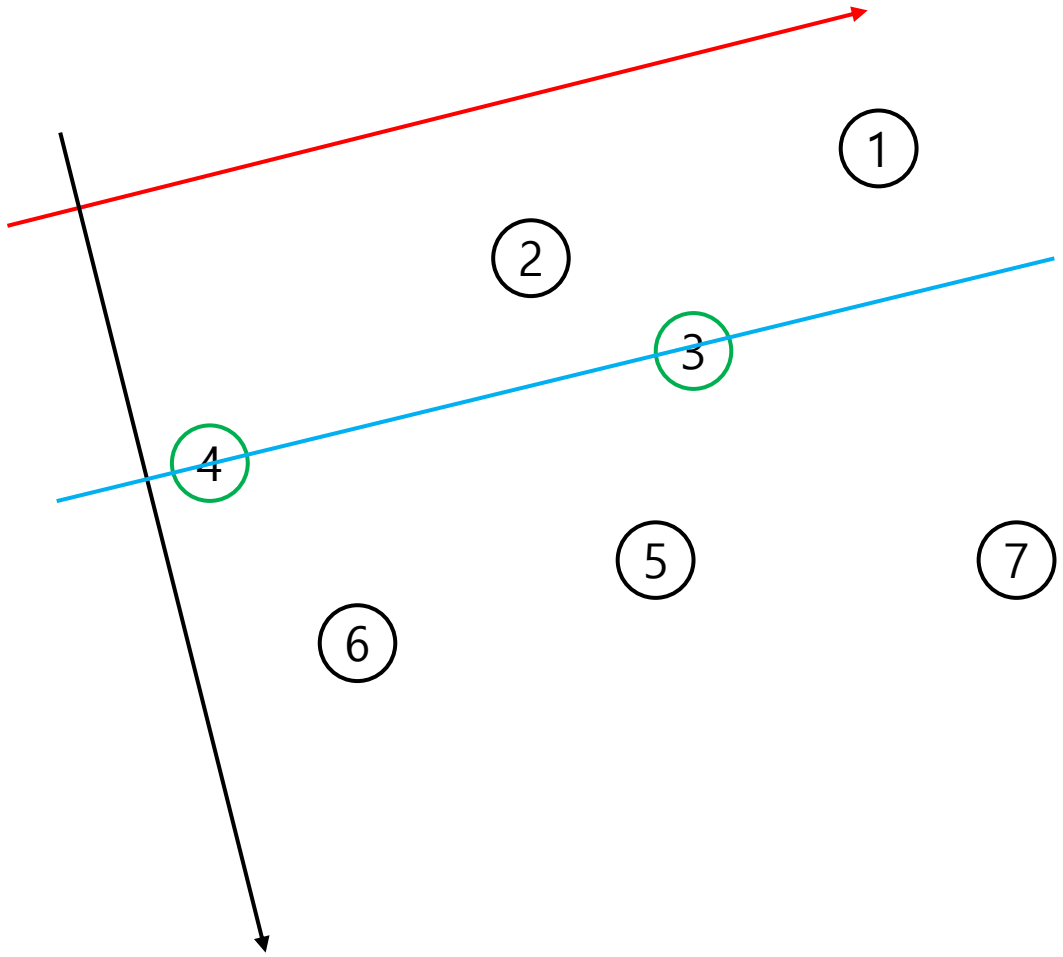
불도저 트릭



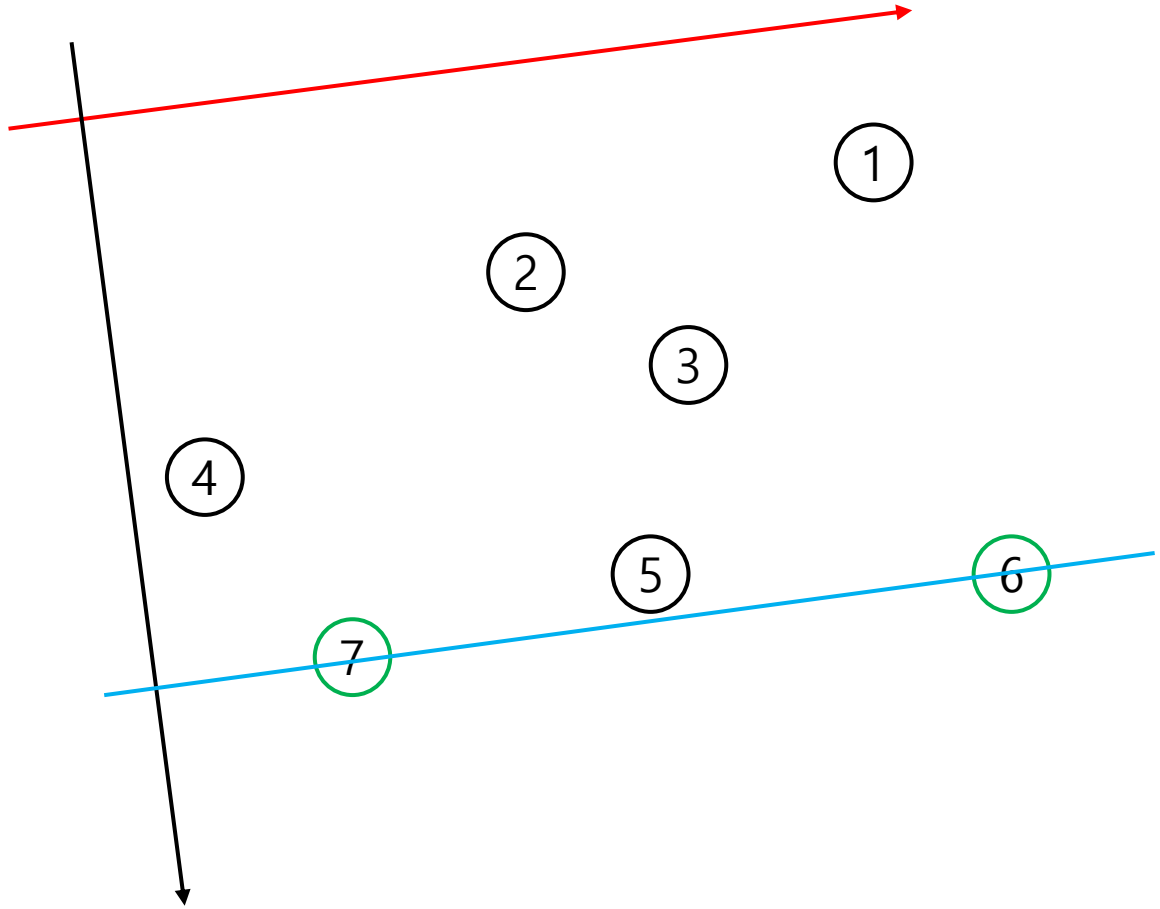
불도저 트릭



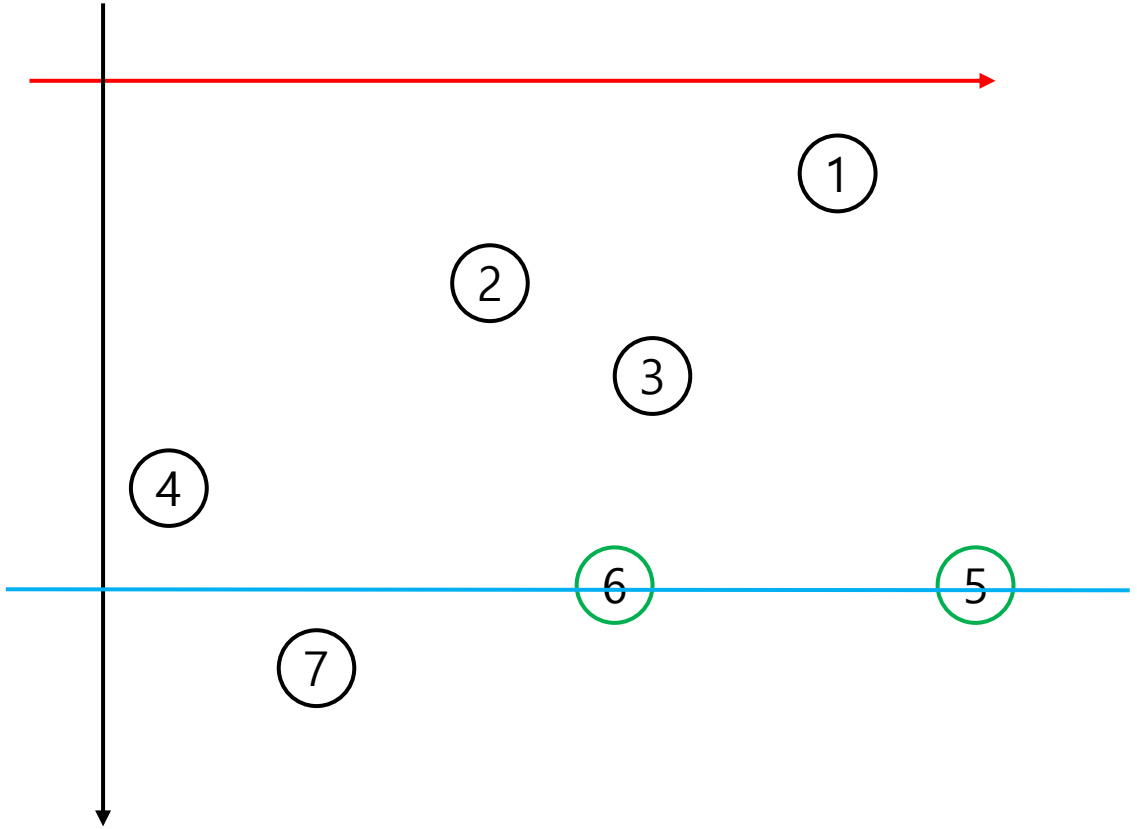
불도저 트릭



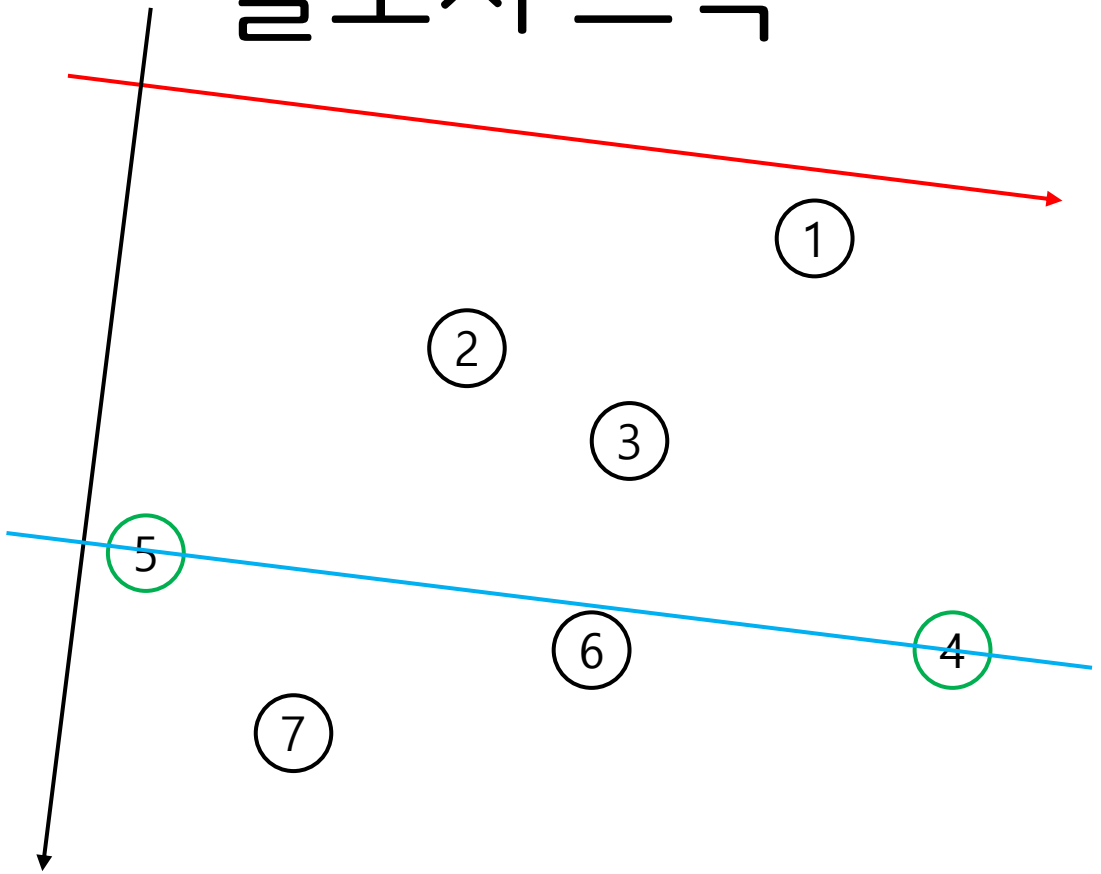
불도저 트릭



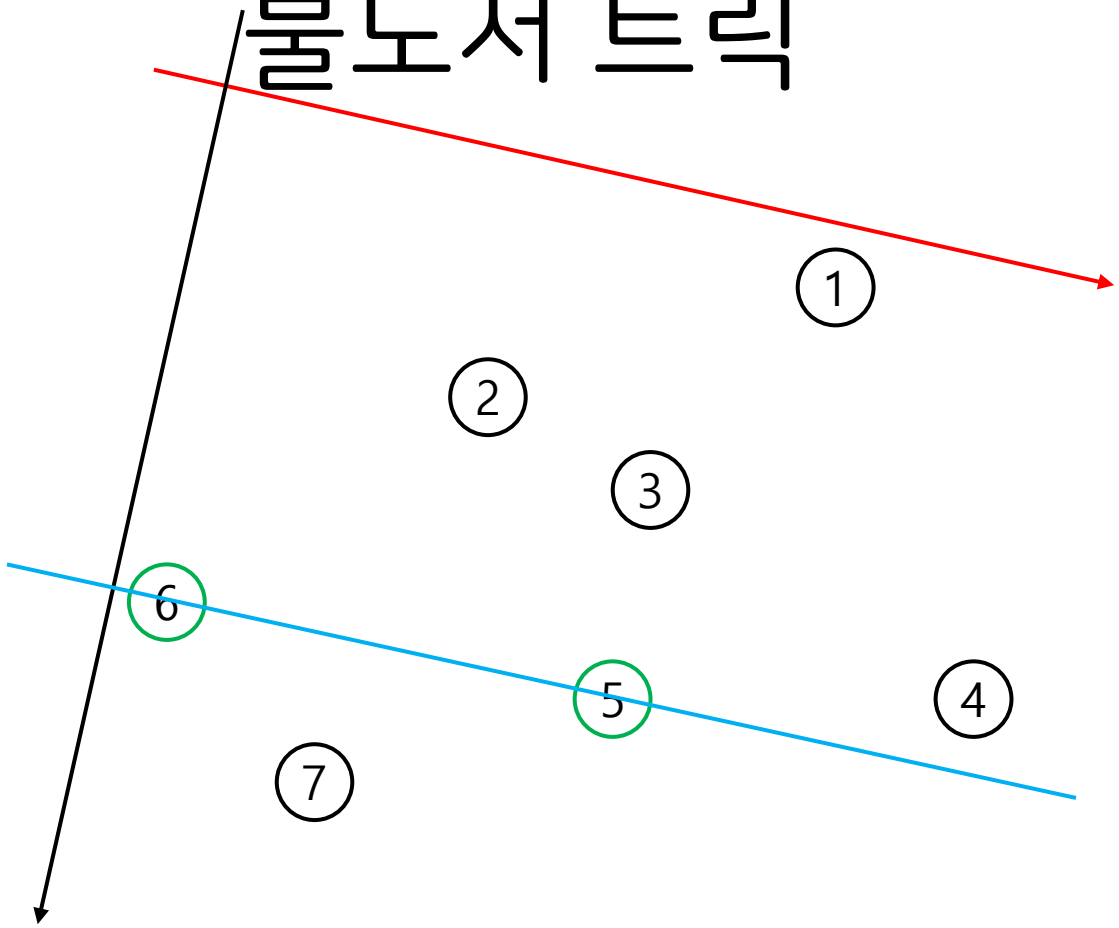
불도저 트릭



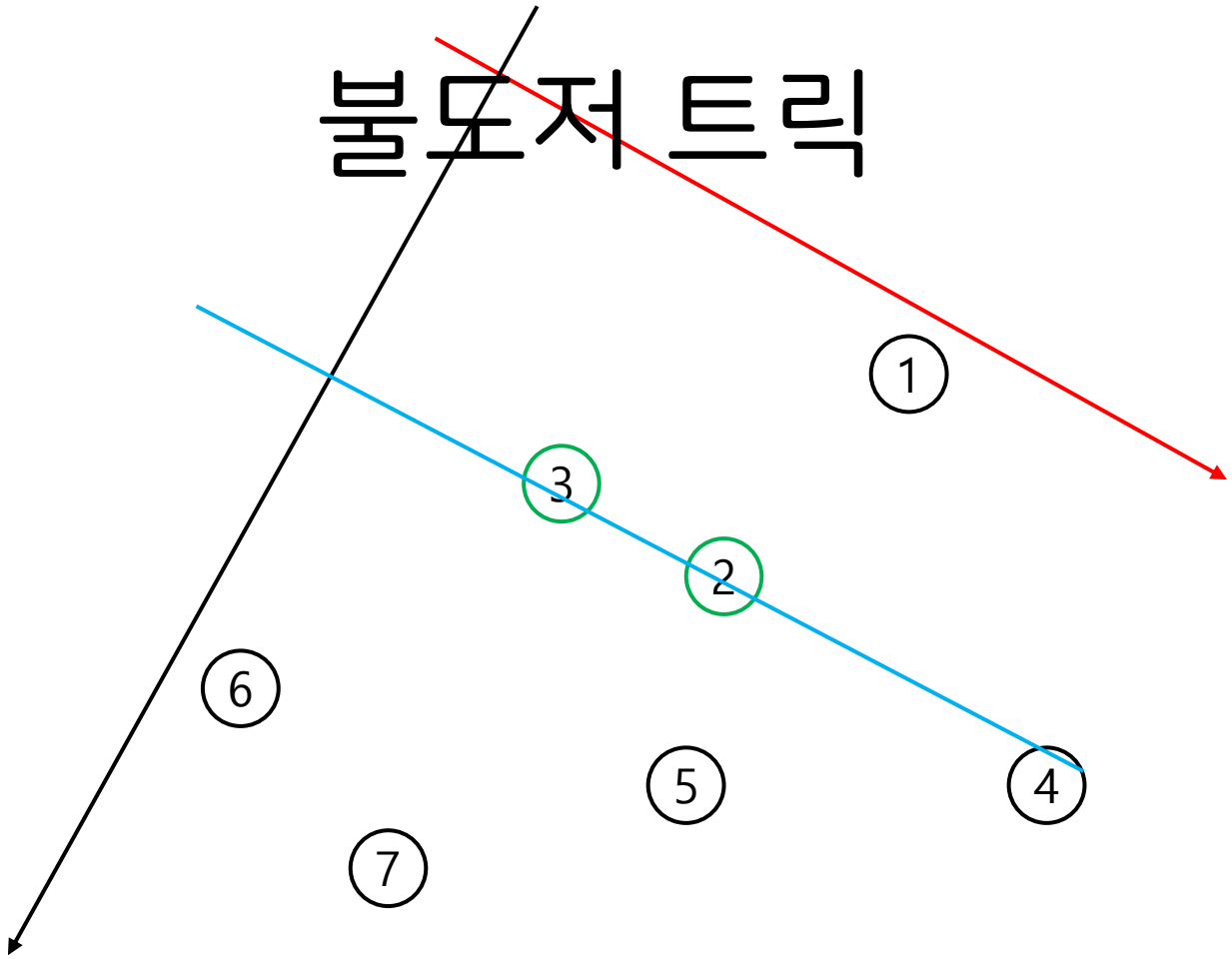
불도저 트릭



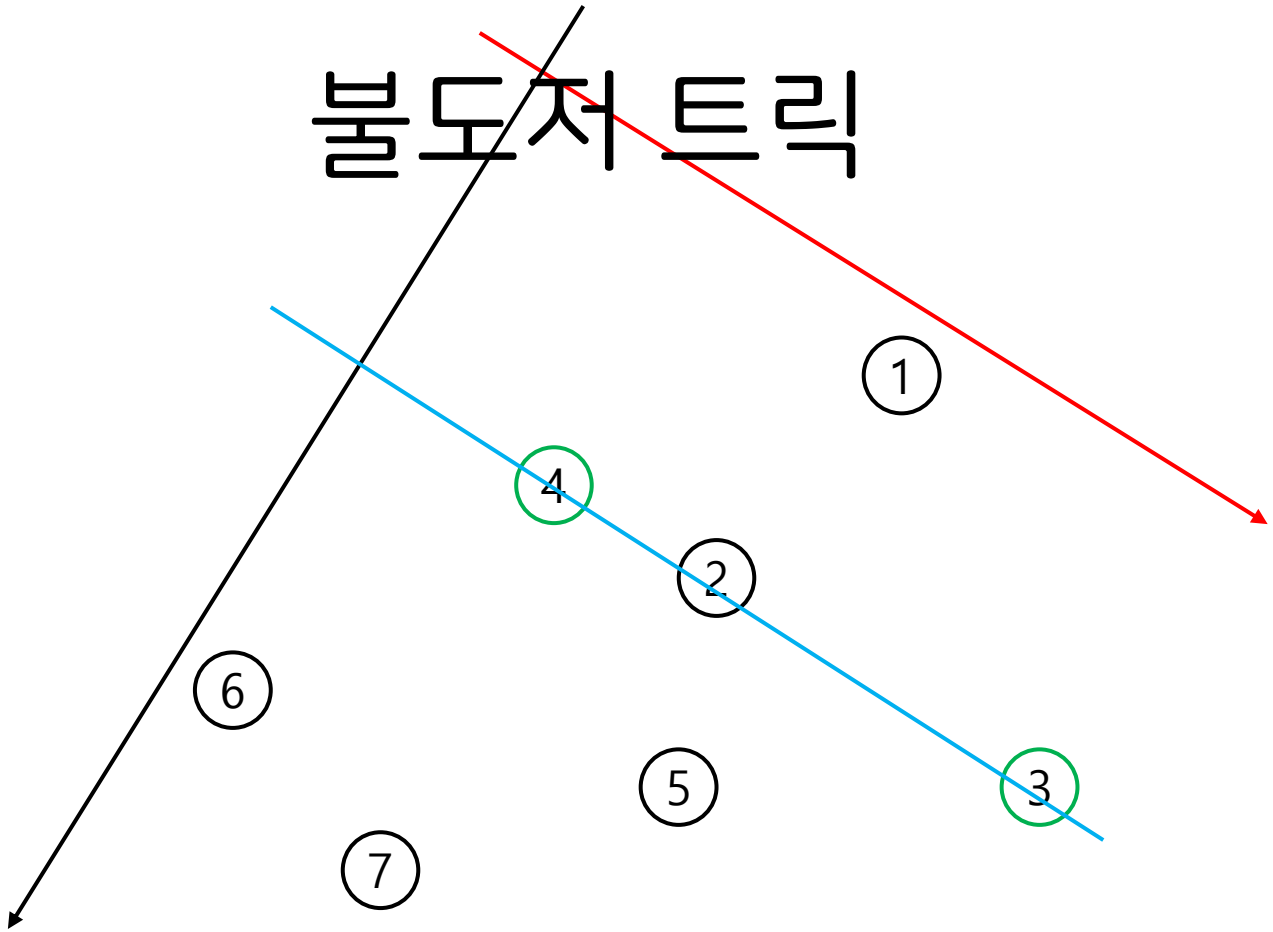
불도저 트릭



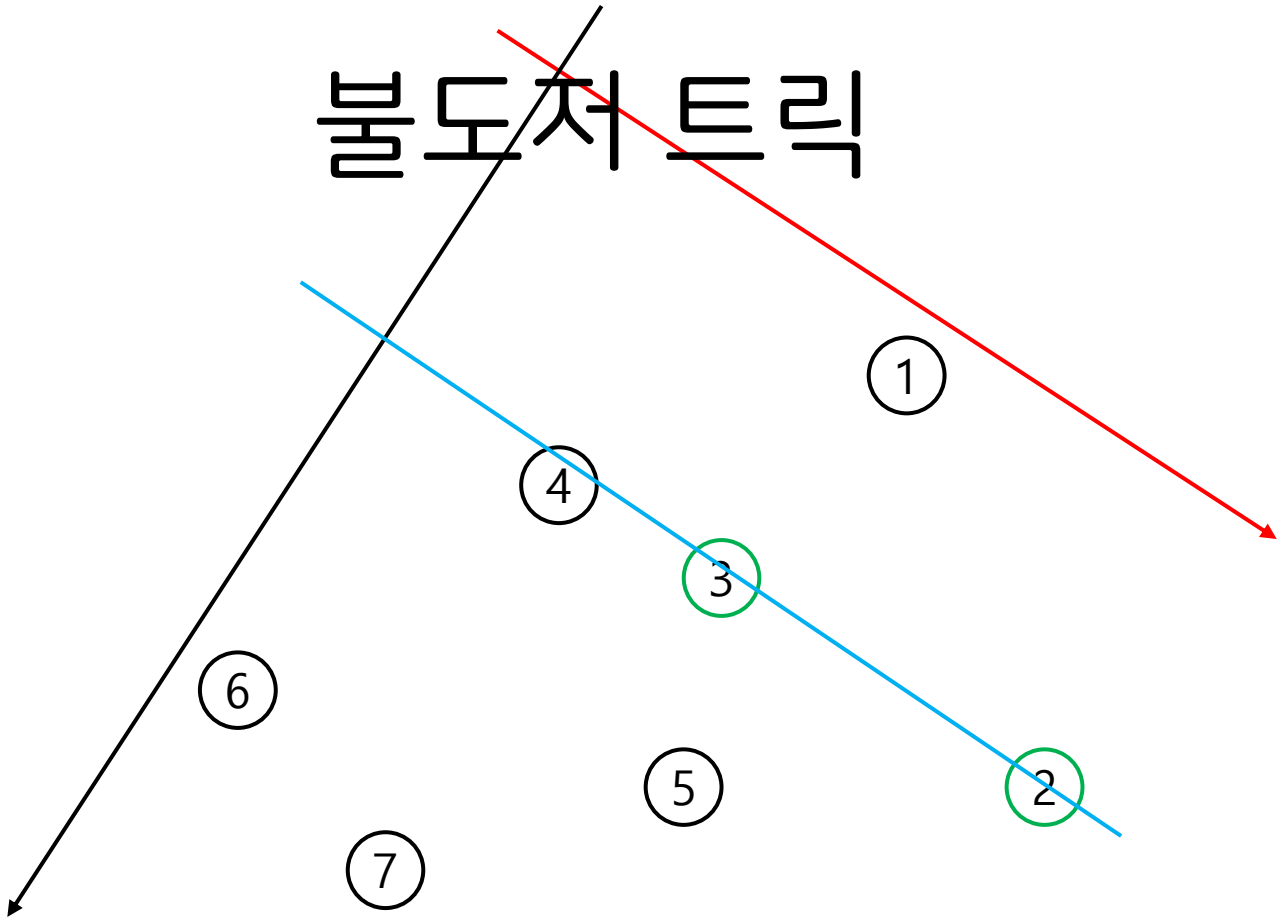
불도저 트릭



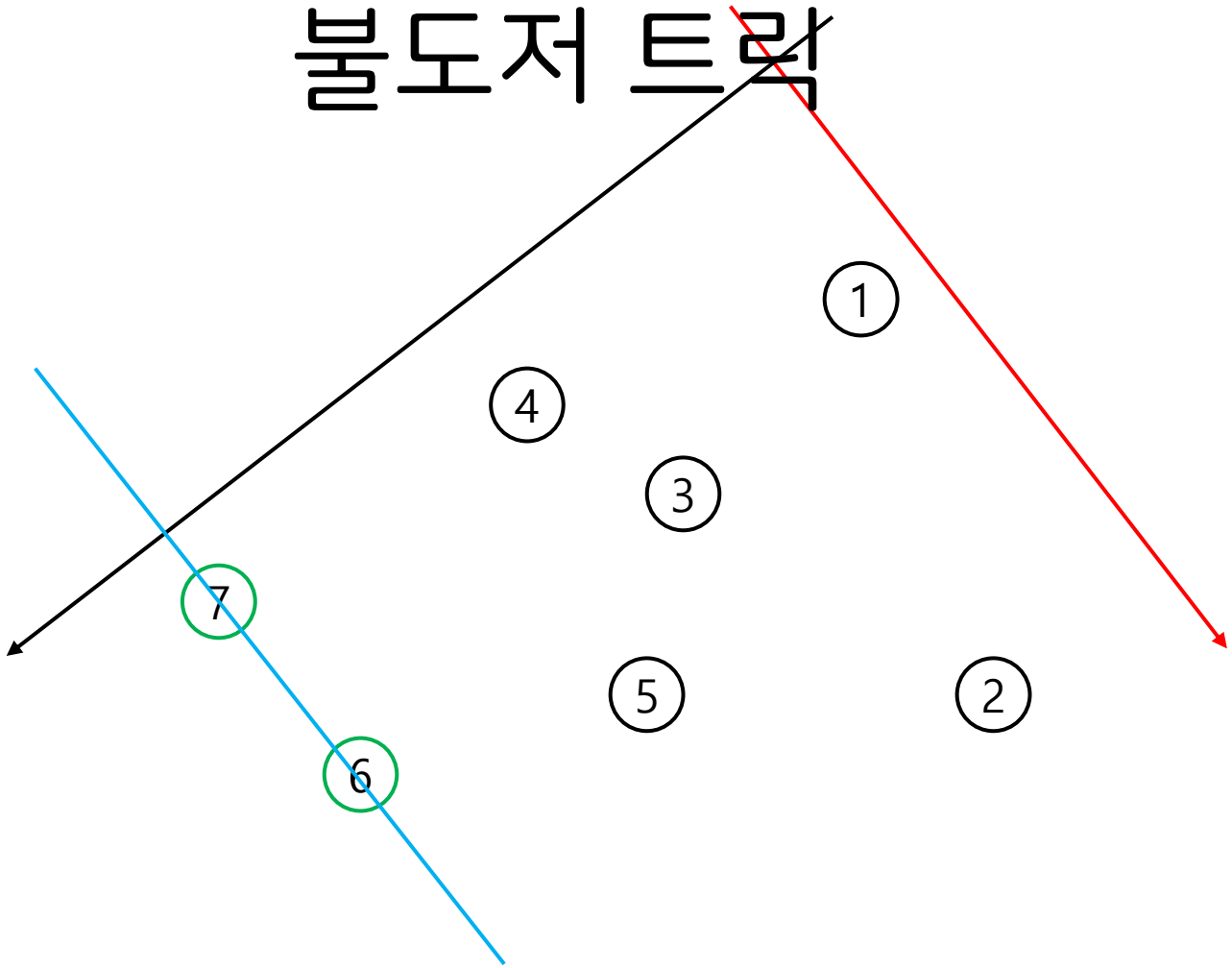
불도저 트릭



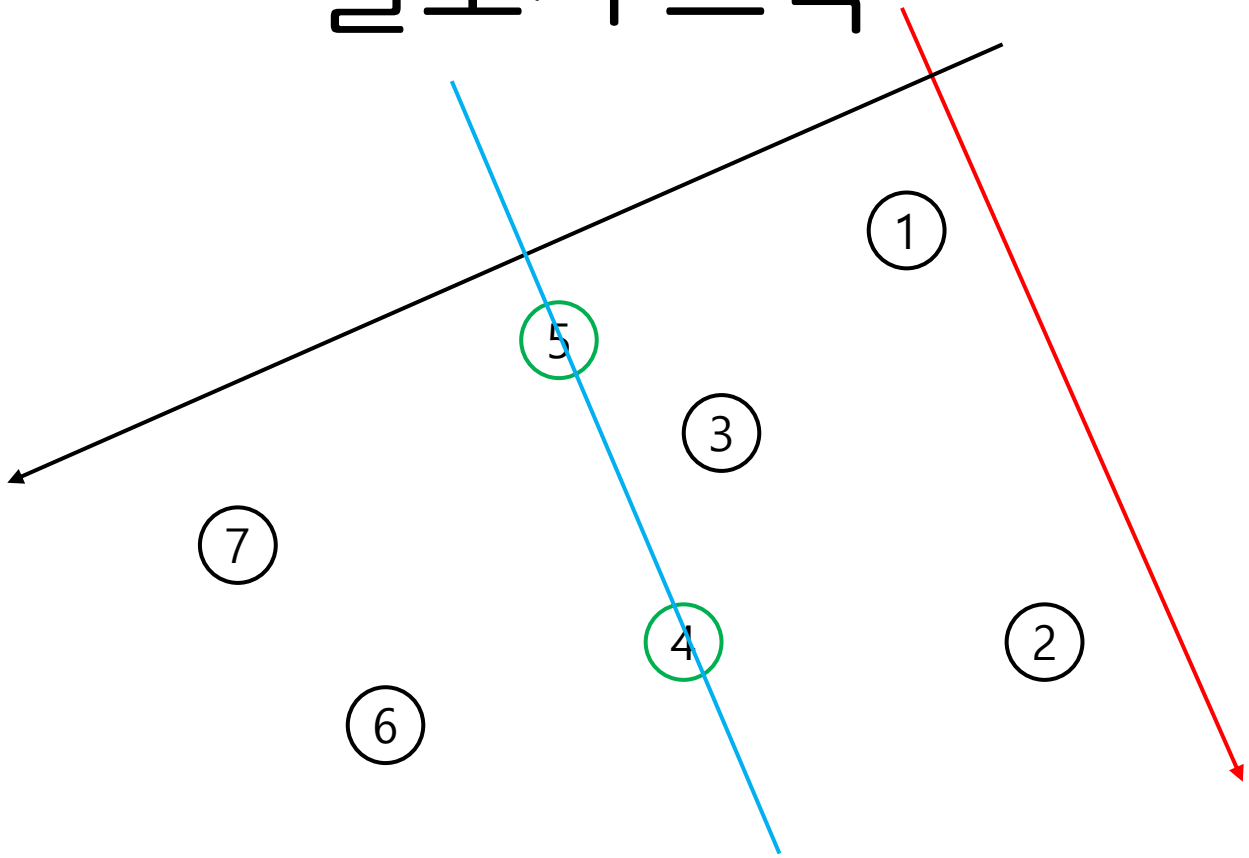
불도저 트릭



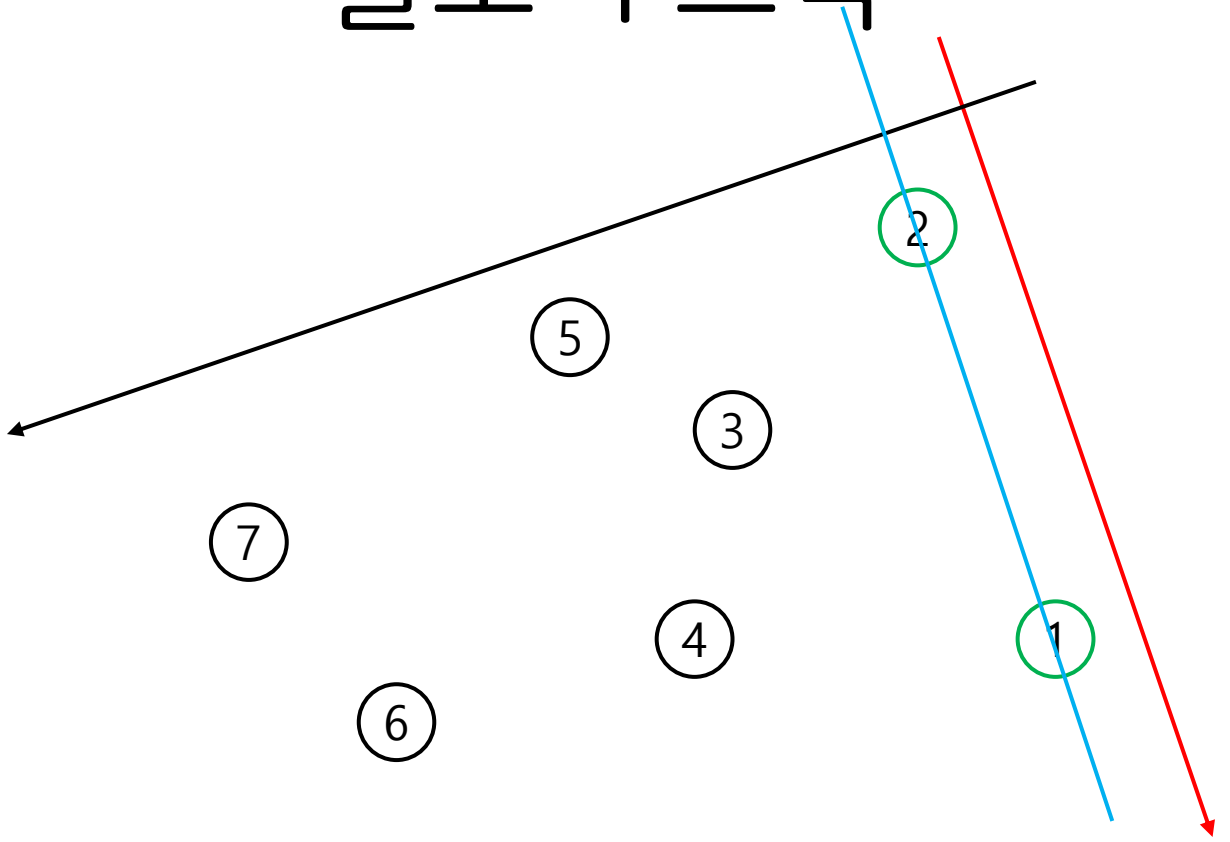
불도저 트럭



불도저 트릭

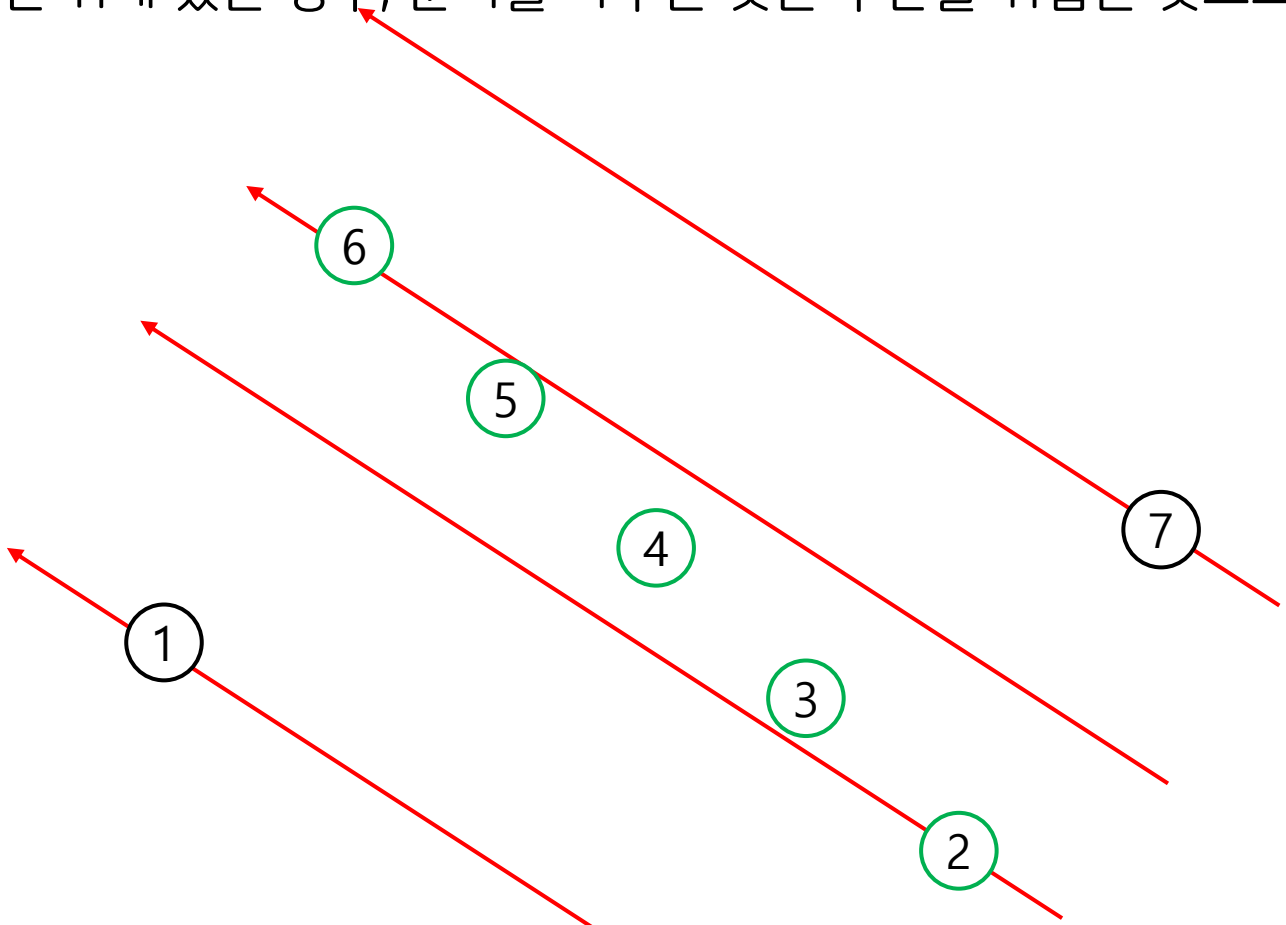


불도저 트릭



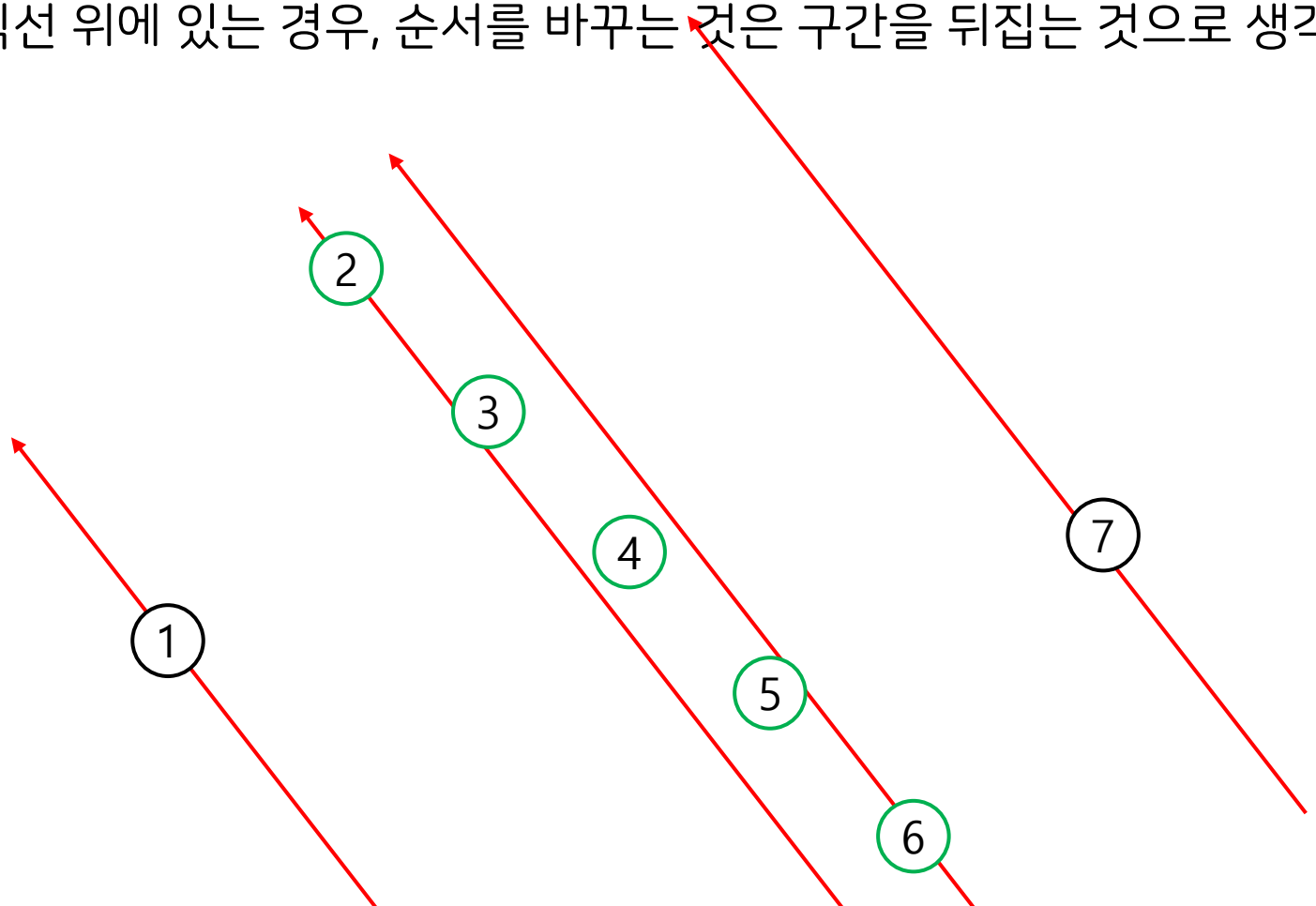
불도저 트릭

- 공선점이 있는 경우
  - 세 점이 한 직선 위에 있는 경우, 순서를 바꾸는 것은 구간을 뒤집는 것으로 생각할 수 있음



불도저 트릭

- 공선점이 있는 경우
  - 세 점이 한 직선 위에 있는 경우, 순서를 바꾸는 것은 구간을 뒤집는 것으로 생각할 수 있음



불도저 트릭

- 공선점이 있는 경우
  - 세 점이 한 직선 위에 있는 경우, 순서를 바꾸는 것은 구간을 뒤집는 것으로 생각할 수 있음
    - 인접한 점들만 서로 교환 가능하므로 교환 순서를 잘 구현해야 함
    - i, i+1, … , j 번 점을 뒤집는 경우
    - swap(i, i+1), swap(i, i+2), … , swap(i, j)를 통해 i를 맨 뒤로 보내고
    - swap(i+1, i+2), … , swap(i+1, i+3), … , swap(i+1, j)를 통해 i+1을 i 바로 앞으로 보내고
    - …

불도저 트릭

- BOJ 9484 최대삼각형, 최소삼각형
  - N개의 점이 주어지면, 이 점을 이용해 만들 수 있는 가장 큰/작은 삼각형의 넓이를 구하는 문제

  - 밑변으로 사용할 두 점을 고정
    - 높이가 최대/최소인 점을 선택하면 됨
    - 고정한 두 점을 u, v (u+1 = v)라고 하면
    - 높이가 최대인 점은 1 또는 N
    - 높이가 최소인 점은 u-1 또는 v+1

불도저 트릭

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

struct Point{
    ll x, y;
    bool operator < (const Point &p) const {
        return tie(x, y) < tie(p.x, p.y);
    }
    bool operator == (const Point &p) const {
        return tie(x, y) == tie(p.x, p.y);
    }
};

struct Line{
    ll i, j, dx, dy; // i < j, dx >= 0
    Line(int i, int j, const Point &pi, const Point &pj)
        : i(i), j(j), dx(pj.x-pi.x), dy(pj.y-pi.y) {}

    // dy / dx < l.dy / l.dx
    bool operator < (const Line &l) const {
        ll le = dy * l.dx, ri = l.dy * dx;
        return tie(le, i, j) < tie(ri, l.i, l.j);
    }
    bool operator == (const Line &l) const {
        return dy * l.dx == l.dy * dx;
    }
};

ll TriangleArea(const Point &p1, const Point &p2, const Point &p3){
    ll cross = (p2.x - p1.x) * (p3.y - p2.y) - (p3.x - p2.x) * (p2.y - p1.y);
    return abs(cross);
}
```

불도저 트릭

```cpp
int N, Pos[2020];
Point A[2020];

void Solve(){
    sort(A+1, A+N+1);
    for(int i=1; i<=N; i++) Pos[i] = i;

    vector<Line> V;
    for(int i=1; i<=N; i++) for(int j=i+1; j<=N; j++) V.emplace_back(i, j, A[i], A[j]);
    sort(V.begin(), V.end());

    ll Min = 0x3f3f3f3f3f3f3f3f, Max = 0xc0c0c0c0c0c0c0c0;
    for(int i=0, j=0; i<V.size(); i=j){
        while(j < V.size() && V[i] == V[j]) j++; // [i, j) -> same slope
        for(int k=i; k<j; k++){
            int u = V[k].i, v = V[k].j; // point id
            swap(Pos[u], Pos[v]);
            swap(A[Pos[u]], A[Pos[v]]);
            if(Pos[u] > Pos[v]) swap(u, v);
            if(Pos[u] > 1){
                Min = min(Min, TriangleArea(A[Pos[u]], A[Pos[v]], A[Pos[u]-1]));
                Max = max(Max, TriangleArea(A[Pos[u]], A[Pos[v]], A[1]));
            }
            if(Pos[v] < N){
                Min = min(Min, TriangleArea(A[Pos[u]], A[Pos[v]], A[Pos[v]+1]));
                Max = max(Max, TriangleArea(A[Pos[u]], A[Pos[v]], A[N]));
            }
        }
    }
    cout << Min/2 << "." << Min%2*5 << " ";
    cout << Max/2 << "." << Max%2*5 << "\n";
}
```

불도저 트릭

- BOJ 16783 Bulldozer
  - 2차원 평면 형태의 광산에서 불도저를 이용해 채굴함
  - i번째 지점 (x_i, y_i)에 금이 있으면 v_i원을 얻고, 돌이 있으면 c_i원을 잃음
  - 불도저는 두 개의 평행선을 그린 다음, 평행선 사이에 있는 모든 지점을 채굴함
  - 최대 이득을 구하는 문제

  - 정렬 상태가 고정되어 있다면?
    - 최대 부분 합 문제

  - 최대 부분 합에서 점 업데이트가 주어지면?
    - 금광 세그먼트 트리
    - 불도저 트릭 + 금광 세그

질문?