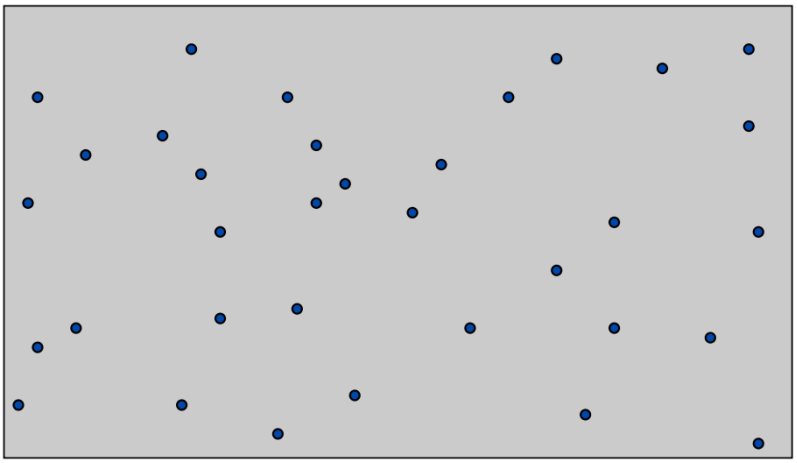
Bài tập tính điểm giữa kỳ

Nguyễn Tuấn Anh*

Ngày 27 tháng 12 năm 2021

Bài toán. Tìm khoảng cách của 2 điểm gần nhất trong tập hợp N điểm trong mặt phẳng tọa độ. Yêu cầu:

- $N \leq 10^5$.
- Các tọa độ là số nguyên có trị tuyệt đối không quá 10^9.
- Kết quả in ra với độ chính xác 10^{-9}.



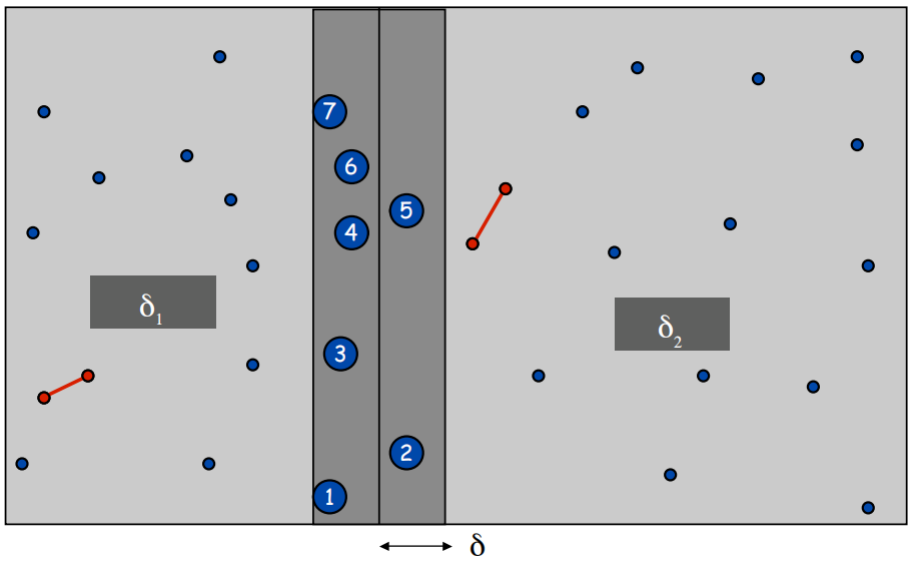
Hình 1: Hình minh họa

*Sinh viên lớp KHTN2021, Trường ĐH Công nghệ Thông tin - ĐHQG TP.HCM

Lời giải: Ta quy ước khoảng cách giữa một điểm với chính nó bằng 0.000000000. Các điểm đã cho được kí hiệu bởi $p_i, \forall i \in [1; N]$.

Nhận xét 1. Trong $N \geq 2$ điểm luôn tồn tại 2 điểm có khoảng cách nhỏ nhất.

Sắp xếp N điểm tăng dần theo tọa độ x (tức là hoành độ).
Đặt $a = \left\lfloor \frac{N}{2} \right\rfloor$.
Đường thẳng $x = x_a$ phân tách N điểm thành 2 tập hợp điểm.
Đó là $A = \{p_1; p_2; ...; p_a\}$ và $B = \{p_{a+1}; p_{a+2}; ...; p_N\}$.
Tìm trong tập A, B lần lượt khoảng cách δ_1, δ_2 nhỏ nhất với mỗi 2 điểm bất kỳ.
Đặt $\delta = min\{\delta_1, \delta_2\}$.
Từ đường thẳng đứng $x = x_a$, tạo vết trái và phải có độ rộng δ.
Đến đây, ta chỉ còn làm việc với các điểm nằm trong 2 vết đã thiết lập. Các điểm ở ngoài không ảnh hưởng đến kết quả.
Sắp xếp các điểm thỏa mãn tăng dần theo tọa độ y (tức là tung độ). Gọi δ_3 là khoảng cách nhỏ nhất giữa mọi cặp điểm trong 2 vết thì đáp án sẽ là $min\{\delta, \delta_3\}$.



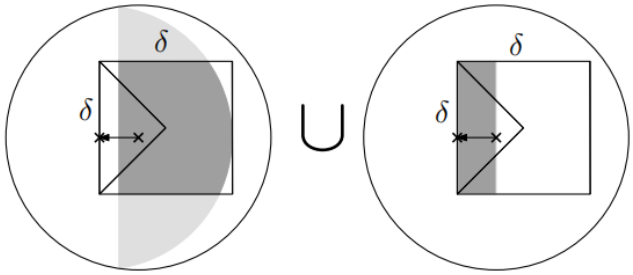
Hình 2: Vùng ngăn cách

Với 2 điểm $P(x_1; y_1), Q(x_2; y_2)$ thì công thức khoảng cách

$$|P - Q| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

nói rằng, với mỗi điểm chúng ta chỉ cần xét những điểm mà $|y_i - y_j| \leq \delta$.

Bất đẳng thức trên cùng với điều kiện để một điểm nằm trong vùng ngăn cách, hợp lại thì mỗi điểm sẽ có một hình vuông bao phủ với tâm là điểm đang xét.



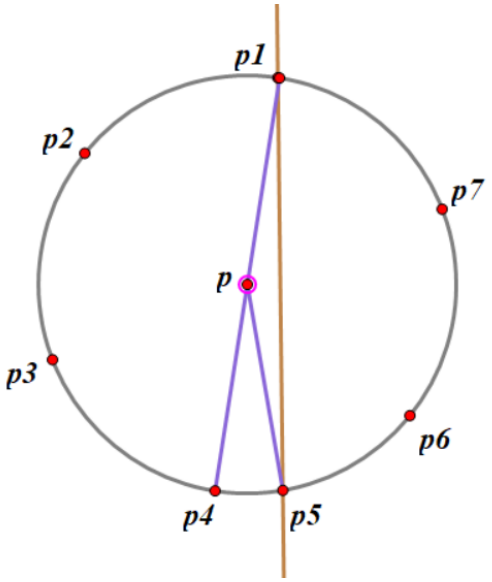
Hình 3: Hình minh họa

Nhận xét 2. Khi δ đủ nhỏ, bài toán sẽ dễ dàng hơn.

Chứng minh. Các điểm đều nguyên nên khi δ càng nhỏ thì số điểm nguyên trong hình vuông đang xét càng ít (có thể quan sát dễ dàng bằng định lý Pick[1]). □

Nhận xét 3. Với một điểm p nào đó, chỉ cần xét tối đa 7 điểm xung quanh.

Chứng minh. Hình vẽ sau đây minh họa điều đó ($p_1p_2p_3p_4p_6p_7$ là lục giác đều). □



Hình 4: 7 điểm cần xét

[1] https://nttuan.org/2020/06/14/pick/

Giả sử sắp xếp các điểm cần $O(n\log(n))$ thì theo lập luận và các nhận xét ở trên, độ phức tạp tổng thể sẽ là $O(n\log^2(n))$. Phần triển khai thuật toán do code quá 1 trang nên em xin chia nhỏ ra ạ.

Phần chuẩn bị sẵn

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

#define pb push_back
#define F first
#define S second
typedef long long ll;
typedef pair<ll,ll> pll;

ll N;
//delta=+oo
ll delta=LLONG_MAX;
pll points[200069];
```

Sắp xếp theo tung độ và tính bình phương khoảng cách

```
//sort y
bool cmpy(const pll &a, const pll &b){
    return a.S<b.S;
}

//square distance
ll dis(pll a, pll b){
    return (b.F-a.F)*(b.F-a.F)+(b.S-a.S)*(b.S-a.S);
}
```

Thực hiện thuật toán

```
ll bruteforce(int l, int r){
    for(int i=l; i<=r; i++)
        for(int j=i+1; j<=r; j++)
            delta=min(delta,dis(points[i],points[j]));
    return delta;
}

ll NTAsolve(ll l, ll r){
    if(r-l+1<=3)
        return bruteforce(l, r);
    ll mid=(ll)(l+r)/2;
    delta=min({delta, NTAsolve(l, mid),NTAsolve(mid+1, r)});
    vector<pll> str;
    for(ll i=l; i<=r; i++)
        if(abs(points[i].F-points[mid].F)*
        abs(points[i].F-points[mid].F)<delta)
            str.pb(points[i]);
    sort(str.begin(),str.end(),cmpy);
    ll siz=str.size();
    for(ll i=0; i<siz-1; i++){
        ll j=i+1, dy=str[j].S-str[i].S;
        while(j<siz && dy*dy<delta){
            delta=min(delta,dis(str[i],str[j]));
            j++;
        }
    }
    return delta;
}
```

Chương trình chính

```
int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cin>>N;
    for(int i=1; i<=N; i++)
        cin>>points[i].F>>points[i].S;
    sort(points+1,points+N+1);//sort x
    cout<<fixed<<setprecision(9)<<sqrt(NTAsolve(1,N));
    return 0;
}
```

Code đã nộp tại CSES [2].

Độ phức tạp: $O(n \log^2(n))$.

[2] https://cses.fi/paste/789e91a22d07b9c532347f/

Đến đây, có thể cải tiến bằng 2 hướng.

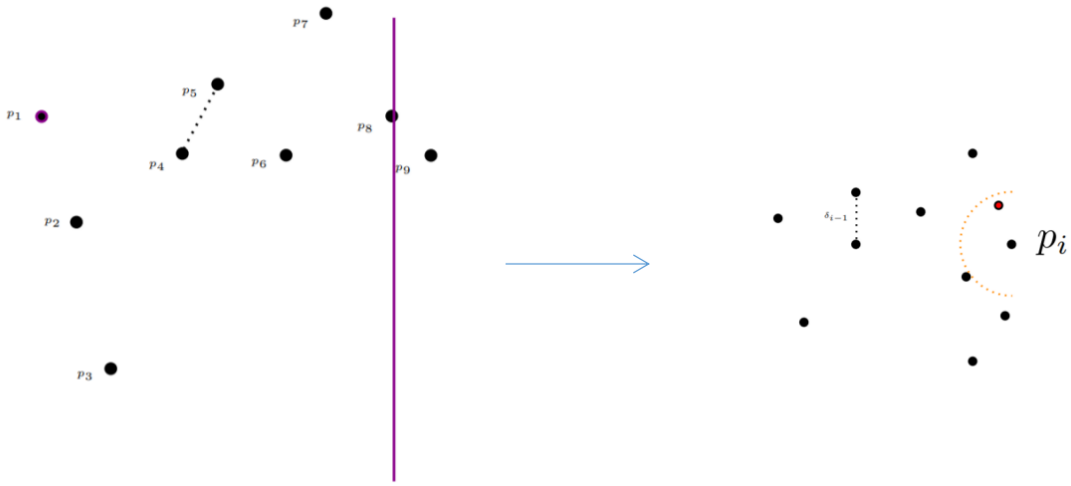
Hướng 1: Ngay từ đầu, chúng ta sắp xếp tất cả điểm tăng dần theo tung độ.

Độ phức tạp giảm còn $O(n \log(n))$.

Hướng 2: Cũng ý tưởng chia để trị[3], nhưng thực hiện tất cả nhận xét cùng một lúc.

Duyệt lần lượt các điểm từ trái sang phải. Trong quá trình duyệt, liên tục duy trì và cập nhật δ tại mỗi thời điểm.

Dựa vào hình 4, ta có thể nhận thấy nếu xét toàn bộ điểm trong bán kính hình tròn thì sẽ mất chi phí do lặp lại. Vì vậy, để giảm thiểu, chỉ cần xét nửa trái hình tròn. Hình dưới đây minh họa điều đó.



Hình 5: Nửa hình tròn

Điểm thứ i trả lời khoảng cách nhỏ nhất của i điểm đầu tiên. δ tại mỗi thời điểm sẽ được cập nhật mới theo công thức

$$\delta_{p_i} = min\{\delta_{p_{i-1}}, K\}$$

Với K là khoảng cách ngắn nhất của p_i và một số điểm trong nửa đường tròn.

[3]Mỗi điểm có vai trò riêng, hợp lại sẽ được đáp án

```cpp
#include<bits/stdc++.h>
using namespace std;
#define int long long
#define F first
#define S second
#define pb push_back
typedef pair<int,int> pii;
//square distance
int dis(pii a, pii b){
    return (b.F-a.F)*(b.F-a.F)+(b.S-a.S)*(b.S-a.S);
}
int NTAsolve(vector<pii> points, int n){
    int ans=LLONG_MAX;
    sort(points.begin(), points.end());
    set<pii> s={{points[0].S, points[0].F}};
    int j=0;
    for(int i=1; i<n; i++){
        int dd=ceil(sqrt(ans));
        while(j<i && points[j].F<points[i].F-dd){
            s.erase({points[j].S, points[j].F});
            j++;
        }
        //left delta
        auto l=s.lower_bound({points[i].S-dd, 0});
        //right delta
        auto r=s.upper_bound({points[i].S+dd, 0});
        for(auto it=l; it!=r; it++){
            ans=min(ans, dis({it->S, it->F}, points[i]));
        }
        s.insert({points[i].S, points[i].F});
    }
    return ans;
}
signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    int N;
    cin>>N;
    vector<pii> points;
    for(int i=1; i<=N; ++i){
        int a,b;
        cin>>a>>b;
        points.pb({a,b});
    }
    cout<<fixed<<setprecision(9)<<sqrt(NTAsolve(points,N));
    return 0;
}
```

Hình 6: Chương trình minh họa

Code đã nộp tại CSES [4].

Độ phức tạp: $O(n \log(n))$.

[4] https://cses.fi/paste/b6d32996715e7fd43224e6/

Tài liệu

[1] CSES Problem Set, *Minimum Euclidean Distance*, `https://cses.fi`

[2] `https://www.geeksforgeeks.org`

[3] Jon Kleinberg - Eva Tardos, *Algorithm Design*, Chapter 5.

[4] José C. Pereira - Fernando G. Lobo, *An Optimized Divide-and-Conquer Algorithm for the Closest-Pair Problem in the Planar Case*, Journal of Computer Science and Technology, 2010.

[5] M. J. Golin, *Closest Pairs*, HKUST, 2014.

[6] Nguyễn Trung Tuân - T's Lab, *A proof of Pick's theorem.*