OJ 10585

Center of Symmetry

Prof. Edson Alves

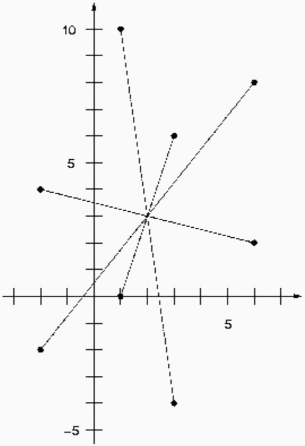
Faculdade UnB Gama

OJ 10585 – Center of symmetry

Problema

É dado um conjunto de n pontos com coordenadas inteiras. Sua tarefa é decidir se o conjunto tem ou não um centro de simetria.

Um conjunto de pontos S tem um centro de simetria se existe um ponto s (não necessariamente em S) tal que para todo ponto p em S existe um ponto q em S tal que $p - s = s - q$.



Entrada e saída

Entrada

A primeira linha da entrada contém um número c que indica o número de casos de teste que se seguem. A primeira linha de um caso de teste contém um número $1 \leq n \leq 10000$. As n linhas subsequentes contém dois números inteiros cada, os quais são as coordenadas x e y de um ponto. Cada ponto é único e temos que $-10000000 \leq x, y \leq 10000000$.

Saída

Para cada caso de teste imprima 'yes' se o conjunto de pontos tem um centro de simetria, e 'no', caso contrário.

Exemplo de entradas e saídas

Entrada

```
1
8
1 10
3 6
6 8
6 2
3 -4
1 0
-2 -2
-2 4
```

Saída

```
yes
```

Solução por força bruta

- Manipulando a expressão $p - s = s - q$ obtemos
$$s = \frac{p+q}{2},$$
ou seja, o centro de simetria é o ponto médio entre p e q

Solução por força bruta

- Manipulando a expressão $p - s = s - q$ obtemos

$$s = \frac{p+q}{2},$$

ou seja, o centro de simetria é o ponto médio entre p e q

- O uso de ponto flutuante pode ser evitado se usarmos a expressão

$$2s = p + q$$

Solução por força bruta

- Manipulando a expressão $p - s = s - q$ obtemos
$$s = \frac{p+q}{2},$$
ou seja, o centro de simetria é o ponto médio entre p e q
- O uso de ponto flutuante pode ser evitado se usarmos a expressão
$$2s = p + q$$
- A solução de força bruta consiste em fixar um ponto A em S e, para todos os pontos B em S, computar $2s$

Solução por força bruta

- Manipulando a expressão $p - s = s - q$ obtemos
$$s = \frac{p+q}{2},$$
ou seja, o centro de simetria é o ponto médio entre p e q
- O uso de ponto flutuante pode ser evitado se usarmos a expressão
$$2s = p + q$$
- A solução de força bruta consiste em fixar um ponto A em S e, para todos os pontos B em S, computar $2s$
- Agora, para todos os pontos p em S, calculamos $q = 2s - p$ e verificamos se q pertence a S ou não

Solução por força bruta

- Manipulando a expressão $p - s = s - q$ obtemos
$$s = \frac{p+q}{2},$$
ou seja, o centro de simetria é o ponto médio entre p e q
- O uso de ponto flutuante pode ser evitado se usarmos a expressão
$$2s = p + q$$

- A solução de força bruta consiste em fixar um ponto A em S e, para todos os pontos B em S, computar $2s$
- Agora, para todos os pontos p em S, calculamos $q = 2s - p$ e verificamos se q pertence a S ou não
- Usando uma estrutura `set`, que permite verificar se q pertence a S em $O(\log n)$, temos uma solução $O(n^2 \log n)$

Solução por força bruta

- Manipulando a expressão $p - s = s - q$ obtemos
$$s = \frac{p+q}{2},$$
ou seja, o centro de simetria é o ponto médio entre p e q
- O uso de ponto flutuante pode ser evitado se usarmos a expressão
$$2s = p + q$$

- A solução de força bruta consiste em fixar um ponto A em S e, para todos os pontos B em S, computar $2s$
- Agora, para todos os pontos p em S, calculamos $q = 2s - p$ e verificamos se q pertence a S ou não
- Usando uma estrutura `set`, que permite verificar se q pertence a S em $O(\log n)$, temos uma solução $O(n^2 \log n)$
- Como $n \leq 10^4$, esta solução pode dar TLE ou AC, a depender da velocidade do servidor

Solução AC/TLE com complexidade $O(n^2 \log n)$

```cpp
#include <bits/stdc++.h>

using namespace std;

struct Point {
    int x, y;

    bool operator<(const Point& p) const {
        return x == p.x ? y < p.y : x < p.x;
    }
};

bool has_center_of_symmetry(const set<Point>& S)
{
    auto A = *S.begin();                              // Ponto qualquer de S

    for (auto& B : S) {
        auto _2s = Point { A.x + B.x, A.y + B.y };    // Candidato à centro de simetria
        bool ok = true;
```

Solução AC/TLE com complexidade $O(n^2 \log n)$

```
        // Verifica se o candidato atende o critério para todos os pontos de S
        for (auto& p : S)
        {
            auto q = Point { _2s.x - p.x, _2s.y - p.y };

            if (S.count(q) == 0) {
                ok = false;
                break;
            }
        }

        if (ok)
            return true;
    }

    return false;
}

```

Solução AC/TLE com complexidade $O(n^2 \log n)$

```cpp
int main() {
    int c, n, x, y;
    cin >> c;

    while (c--) {
        cin >> n;

        set<Point> S;

        while (n--) {
            cin >> x >> y;
            S.insert(Point { x, y });
        }

        cout << (has_center_of_symmetry(S) ? "yes" : "no") << '\n';
    }

    return 0;
}
```

Solução mais eficiente

- Para reduzir a complexidade assintótica da solução, é preciso investigar as propriedades do problema

Solução mais eficiente

- Para reduzir a complexidade assintótica da solução, é preciso investigar as propriedades do problema
- Suponha que o centro de simetria s pertença ao conjunto S. Então fazendo $p = s$ na expressão $p - s = s - q$ temos que $q = s$, ou seja, o ponto de simetria fica pareado consigo mesmo

Solução mais eficiente

- Para reduzir a complexidade assintótica da solução, é preciso investigar as propriedades do problema
- Suponha que o centro de simetria s pertença ao conjunto S. Então fazendo $p = s$ na expressão $p - s = s - q$ temos que $q = s$, ou seja, o ponto de simetria fica pareado consigo mesmo
- Como todos os pontos são distintos, então se $p \neq s$ então $q \neq s$, isto é, os pontos são pareados dois a dois

Solução mais eficiente

- Para reduzir a complexidade assintótica da solução, é preciso investigar as propriedades do problema
- Suponha que o centro de simetria s pertença ao conjunto S. Então fazendo $p = s$ na expressão $p - s = s - q$ temos que $q = s$, ou seja, o ponto de simetria fica pareado consigo mesmo
- Como todos os pontos são distintos, então se $p \neq s$ então $q \neq s$, isto é, os pontos são pareados dois a dois
- Deste modo, se existir, s só estará em S se n for ímpar

Solução mais eficiente

- Para reduzir a complexidade assintótica da solução, é preciso investigar as propriedades do problema
- Suponha que o centro de simetria s pertença ao conjunto S. Então fazendo $p = s$ na expressão $p - s = s - q$ temos que $q = s$, ou seja, o ponto de simetria fica pareado consigo mesmo
- Como todos os pontos são distintos, então se $p \neq s$ então $q \neq s$, isto é, os pontos são pareados dois a dois
- Deste modo, se existir, s só estará em S se n for ímpar
- Por um breve momento, vamos pensar no caso especial onde todos os pontos tem coordenada y igual a zero

Solução mais eficiente

- Para reduzir a complexidade assintótica da solução, é preciso investigar as propriedades do problema
- Suponha que o centro de simetria s pertença ao conjunto S. Então fazendo $p = s$ na expressão $p - s = s - q$ temos que $q = s$, ou seja, o ponto de simetria fica pareado consigo mesmo
- Como todos os pontos são distintos, então se $p \neq s$ então $q \neq s$, isto é, os pontos são pareados dois a dois
- Deste modo, se existir, s só estará em S se n for ímpar
- Por um breve momento, vamos pensar no caso especial onde todos os pontos tem coordenada y igual a zero
- Considere agora p o ponto com menor coordenada x

Solução mais eficiente

- Para reduzir a complexidade assintótica da solução, é preciso investigar as propriedades do problema
- Suponha que o centro de simetria s pertença ao conjunto S. Então fazendo $p = s$ na expressão $p - s = s - q$ temos que $q = s$, ou seja, o ponto de simetria fica pareado consigo mesmo
- Como todos os pontos são distintos, então se $p \neq s$ então $q \neq s$, isto é, os pontos são pareados dois a dois
- Deste modo, se existir, s só estará em S se n for ímpar
- Por um breve momento, vamos pensar no caso especial onde todos os pontos tem coordenada y igual a zero
- Considere agora p o ponto com menor coordenada x
- Neste cenário, podemos observar que q deve ser, obrigatoriamente, o ponto com maior coordenada x

Solução mais eficiente

- De fato, seja $r \neq q$ o ponto de maior coordenada x. Então r deve parear com um ponto t com coordenada maior do que x_p (pois os pontos são todos distintos), de modo que teremos
$$\frac{x_t + x_r}{2} > \frac{x_p + x_q}{2},$$
pois $x_p < x_t$ e $x_q < x_r$, o que impossibilita a existência de um centro de simetria

Solução mais eficiente

- De fato, seja $r \neq q$ o ponto de maior coordenada x. Então r deve parear com um ponto t com coordenada maior do que x_p (pois os pontos são todos distintos), de modo que teremos
$$\frac{x_t + x_r}{2} > \frac{x_p + x_q}{2},$$
pois $x_p < x_t$ e $x_q < x_r$, o que impossibilita a existência de um centro de simetria
- Assim, se os pontos estiverem ordenados, o primeiro deve parear com o último, de modo que é necessário verificar apenas um único candidato

Solução mais eficiente

- De fato, seja $r \neq q$ o ponto de maior coordenada x. Então r deve parear com um ponto t com coordenada maior do que x_p (pois os pontos são todos distintos), de modo que teremos
$$\frac{x_t + x_r}{2} > \frac{x_p + x_q}{2},$$
pois $x_p < x_t$ e $x_q < x_r$, o que impossibilita a existência de um centro de simetria
- Assim, se os pontos estiverem ordenados, o primeiro deve parear com o último, de modo que é necessário verificar apenas um único candidato
- Porém o fato acima foi deduzido para pontos sobre o eixo-x

Solução mais eficiente

- De fato, seja $r \neq q$ o ponto de maior coordenada x. Então r deve parear com um ponto t com coordenada maior do que x_p (pois os pontos são todos distintos), de modo que teremos
$$\frac{x_t + x_r}{2} > \frac{x_p + x_q}{2},$$
pois $x_p < x_t$ e $x_q < x_r$, o que impossibilita a existência de um centro de simetria
- Assim, se os pontos estiverem ordenados, o primeiro deve parear com o último, de modo que é necessário verificar apenas um único candidato
- Porém o fato acima foi deduzido para pontos sobre o eixo-x
- Contudo, é fácil estender o resultado para duas dimensões: uma vez ordenados por coordenada x, o ponto com menor coordenada x e menor coordenada y deve parear com o ponto com maior coordenada x e maior coordenada y, pelo mesmo motivo já apresentado

Solução mais eficiente

- De fato, seja $r \neq q$ o ponto de maior coordenada x. Então r deve parear com um ponto t com coordenada maior do que x_p (pois os pontos são todos distintos), de modo que teremos
$$\frac{x_t + x_r}{2} > \frac{x_p + x_q}{2},$$
pois $x_p < x_t$ e $x_q < x_r$, o que impossibilita a existência de um centro de simetria
- Assim, se os pontos estiverem ordenados, o primeiro deve parear com o último, de modo que é necessário verificar apenas um único candidato
- Porém o fato acima foi deduzido para pontos sobre o eixo-x
- Contudo, é fácil estender o resultado para duas dimensões: uma vez ordenados por coordenada x, o ponto com menor coordenada x e menor coordenada y deve parear com o ponto com maior coordenada x e maior coordenada y, pelo mesmo motivo já apresentado
- Assim, a solução passa a ter complexidade $O(n \log n)$

Solução AC com complexidade $O(n \log n)$

```cpp
#include <bits/stdc++.h>

using namespace std;

struct Point {
    int x, y;

    bool operator<(const Point& p) const {
        return x == p.x ? y < p.y : x < p.x;
    }
};

bool has_center_of_symmetry(const set<Point>& S)
{
    auto A = *S.begin();            // Primeiro ponto, após ordenação
    auto B = *S.rbegin();           // Último ponto, após ordenação

    // Candidato à centro de simetria
    auto _2s = Point { A.x + B.x, A.y + B.y };
```

Solução AC com complexidade $O(n \log n)$

```
    // Verifica se o candidato atende todos os pontos de S
    for (auto& p : S)
    {
        auto q = Point { _2s.x - p.x, _2s.y - p.y };

        if (S.count(q) == 0)
            return false;
    }

    return true;
}

int main()
{
    int c, n;
    cin >> c;

    while (c--) {
        cin >> n;
```

Solução AC com complexidade $O(n \log n)$

```cpp
        set<Point> S;

        while (n--)
        {
            int x, y;
            cin >> x >> y;

            S.insert(Point { x, y });
        }

        cout << (has_center_of_symmetry(S) ? "yes" : "no") << '\n';
    }

    return 0;
}
```