Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по «Алгоритмам и структурам данных»

Выполнил:

Студент группы P3212

Куприянов А.А

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2020

Блок - 2

Выполнил: Куприянов А.А.

Медианна на плоскости

1207.cpp

```cpp
#include <iostream>
#include <cmath>

using namespace std;

#define ll long long
#define PI 3.14159265358979323846

struct Point {
    ll x;
    ll y;
    double angle;
    int id;
};

const int MAX_N = 10001;
Point arr[MAX_N];

bool compare(Point p1, Point p2) {
    return p1.angle < p2.angle;
}

void quicksort(int left, int right) {
    int i = left;
    int j = right;

    Point x = arr[(left + right) / 2];

    while (i <= j) {
        while (compare(arr[i], x)) {
            i++;
        }
        while (compare(x, arr[j])) {
            j--;
        }

        if (i <= j) {
            swap(arr[i], arr[j]);

            i++;
            j--;
        }
    }
```

```
    if (i < right) {
        quicksort(i, right);
    }
    if (left < j) {
        quicksort(left, j);
    }
}

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

    ll min_x = 1e10l;
    int first = 0;

    for (int i = 0; i < n; i++) {
        ll a, b;
        cin >> a >> b;

        if (a < min_x) {
            min_x = a;
            first = i;
        }

        arr[i].x = a;
        arr[i].y = b;
        arr[i].id = i;
    }

    for (int i = 0; i < n; i++) {
        if (arr[i].id == first) {
            arr[i].angle = -1e10f;
        } else if (arr[i].x == arr[first].x) {
            arr[i].angle = (arr[i].y > arr[first].y) ? 90 : -90;
        } else {
            arr[i].angle = atan((double) (arr[i].y - arr[first].y) /
(arr[i].x - arr[first].x)) * 180.0 / PI;
        }
    }

    quicksort(0, n - 1);

    cout << first + 1 << " " << arr[n / 2].id + 1 << endl;

    return 0;
}
```

Возьмем одну точку, пусть, это будет самая правая. Теперь подсчитаем все углы от выбранной точки до остальных и отсортируем их. Таким образом, из отсортированных углов мы сможем найти точку посередине, проведя через которую прямую от нашей выбранной точки – мы получим прямую, которая делит точки так, чтобы половина была в одной, а другой с той стороны прямой

В стране дураков

1604.cpp

```cpp
#include <iostream>

using namespace std;

bool compare(pair<int, int> a, pair<int, int> b) {
    return a.second > b.second;
}

const int MAX_K = 10001;
pair<int, int> arr[MAX_K];

void quicksort(int left, int right) {
    int i = left;
    int j = right;

    pair<int, int> x = arr[(left + right) / 2];

    while (i <= j) {
        while (compare(arr[i], x)) {
            i++;
        }
        while (compare(x, arr[j])) {
            j--;
        }

        if (i <= j) {
            swap(arr[i], arr[j]);

            i++;
            j--;
        }
    }

    if (i < right) {
        quicksort(i, right);
    }
    if (left < j) {
        quicksort(left, j);
    }
}

int main() {
    int k;
    cin >> k;

    int sum = 0;

    for (int i = 0; i < k; i++) {
        arr[i].first = i + 1;
        cin >> arr[i].second;
        sum += arr[i].second;
```

```
    }

    quicksort(0, k - 1);

    int ans[sum];

    bool done = false;
    if (arr[0].second > (sum + 1) / 2) {
        int j = 1;
        int i = k - 1;

        while (j < sum) {
            ans[j] = arr[i].first;
            arr[i].second--;

            if (arr[i].second == 0) {
                i--;
            }

            j += 2;

            if (j > sum - 1 && !done) {
                j = 0;
                done = true;
            }
        }
    } else {
        int j = 0;
        int i = 0;

        while (j < sum) {
            ans[j] = arr[i].first;
            arr[i].second--;

            if (arr[i].second == 0) {
                i++;
            }

            j += 2;

            if (j > sum - 1 && !done) {
                j = 1;
                done = true;
            }
        }
    }

    for (int i = 0; i < sum; i++) {
        cout << ans[i] << " ";
    }

    return 0;
}
```

Рассмотрим два случая:

- Есть знак превышающий, количество которого превышает половину всего количества знаков (первый случай)
- Количество знаков каждого типа – меньше половины всего количества знаков (второй случай)

В первом случае, мы можем сначала расставить все знаки (кроме того, который имеет наибольшее количество знаков) через одно. А затем расставить между ними те знаки, количество которых превышает половину всех знаков (на свободные места). Таким образом, мы достигаем максимальное количество чередований.

Накормить элефпотама

1444.cpp

```
#include <iostream>
#include <cmath>

using namespace std;

#define ll long long
#define PI 3.14159265358979323846
#define EPS 1e-10

struct Point {
    ll x;
    ll y;
    double angle;
    int id;
    double len;
};

const int MAX_N = 30001;
Point arr[MAX_N];

bool compare(Point p1, Point p2) {
    if (abs(p1.angle - p2.angle) > EPS) {
        return p1.angle < p2.angle;
    }

    return p1.len < p2.len;
}

void quicksort(int left, int right) {
    int i = left;
    int j = right;

    Point x = arr[(left + right) / 2];

    while (i <= j) {
        while (compare(arr[i], x)) {
            i++;
```

```
        }
        while (compare(x, arr[j])) {
            j--;
        }

        if (i <= j) {
            swap(arr[i], arr[j]);

            i++;
            j--;
        }
    }

    if (i < right) {
        quicksort(i, right);
    }
    if (left < j) {
        quicksort(left, j);
    }
}

int main() {
    int n;

    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> arr[i].x;
        cin >> arr[i].y;
        arr[i].id = i;

        if (i == 0) {
            arr[i].angle = INT64_MIN;
            arr[i].len = 0;
            continue;
        }

        if (arr[i].x == arr[0].x) {
            arr[i].angle = (arr[i].y > arr[0].y) ? 90 : -90;
            arr[i].len = abs(arr[i].y - arr[0].y);
            continue;
        }

        arr[i].angle = atan((double) (arr[i].y - arr[0].y) / (arr[i].x
- arr[0].x)) * 180.0 / PI;
        if (arr[i].x <= arr[0].x) {
            arr[i].angle -= 180;
        }

        arr[i].len = abs(sqrt((double) (pow((arr[i].x - arr[0].x), 2)
+ pow((arr[i].y - arr[0].y), 2))));
    }

    quicksort(0, n - 1);

    double max_a = 360 + arr[1].angle - arr[n - 1].angle;
```

```
    int k = 1;

    for (int i = 1; i < n - 1; i++) {
        if (arr[i + 1].angle - arr[i].angle > max_a) {
            max_a = arr[i + 1].angle - arr[i].angle;
            k = i + 1;
        }
    }

    cout << n << endl; // all
    cout << 1 << endl; // print first food

    for (int i = k; i < n; i++) {
        cout << arr[i].id + 1 << endl;
    }

    for (int i = 1; i < k; i++) {
        cout << arr[i].id + 1 << endl;
    }

    return 0;
}
```

Возьмем углы от начальной тыквы до остальных и отсортируем их. Теперь мы можем проходится по ним от меньшей к большему, при этом не пересекая своих следов. В том случае, если углы совпадают – берем ближний.