Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Решения задач timus

Алгоритмам и структурам данных

Работу выполнил:

Гаврилин О.С.

Группа:

P3230

Санкт-Петербург,

2025

1.1604

```
#include <iostream>
#include <vector>
#include <limits>
using namespace std;

pair<int, int> getMinMaxIndices(const
vector<pair<int, int>>& signs) {
    int minIdx = -1, maxIdx = -1;
    int minAmt = numeric_limits<int>::max(), maxAmt
= numeric_limits<int>::min();
    for (size_t i = 0; i < signs.size(); i++) {
        if (signs[i].first > 0) {
            if (signs[i].first >= maxAmt) {
                maxAmt = signs[i].first;
                maxIdx = i;
            }
            if (signs[i].first < minAmt) {
                minAmt = signs[i].first;
                minIdx = i;
            }
        }
    }
    return {minIdx, maxIdx};
}

void solve() {
    int k;
    cin >> k;
    vector<pair<int, int>> signs(k);
    int totalSigns = 0;
    for (int i = 0; i < k; i++) {
        cin >> signs[i].first;
        signs[i].second = i + 1;
        totalSigns += signs[i].first;
    }
    while (totalSigns > 0) {
```

```
        auto [minIdx, maxIdx] =
getMinMaxIndices(signs);
        if (maxIdx != -1 && signs[maxIdx].first >
0) {
            cout << signs[maxIdx].second << ' ';
            signs[maxIdx].first--;
            totalSigns--;
        }
        if (minIdx != -1 && signs[minIdx].first >
0) {
            cout << signs[minIdx].second << ' ';
            signs[minIdx].first--;
            totalSigns--;
        }
    }
}
int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    solve();
}
```

Описание: Выводим индексы, чередуя знаки, которые встречаются чаще всего и реже всего во входных данных. На каждом шаге находим индексы максимального и минимального количества и уменьшаем их.
Сложность по времени: O(n*k)
Сложность по памяти: O(k)

2.1444

```
#include <iostream>
#include <vector>
#include <cmath>
#include <algorithm>

using namespace std;
#define PI 3.142
const double NEGATIVE_INFINITY = -1e10;
```

```
struct Coord {
    long x, y;
    double angle;
    int index;
};

int totalPoints;
vector<Coord> coords;

int findMaxGapIndex() {
    double maxDifference = (coords[1].angle -
coords.back().angle) + 360;
    int startingIndex = 1;
    for (size_t i = 1; i < coords.size() - 1; ++i) {
        double difference = coords[i + 1].angle -
coords[i].angle;
        if (difference > maxDifference) {
            maxDifference = difference;
            startingIndex = i + 1;
        }
    }
    return startingIndex;
}

bool compareByAngle(const Coord& p1, const Coord&
p2) {
    if (abs(p1.angle - p2.angle) > 1e-10)
        return p1.angle < p2.angle;
    long distance1 = (p1.x - coords[0].x) * (p1.x -
coords[0].x) + (p1.y - coords[0].y) * (p1.y -
coords[0].y);
    long distance2 = (p2.x - coords[0].x) * (p2.x -
coords[0].x) + (p2.y - coords[0].y) * (p2.y -
coords[0].y);
    return distance1 < distance2;
}
```

```
void process() {
    cin >> totalPoints;
    coords.resize(totalPoints);
    for (int i = 0; i < totalPoints; ++i) {
        cin >> coords[i].x >> coords[i].y;
        coords[i].index = i;
        if (i == 0) {
            coords[i].angle = NEGATIVE_INFINITY;
        } else {
            coords[i].angle  =  atan2(coords[i].y  -
coords[0].y,  coords[i].x  -  coords[0].x)  *  180.0  /
PI;
        }
    }
    sort(coords.begin()    +    1,    coords.end(),
compareByAngle);

    cout << totalPoints << "\n1\n";
    int startIdx = findMaxGapIndex();
    for (int i = startIdx; i < totalPoints; ++i) {
        cout << coords[i].index + 1 << "\n";
    }
    for (int i = 1; i < startIdx; ++i) {
        cout << coords[i].index + 1 << "\n";
    }
}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    process();
}
```

Описание: Сортируем точки на плоскости по углам, относительно первой заданной точки. Затем, основываясь на максимальном промежутке углов между

соседними точками выбирается следующая точка обхода, обходим точки по часовой стрелке.
Сложность по времени: O(n*log(n))
Сложность по памяти: O(n)