Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Лабораторная работа № 3
"Задачи 1207, 1322, 1444"

по дисциплине Алгоритмы и структуры данных

Выполнила: студентка гр. **R3238**
поток **2.1**

Нечаева А. А.

Преподаватель: *Тропченко Андрей Александрович*

Санкт-Петербург, 2024

1 Цель

Разработать и реализовать алгоритмы для решения задач 1207, 1322 и 1444.

2 Задача 1207

1207. Медиана на плоскости

Ограничение времени: 0.5 секунды

Ограничение памяти: 64 МБ

На плоскости находятся N точек (N чётно). Никакие три точки не лежат на одной прямой. Ваша задача — выбрать две точки так, что прямая линия, проходящая через них, делит множество точек на две части одинакового размера.

Исходные данные

Первая строка содержит целое число N ($4 \le N \le 10\,000$). Каждая из следующих N строк содержит пары целых чисел x_i, y_i ($-10^6 \le x_i, y_i \le 10^6$) — координаты i-й точки.

Результат

Выведите номера выбранных точек.

Пример

исходные данные	результат
4 0 0 1 0 0 1 1 1	1 4

Автор задачи: Павел Атнашев

Источник задачи: Соревнование команд УрГУ, март 2002

Рис. 1. Условие задачи 1207.

2.1 Краткое описание алгоритма

1. Входные данные: первая строка содержит целое число $N\,(4 \le N \le 10000)$. Каждая из следующих N строк содержит пары целых чисел x_i, y_i $(-10^6 \le x_i, y_i \le 10^6)$– координаты i-й точки.
2. Для разделения плоскости на 2 части необходимо найти точку, которая расположена левее всех остальных.
3. Далее определим угол прямой, проведенной до каждой из оставшихся точек и отсортируем точки по величине угла.
4. Теперь мы можем определить точку, через которую проходит прямая, разделяющая точки на плоскости пополам.
5. Выходные данные: порядковые номера 2-х точек, прямая через которые разделитт все точки плоскости на 2 равные половины.

2.2 Листинг

Листинг 1. Исходный код для 1207

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

#define PI 3.14159265358979323846

// new struct for points, it keeps x, y, number of the point
    and angle
struct point_struct {
    int x;
    int y;
    double angle;
    int num;
};

point_struct data[10000];

bool compare(point_struct a, point_struct b) {
    return a.angle < b.angle;
}

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

    point_struct median = data[(left + right) / 2];

    while (i <= j) {
        while (compare(data[i], median)) {
            ++i;
        }
        while (compare(median, data[j])) {
            --j;
        }
        if (i <= j) {
            std::swap(data[i], data[j]);
            ++i;
            --j;
        }
    }
```

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

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

    int first_x = 1000000;
    int first_id = 0;

    for (int i = 0; i < n; ++i) {
        int p_x;
        int p_y;

        std::cin >> p_x >> p_y;

        if (p_x < first_x) {
            first_id = i;
            first_x = p_x;
        }

        data[i].x = p_x;
        data[i].y = p_y;
        data[i].num = i;
    }

    // here we analyze the angles
    for (int i = 0; i < n; ++i) {
        if (data[i].num == first_id) {
            data[i].angle = -360;
        } else {
            if (data[i].x == data[first_id].x) {
                if (data[i].y > data[first_id].y) {
                    data[i].angle = 90;
                } else {
                    data[i].angle = -90;
                }
            } else {
```

```
                    data[i].angle = atan((double) (data[i].y - data[first_id].y) / (data[i].x - data[first_id].x)) * 180.0 / PI;
                }
            }
        }

    quick_sort(0, n - 1);

    std::cout << first_id + 1 << " " << data[n / 2].num + 1 << std::endl;

    return 0;
}
```

2.3 Результат

Автор: mistDragon • Задача: Медиана на плоскости

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10587105	16:55:35 15 мар 2024	mistDragon	1207. Медиана на плоскости	Clang++ 17 x64	Accepted		0.046	620 КБ
[illegible]	16:53:44 [illegible]	[illegible]	[illegible]	[illegible]	[illegible]			

Рис. 2. Результат отправки задачи 1207.

3 Задача 1322

1322. Шпион

Ограничение времени: 0.25 секунды
Ограничение памяти: 64 МБ

Спецслужбы обнаружили действующего иностранного агента. Шпиона то есть. Установили наблюдение и выяснили, что каждую неделю он через Интернет посылает кому-то странные нечитаемые тексты. Чтобы выяснить, к какой информации получил доступ шпион, требуется расшифровать информацию. Сотрудники спецслужб проникли в квартиру разведчика, изучили шифрующее устройство и выяснили принцип его работы.

На вход устройства подается строка текста $S_1 = s_1s_2...s_N$. Получив ее, устройство строит все циклические перестановки этой строки, то есть $S_2 = s_2s_3...s_Ns_1$, ..., $S_N = s_Ns_1s_2...s_{N-1}$. Затем множество строк S_1, S_2, ..., S_N сортируется лексикографически по возрастанию. И в этом порядке строчки выписываются в столбец, одна под другой. Получается таблица размером N × N. В какой-то строке K этой таблицы находится исходное слово. Номер этой строки вместе с последним столбцом устройство и выдает на выход.

Например, если исходное слово S_1 = abracadabra, то таблица имеет такой вид:

1. aabracadabr = S_{11}
2. abraabracad = S_8
3. abracadabra = S_1
4. acadabraabr = S_4
5. adabraabrac = S_6
6. braabracada = S_9
7. bracadabraa = S_2
8. cadabraabra = S_5
9. dabraabraca = S_7
10. raabracadab = S_{10}
11. racadabraab = S_3

И результатом работы устройства является число 3 и строка rdarcaaaabb.

Это все, что известно про шифрующее устройство. А вот дешифрующего устройства не нашли. Но поскольку заведомо известно, что декодировать информацию можно (а иначе зачем же ее передавать?), Вам предложили помочь в борьбе с хищениями секретов и придумать алгоритм для дешифровки сообщений. А заодно и реализовать дешифратор.

Исходные данные

В первой и второй строках находятся соответственно целое число и строка, возвращаемые шифратором. Длина строки и число не превосходят 100000. Строка содержит лишь следующие символы: a-z, A-Z, символ подчеркивания. Других символов в строке нет. Лексикографический порядок на множестве слов задается таким порядком символов:

ABCDEFGHIJKLMNOPQRSTUVWXYZ_abcdefghijklmnopqrstuvwxyz

Символы здесь выписаны в порядке возрастания.

Результат

Выведите декодированное сообщение в единственной строке.

Пример

исходные данные	результат
3 rdarcaaaabb	abracadabra

Автор задачи: Идея — Александр Клепинин, подготовка — Александр Клепинин, Станислав Васильев
Источник задачи: VIII Командный студенческий чемпионат Урала по программированию. Екатеринбург, 11-16 марта 2004 г.

Рис. 3. Условие задачи 1322.

3.1 Основная идея

Задача сводится к работе с **преобразованием Барроуза–Уилера**, в нашем случае необходимо выполнить обратное преобразование.

3.2 Краткое описание алгоритма

1. Входные данные: целое число s ($s \leq 100000$) и строка str.
2. Последовательно восстанавливаем строку. Пусть изначально известно, каким по порядку является приписанный в начало символ (его порядок в столбце)
3. Из предыдущего шага известно, какое место занимала строка без первого символа (i-ое).
4. Несложно заметить, что при выполнении такой операции строка с номером i всегда будет перемещаться на позицию с номером j.
5. Выходные данные: строка.

3.3 Листинг

Листинг 2. Исходный код для 1322

```
#include <iostream>


const int maximum_n = 100000;

std::pair<char, int> data[maximum_n];

bool comparator(std::pair<char, int> first, std::pair<char,
    int> second) {
    return (first.first != second.first) ? first.first <
        second.first : first.second < second.second;
}

//sort in lexicographic order (quick sort)
void sorter(int left, int right) {
    int i = left;
    int j = right;

    std::pair<char, int> cur = data[(left + right) / 2];

    while (i <= j) {
        while (comparator(data[i], cur)) {
            ++i;
        }
        while (comparator(cur, data[j])) {
            --j;
```

```
        }

        if (i <= j) {
            std::swap(data[i++], data[j--]);
        }
    }

    if (i < right) sorter(i, right);

    if (j > left) sorter(left, j);

}

int main() {
    int s;
    std::string str;

    std::cin >> s >> str;

    int symbols_counter = str.length();

    for (int i = 0; i < symbols_counter; ++i) {
        data[i].first = str[i];
        data[i].second = i;
    }

    sorter(0, symbols_counter - 1);

    int result[symbols_counter];

    for (int i = 0; i < symbols_counter; ++i) {
        result[i] = data[i].second;
    }

    int current = s - 1;

    for (int i = 0; i < symbols_counter; ++i) {
        std::cout << data[current].first;
        current = result[current];
    }

    return 0;
}
```

3.4 Результат

Рис. 4. Результат отправки задачи 1322.

4 Задача 1444

1444. Накормить элефпотама

Ограничение времени: 0.5 секунды
Ограничение памяти: 64 МБ

Гарри Поттер сдаёт экзамен по предмету «Уход за магическими существами». Его задание — накормить карликового элефпотама. Гарри помнит, что элефпотамы отличаются прямолинейностью и невозмутимостью. Они настолько прямолинейны, что ходят строго по прямой, и настолько невозмутимы, что заставить их идти можно, только если привлечь его внимание к чему-нибудь действительно вкусному. И главное, наткнувшись на цепочку своих собственных следов, элефпотам впадает в ступор и отказывается идти куда-либо. По словам Хагрида, элефпотамы обычно возвращаются домой, идя в обратную сторону по своим собственным следам. Поэтому они никогда не пересекают их, иначе могут заблудиться. Увидев свои следы, элефпотам детально вспоминает все свои перемещения от выхода из дома (поэтому-то они и ходят только по прямой и лишний раз не меняют направление — так легче запоминать). По этой информации элефпотам вычисляет, в какой стороне расположена его нора, после чего поворачивается и идет прямо к ней. Эти вычисления занимают у элефпотама некоторое (довольно большое) время. А то, что некоторые невежды принимают за ступор, на самом деле есть проявление выдающихся вычислительных способностей этого чудесного, хотя и медленно соображающего животного!

Любимое лакомство элефпотамов — слоновьи тыквы, именно они и растут на лужайке, где Гарри должен сдавать экзамен. Перед началом испытания Хагрид притащит животное к одной из тыкв. Скормив элефпотаму очередную тыкву, Гарри может направить его в сторону любой оставшейся тыквы. Чтобы сдать экзамен, надо провести элефпотама по лужайке так, чтобы тот съел как можно больше тыкв до того, как наткнется на свои следы.

Исходные данные

В первой строке входа находится число N ($3 \le N \le 30000$) — количество тыкв на лужайке. Тыквы пронумерованы от 1 до N, причем номер один присвоен той тыкве, у которой будет стоять элефпотам в начале экзамена. В следующих N строках даны координаты всех тыкв по порядку. Все координаты — целые числа от −1000 до 1000. Известно, что положения всех тыкв различны, и не существует прямой, проходящей сразу через все тыквы.

Результат

В первой строке выхода вы должны вывести K — максимальное количество тыкв, которое может съесть элефпотам. Далее по одному числу в строке выведите K чисел — номера тыкв в порядке их обхода. Первым в этой последовательности всегда должно быть число 1.

Пример

исходные данные	результат
4 0 0 10 10 0 10 10 0	4 1 3 2 4

Автор задачи: Идея и текст: Екатерина Васильева, программирование: Александр Мироненко, Алексей Лахтин, Ден Расковалов
Источник задачи: X командный Чемпионат Урала по спортивному программированию, 24-25 марта 2006 года

Рис. 5. Условие задачи 1444.

4.1 Краткое описание алгоритма

1. Входные данные: в первой строке находится целое цисло N ($3 \le N \le 30000$) – количество тыкв на лужайке. Тыквы пронумерованы от 1 до

N, причем номер один присвоен той тыкве, у которой будет стоять элефпотам в начале экзамена. В следующих N строках даны координаты всех тыкв по порядку. Все координаты – целые числа от -1000 до 1000. Известно, что положения всех тыкв различны, и не существует прямой, проходящей через все тыквы сразу.

2. Сначала найдем уголы, под которыми расположены все тыквы от начальной.

3. Отсортируем тыквы по значению угла. При совпадении значения угла у нескольких тыкв элефпотам сначала съест ту, которая находится на наименьшем расстоянии, далее он будет двигаться вдоль прямой до остальных тыкв.

4. Все тыквы отсортированы по величине угла от начальной, значит, пока элефпотам идет от одной тыквы до следующей, он не пересечет следы. Важно проверять, что разность между соседними углами лучей от начальной тыквы всегда меньше 180 градусов. Таким образом, элефпотам может обойти все тыквы.

5. Выходные данные: В первой строке выхода вывести K – максимальное количество тыкв, которое может съесть элефпотам. Далее по одному числу в строке вывести K чисел – номера тыкв в порядке их обхода. Первый в этой последовательности всегда должно быть число 1.

4.2 Листинг

Листинг 3. Исходный код для 1444

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

using namespace std;

#define PI 3.14159265358979323846
#define EPS 1e-10

// new struct for points, it keeps x, y, number of the point,
    length and angle
struct point_struct {
    int x;
    int y;
    double angle;
    int id;
    double len;
```

```
};

const int MAX_N = 30001;
point_struct data_points[MAX_N];

bool compare(point_struct a, point_struct b) {
    if (abs(a.angle - b.angle) > EPS) {
        return a.angle < b.angle;
    }

    return a.len < b.len;
}

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

    point_struct x = data_points[(left + right) / 2];

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

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

            i++;
            j--;
        }
    }

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

```

```
int main() {
    int n;

    cin >> n;

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

        if (i == 0) {
            data_points[i].angle = INT_MIN;
            data_points[i].len = 0;
        } else if (data_points[i].x == data_points[0].x){
            if (data_points[i].y  > data_points[0].y) {
                data_points[i].angle = 90;
            } else {
                data_points[i].angle = -90;
            }
            data_points[i].len = abs(data_points[i].y -
                data_points[0].y);
        } else {
            data_points[i].angle = atan((double) (data_points[
                i].y - data_points[0].y) / (data_points[i].x -
                 data_points[0].x)) * 180.0 / PI;
            if (data_points[i].x <= data_points[0].x) {
                data_points[i].angle -= 180;
            }
            data_points[i].len = abs(sqrt((double) (pow((
                data_points[i].x - data_points[0].x), 2) + pow
                ((data_points[i].y - data_points[0].y), 2))));
        }
    }

    quick_sort(0, n - 1);

    double max_a = 360 + data_points[1].angle - data_points[n
        - 1].angle;
    int counter = 1;

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

```
            max_a = data_points[i + 1].angle - data_points[i].angle;
            counter = i + 1;
        }
    }

    cout << n << endl;
    cout << 1 << endl;

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

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

    return 0;
}
```

4.3 Результат

Рис. 6. Результат отправки задачи 1444.

5 Вывод по работе

В ходе выполнения данной лабораторной работы были реализованы алгоритмы для решения задач 1207, 1322 и 1444. Все эти задачи объединяет использование алгоритма быстрой сортировки массива ***quicksort***.