Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Направление подготовки бакалавриата

09.03.04 — Программная инженерия

Профиль направления подготовки бакалавриата

Системное и прикладное программное обеспечение

Отчет о прохождении производственной практики

Чемпионат по алгоритмическому программированию Rucode 6

Выполнила:
студентка 3 курса группы 22307

А. Р. Хуснутдинова _____
подпись

Место прохождения практики:
кафедра ИМО

Период прохождения практики:

15.05.23 - 21.05.23

Руководитель:
О. Ю. Богоявленская

_____
подпись

Итоговая оценка:

_____
оценка

Петрозаводск — 2023

Содержание

Введение

С 15.05.23 по 19.05.23 в рамках Всероссийского учебного фестиваля по искуственному интеллекту и программированию Rucode проводились интенсивы и чемпионат по направлениям "Искусственный интеллект"и "Алгоритмическое программирование". Для участия в фестивале было выбрано направление "Алгоритмическое программирование".

Интенсивы длились на протяжении 5-ти дней и завершались чемпионатом. Для участия командой из трёх человек в составе Хуснутдинова Айгуль, Рейкенен Ярослав, Горчаков Роберт был выбран дивизион C/D.

Цель практики: получение и применение теоретических знаний в области алгоритмического программирования

Задачи практики:

- Получение теоретических знаний в области алгоритмического программирования в ходе участия в интенсивах Rucode;
- Применение знаний в области алгоритмического программирования при решении олимпиадных задач;
- Организация командной работы для эффективного решения задач.

1 Чемпионат

Фестиваль Rucode 6 проводился в онлайн-формате с присутствием очных точек по всей стране, одной из которых являлся Петрозаводский государственный университет. Организацией очной точки занимались участники и тренера Клуба Творчества Программистов.

Чемпионат проходил с 10:00 до 15:00 21.05.23 в Главном Корпусе ПетрГУ. По истечению соревнования участники отправлялись на разбор чемпионата и объявление результатов.

В ходе чемпионата мною были решены 3 задачи из 7, далее описываются идеи решения непосредственно этих задач.

2 Задачи чемпионата

2.1 Задача C. «Выделяем римские анаграммы»

Полное условие задачи C в Приложении А. [1]

2.1.1 Условие

В задачи было дано конечное число тестов. Каждый тест представлял собой строку из букв, используемых при записи римских чисел. Для каждого теста требовалось вывести количество перестановок исходной строки, которые являются корректными римскими числами.

2.1.2 Алгоритм решения

Решение задачи основывалось на следующем алгоритме. Для каждого числа от 1 до 3999 (количество римских чисел не превышает значения 3999, как сказано в условиях) получим его римскую запись. Данную строку отсортируем и для полученного значения увеличим значение в словаре. Таким образом мы получим словарь, в котором каждой отсортированной строке будет соответсвовать количество её перестановок, являющихся корректным римским числом. Для следующих тестов отсортируем полученную строку и получим для неё ответ из предподсчитанного словаря.

2.1.3 Решение

Задача решалась на языке C++.

Листинг 1 – C. Выделяем римские анаграммы

```
string st;
int n, t;
map<char, int> val;

string rev[10][15];

map<string, int> mp;

string cnt(int v)
```

[1] Источник данной и нижеуказанных задач: https://rucode.net/

```
{
        int a = 0;
        string res = "";
        while(v > 0)
        {
                int r = v % 10;
                v /= 10;
                res += rev[a][r];
                a++;
        }
        return res;
}

void solve()
{
        cin >> t;

        val['I'] = 1;
        val['V'] = 5;
        val['X'] = 10;
        val['L'] = 50;
        val['C'] = 100;
        val['D'] = 500;
        val['M'] = 1000;

        rev[0][0] = "";
        rev[0][1] = "I";
        rev[0][2] = "II";
        rev[0][3] = "III";
        rev[0][4] = "IV";
        rev[0][5] = "V";
        rev[0][6] = "VI";
        rev[0][7] = "VII";
```

```
rev[0][8] = "VIII";
rev[0][9] = "IX";


rev[1][0] = "";
rev[1][1] = "X";
rev[1][2] = "XX";
rev[1][3] = "XXX";
rev[1][4] = "XL";
rev[1][5] = "L";
rev[1][6] = "LX";
rev[1][7] = "LXX";
rev[1][8] = "LXXX";
rev[1][9] = "XC";

rev[2][0] = "";
rev[2][1] = "C";
rev[2][2] = "CC";
rev[2][3] = "CCC";
rev[2][4] = "CD";
rev[2][5] = "D";
rev[2][6] = "DC";
rev[2][7] = "DCC";
rev[2][8] = "DCCC";
rev[2][9] = "CM";

rev[3][0] = "";
rev[3][1] = "M";
rev[3][2] = "MM";
rev[3][3] = "MMM";

for (int i = 1; i <= 3999; i++)
{
```

```
            string cur = cnt(i);
            sort(cur.begin(), cur.end());
            mp[cur]++;
        }


        while(t--)
        {
            n = 0;
            int k = 0;
            cin >> st;

            sort(st.begin(), st.end());

            k = mp[st];

            cout << k << "\n";
        }
}
```

Задача была сдана со второй попытки. В первоначальном варианте не происходило предподсчёта словаря, и пересчёт подъодящих вариантов проводился на каждом тесте, что не укладывалось в ограничения в 1 секунду, и решение получило вердикт «TL» - Time Limit (Превышение ограничения по времени).

Со второй попытки задача получила результат «OK».

2.2 Задача D. «Графопостроитель»

Полное условие задачи D в Приложении Б.

2.2.1 Условие

По условию задачи на плоскости совершались построения отрезков. Было известно, что никакие два построенных отрезков не пересекаются или пересекаются строго в одной точке, не являющейся внутренней точкой ни для одного из сегментов. На каждом шагу на плоскосте появлялся новый отрезок. Считалось, что два отрезка принадлежат одной фигуре, "если отрезки соединенны прямо или опосредовано, то есть существует последовательность отрезков с одним из этих двух отрезков как первым элементом, вторым — как последним, обладающая следуюзим свойством: любые два отрезка, соседние в этой последовательности, имеют общую вершину"[2]. Фигура называлась забавной, если можно было нарисовать её, не отрывая карандаша от бумаги и без рисования какого-либо участка дважды. Концы отрезков можно было посещать любое число раз.

В качестве входных данных было дано количество шагов, для каждого шага задавались начало и конец отрезка. Для каждого шага было необходимо вывести количество забавных фигур.

2.2.2 Алгоритм решения

Сведём входные данные к графу, в котором вершинами будут являться концы проведённых отрезков, а наличием рёбра - наличие отрезка между вершинами. Таким образом, сведём задачу к нахождению Эйлерова пути для каждой фигуры. «Эйлеров путь - это путь в графе, проходящий через все его рёбра» [3] По теореме о существовании Эйлерова пути Эйлеров путь существует тогда и только тогда, когда количество вершин с нечётными степенями равно двум или нулю.

Для нахождения фигур использовалась система непересекающихся множеств (далее - СНМ). При проведении отрезка каждая точка, если раннее она не была добавлена в СНМ, объявлялась отдельным множеством. При проведении отрезка множества соответствующих точек объединялись. Также при создании новых рёбер подсчитывалась степень вершин отрезка и обновлялось число нечётных вершин отрезка. В зависимости от того, на-

[2]Условия задачи С, Приложение Б

[3]Нахождение Эйлерова пути за O(M) [Электронный ресурс] // URL: http://e-maxx.ru/algo/euler$_path$(: 07.06.23)

ходился ли Эйлеров путь в полученной фигуре, обновлялось общее число забавных фигур, что являлось текущим ответом

2.2.3 Решение

Задача решалась на языке C++.

Листинг 2 – D. Графопостроитель

```
map<pair<int,int>, int> g;
#define map unordered_map
map<int, int> parent;
map<int, int> rankk;
map<int, int> cnt;
map<int, int> cnt1;

int ans = 0;
void make_set (int v) {
        cnt[v] = 0;
        cnt1[v] = 0;
        ans++;
        parent[v] = v;
        rankk[v] = 0;
}

int find_set (int v) {
        if (v == parent[v])
                return v;
        return parent[v] = find_set (parent[v]);
}
void union_sets (int a, int b) {
        int v1 = a, v2 = b;

        a = find_set (v1);
        b = find_set (v2);
```

```
if (a == b) {
        if (cnt1[a] == 2 || cnt1[a] == 0) ans---;

        if (cnt[v1] & 1) cnt1[a]---;
        if (cnt[v2] & 1) cnt1[b]---;
        cnt[v1]++;
        cnt[v2]++;
        if (cnt[v1] & 1) cnt1[a]++;
        if (cnt[v2] & 1) cnt1[b]++;

        if (cnt1[a] == 2 || cnt1[a] == 0) ans++;
        return;
}

if (cnt1[a] == 2 || cnt1[a] == 0) ans---;
if (cnt1[b] == 2 || cnt1[b] == 0) ans---;

if (cnt[v1] & 1) {
        cnt1[a]---;
}

if (cnt[v2] & 1) {
        cnt1[b]---;
}

cnt[v1]++; cnt[v2]++;

if (cnt[v1] & 1) {
        cnt1[a]++;

}
if (cnt[v2] & 1) {
        cnt1[b]++;
```

```
        }
        if ((cnt1[a] + cnt1[b]) == 2 || (cnt1[a] + cnt1[b]) == 0) {
                ans++;
        }
        if (a != b) {
                if (rankk[a] < rankk[b])
                        swap (a, b);
                parent[b] = a;
                cnt1[a] += cnt1[b];

                if (rankk[a] == rankk[b])
                        ++rankk[a];
        }

}

void solve()
{
        int n;
        cin >> n;
        int count = 1;
        for (int i = 0; i < n; i++) {
                int x1, y1, x2, y2;
                cin >> x1 >> y1 >> x2 >> y2;
                if (g[pii{x1, y1}] == 0) {
                        g[pii{x1, y1}] = count++; make_set(g[pii{x1,y1}])
                }
                if (g[pii{x2, y2}] == 0) {
                        g[pii{x2, y2}] = count++; make_set(g[pii{x2,y2}])
                }

                union_sets(g[pii{x1,y1}], g[pii{x2,y2}]);
```

```
                cout << ans << endl;
        }

}
```

Задача была сдана со второй попытки, первый раз решение получило вердикт «TL» - Time Limit (Превышение ограничения по времени).

Со второй попытки задача получила результат «OK».

2.3 Задача F. «Если лодку украли...»

Полное условие задачи F в Приложении B.

2.3.1 Условие

Имеется ящик в форме прямоугольного параллелепипеда с грузом внутри. Ящик плывёт по воде. Необходимо найти минимальные длину, ширину и высоту ящика, имея следующие данные:

- Длина, ширина и высота измеряются в сантиметрах и являются целыми числами;
- Ширина лодки равна 1/3 от длины;
- Высота равна 1/2 от ширины;
- Ящик всегда весит 50 килограмм;
- Груз весит W килограмм (значение даётся в качестве входных данных);
- Ящик с грузом плавает, если суммарный вес ящика и груза в точности равен весу воды, вытесненной ящиком;
- Верхняя сторона ящика и линия воды параллельны;
- Растояние от верхней стороны ящика до линии контакта с водой не менее 30 сантиметров;
- 1000 кубических сантиметров воды весят 1 килограмм.

2.3.2 Алгоритм решения

Из известных данных была выведена следующая формула:

$$50 + W = \frac{12xd^2}{1000}, \tag{1}$$

где W — масса груза, x — глубина погружения ящика в воду, d — высота ящика. Выразив из формулы x, мы перебирали все возможные значения d, начиная с 1, пока разница между x и d не становилась больше или равной 30. Высота и ширина ящика находились как 6*d и 2*d соотвественно.

2.3.3 Решение

Задача решалась на языке C++.

Листинг 3 – D. Графопостроитель

```
ll d, w, val, h;
const long double eps = 0.0000000001;

void solve()
{
        cin >> w;

        for (ll i = 1; i <= 1050000; i++)
        {
                long double x = (50000 + w*1000) / (12 * i * i);
                if ((long double)i - 30 - x >= eps)
                {
                        h = i;
                        break;
                }
        }
        cout << 6*h << ' ' << 2*h << ' ' << h;
}
```

Задача была сдана со первой попытки, получив вердикт «OK».

Заключение

В результате участия в чемпионате командой было решено 7 задач. В процессе решения задач были применены знания в области алгоритмического программирования, полученные во время участия в интенсивах Rucode и обучению в Клубе Творчества Программистов. Благодаря слаженной работе в команде и распределению её членов по задачам все участники смогли применить свои знания на практике. По итогам чемпионата команда заняла 34 место из 227, получив диплом II степени (Приложение Г).

Список литературы

1. Нахождение Эйлерова пути за O(M) [Электронный ресурс]. — URL: http://e-maxx.ru/algo/euler_path (дата обращения: 07.06.23)

3 Приложение А

C. Выделяем римские анаграммы

Ограничение времени	1 секунда
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Вам дана строка S, составленная из букв `I`, `V`, `X`, `L`, `C`, `D` и `M`.
Ваша задача — подсчитать количество различных анаграмм этой строки, которые являются корректными римскими числами (сама строка также считается своей анаграммой).
Таблица из Википедии показывает, каким образом записываются римские числа:

Разряды	Тысячи	Сотни	Десятки	Единицы
1	M	C	X	I
2	MM	CC	XX	II
3	MMM	CCC	XXX	III
4		CD	XL	IV
5		D	L	V
6		DC	LX	VI
7		DCC	LXX	VII
8		DCCC	LXXX	VIII
9		CM	XC	IX

Заметим, что:

- Обозначения для 4, 9, 40, 90, 400 и 900 используют так называемую *инвертированную запись*, когда первый символ вычитается из последующего (например, для for 40 (`XL`) `X` (10) вычитается из `L` (50)). Это **единственные** случаи инвертированной записи.
- Число, содержащее несколько десятичных цифр, строится конкатенацией римской записи каждой цифры, слева направо от большей к меньшей.
- Если в каком-то разряде стоит 0, то в соответствующем месте ничего не приписывается.
- Наибольшее число, которое может быть представлено в римской записи, это 3,999 (MMMCMXCIX).

Рис. 1 – Задача C, условия

Формат входных данных

Первая строка входных данных содержит одно целое число T — количество тестовых примеров ($1 \leqslant T \leqslant 77\,777$).
Каждая из последующих T строк содержит один тестовый пример — слово s, состоящее из букв `I`, `V`, `X`, `L`, `C`, `D` и `M` ($1 \leqslant |s| \leqslant 15$).

Формат выходных данных

Для каждого тестового примера выведите одно целое число — количество анаграмм заданной строки, являющихся корректными римскими числами.

Пример

стандартный ввод	стандартный вывод
2 IC XV	1 1

Пример

Ввод	Вывод
2 IC XV	1 1

Рис. 2 – Задача C, описание данных

4 Приложение Б

D. Графопостроитель

Ограничение времени	2 секунды
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Графопостроитель строит чертёж на бумаге.
Графопостроитель работает следующим образом:

- За один шаг рисуется один отрезок.
- Если новый отрезок пересекается с каким-то из существующих отрезков, то он пересекается ровно по одной точке и точка пересечения не может быть внутренней точкой ни для какого из отрезков.

Два отрезка считаются принадлежащими одной *фигуре*, если отрезки соединены прямо или опосредованно, то есть существует последовательность отрезков с одним из этих двух отрезков как первым элементом, вторым — как последним, обладающая следующим свойством: любые два отрезка, соседние в этой последовательности, имеют общую вершину.
После каждого шага, тем самым, на бумаге нарисовано несколько фигур.
Назовём фигуру *забавной*, если она может быть нарисована за один росчерк, то есть можно нарисовать фигуру целиком без отрыва карандаша от бумаги и без рисования какого-либо участка ненулевой длины дважды. При этом **не имеет значения**, сколько раз были посещены концы отрезков.
Ваша задача — вывести количество забавных фигур на бумаге после каждого шага графопостроителя.

Рис. 3 – Задача D, условия

Формат входных данных

Первая строка входных данных содержит одно целое число N — количество шагов графопостроителя ($1 \leqslant N \leqslant 2 \cdot 10^5$).

i-я из последующих N строк задаёт i-й шаг и содержит четыре целых числа $x_{i,1}$, $y_{i,1}$, $x_{i,2}$, $y_{i,2}$ — координаты концов отрезка, начерченного графопостроителем на i-м шаге ($0 \leqslant x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2} \leqslant 10^9$, длина отрезка строго больше нуля).

Гарантируется, что для каждой пары отрезков их пересечение или пусто, или состоит из одной точки, которая является концом для обоих отрезков.

Формат выходных данных

Выведите N строк. В i-й строке выведите одно целое число — количество забавных фигур на бумаге после i-го шага.

Пример

стандартный ввод	стандартный вывод
8 5 1 8 4 5 7 2 4 10 6 10 2 2 4 5 1 5 1 5 7 8 4 5 7 8 4 10 2 10 6 8 4	1 2 3 2 2 2 0 1

Пример

Ввод	Вывод
8 5 1 8 4 5 7 2 4 10 6 10 2 2 4 5 1 5 1 5 7 8 4 5 7 8 4 10 2 10 6 8 4	1 2 3 2 2 2 0 1

Рис. 4 – Задача D, описание данных

5 Приложение B

F. Если лодку украли...

Ограничение времени	1 секунда
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

У знаменитого изобретателя Леонардо украли лодку. Но ему надо перевезти на другой берег свою библиотеку. Леонардо решил сколотить вместо лодки плавучий ящик в форме прямоугольного параллелепипеда. Быстро написав трактат «О прямоугольных лодках», он определил, что длина, ширина и высота, измеренные в сантиметрах, должны быть целыми числами, ширина лодки должна быть в точности равна 1/3 от длины, а высота должна быть в точности равна 1/2 от ширины.

Сам ящик (вне зависимости от размеров — всё же не зря Леонардо считают гением) весит 50 килограммов. Библиотека Леонардо весит W килограммов, и он хочет перевезти её за один раз. Напоминаем, что лодка с грузом плавает, если суммарный вес лодки и груза в точности равен весу воды, вытесненной лодкой. Верхняя сторона ящика и поверхность воды параллельны. Леонардо хочет, чтобы расстояние от верхней стороны ящика до ватерлинии (линии контакта с поверхностью воды) была не менее 30 сантиметров.

Напишите программу, которая найдёт минимальное значение длины, ширины и высоты ящика, который должен сделать Леонардо. Напоминаем, что 1000 кубических сантиметров воды весят 1 килограмм.

Рис. 5 – Задача F, условия

Формат входных данных

Входные данные состоят из одного целого числа W ($1 \leqslant W \leqslant 1000$) — суммарный вес библиотеки Леонардо в килограммах.

Формат выходных данных

Выведите три целых числа в одной строке — минимальную длину, ширину и высоту ящика, в соответствующем порядке.

Примеры

стандартный ввод	стандартный вывод
42	216 72 36
128	240 80 40

Пример 1

Ввод	Вывод
42	216 72 36

Пример 2

Ввод	Вывод
128	240 80 40

Рис. 6 – Задача F, описание данных

6 Приложение Г

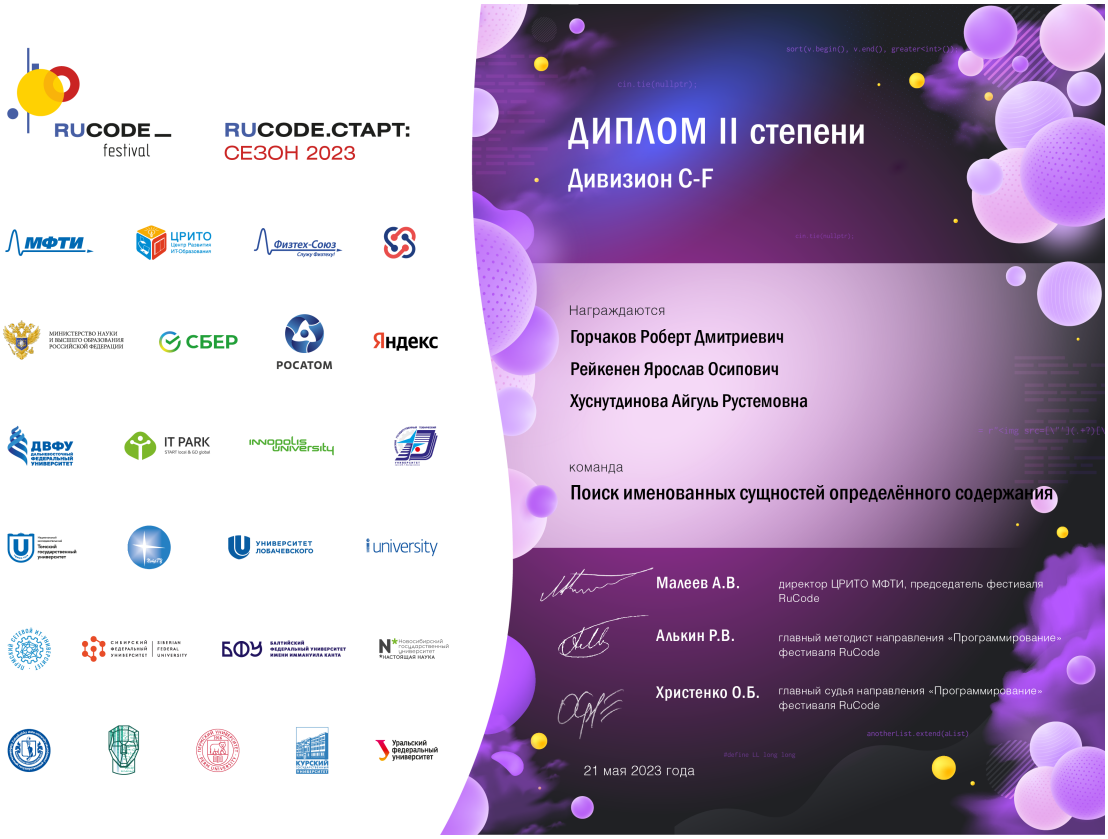

Рис. 7 – Диплом