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

Петрозаводский государственный университет

Институт математики и информационных технологий

09.03.04 - Программная инженерия

Отчет о прохождении производственной практики

Выполнил студент группы 22207:
Рейкенен Ярослав Осипович

Направление подготовки:
Системное и прикладное программное обеспечение

Место прохождения практики:
Кафедра информатики и математического обеспечения

Сроки прохождения практики:
29.05.23-09.06.23

Руководитель:
Богоявленская Ольга Юрьевна, к.т.н., доцент

Оценка _____

Дата _____

Петрозаводск — 2023

Содержание

Введение

Цель практики: развитие навыков решения олимпиадных задач в рамках чемпионата по алгоритмическому программированию RuCode.

Задачи учебной практики:

1. Пройти онлайн-интенсивы по спортивному программированию дивизиона C и изучить теоретический материал по алгоритмическому программированию;

2. Используя полученные теоретические знания, принять участие в чемпионате по спортивному программированию.

План прохождения практики:
В рамках Всероссийского учебного фестиваля по искусственному интеллекту и программированию RuCode проводились интенсивы и чемпионат по двум направлениям: искусственный интеллект и алгоритмическое программирование (далее АП). Во время прохождения практики было принято участие в направлении АП.

В течение 5 дней проводились онлайн-интенсивы по АП. В установленную дату, 21.05.2023, был проведен чемпионат по алгоритмическому программированию. Одной из площадок для его проведения был Петрозаводский государственный университет, где команды из 3-х и менее человек принимали участие. Во время прохождения практики участие в чемпионате производилось в составе команды "Поиск Именованных Сущностей Определённого Содержания"(далее Команда). Чемпионат проходил в течение 5 астрономических часов.

Команда организовала работу по достижению целей и выполнению задач, разделив рабочие обязанности:

1. Проектирование алгоритма для решения конкретной задачи;

2. Программная реализация алгоритма.

1 Интенсивы

1.1 Дивизионы алгоритмического программирования

Обучение на интенсиввах разделялось на 4 дивизиона:

1. Дивизион C;

2. Дивизион D;

3. Дивизион E;

4. Дивизион F.

Сложность алгоритмов, проходимых в рамках дивизиона, по убывающей (дивизион C — самый сложный, дивизион F — самый простой).

1.2 Прохождение интенсивов

Возможность проходить интенсивы в течение 5 дней была как дистанционная, так и очная, в стенах ПетрГУ. Был выбран очный вариант

На основании анализа своих навыков в алгоритмическом программировании был выбран дивизион C.

2 Чемпионат

2.1 Формат чемпионата

Чемпионат проводился в различных площадках городов России, одной из площадок был Петрозаводский Государственный университет, где мы приняли участие. Участвовали в чемпионате дивизиона C-F.

Чемпионат проходил 21.05.2023 10:00-15:00.

Все программы нашей команды написанны на языке C++.

2.2 Задача A. Анализ хода матча

2.2.1 Формулировка задачи

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

В спин-оффе популярного фильма Алиса играет шахматный матч с Бобом. Матч состоит из нескольких игр и начинается со счёта 0:0. Если какой-то игрок выигрывает игру, к её/его баллам прибавляется 1, в противном случае (в случае ничьей) каждый игрок получает по 0.5 балла за эту партию. Вам дан текущий счёт. Найдите общее количество

сыгранных партий, а также наименьшее и наибольшее возможное количество ничьих в матче. [1]

Формат входных данных

Первая строка входных данных содержит одно вещественное число A — количество баллов, набранных Алисой. Вторая строка содержит одно целое число B — количество баллов, набранных Бобом (0 <= A, B <= 6). Числа даны без лишних знаков (то есть без лишних нулей после десятичной точки и без самой десятичной точки в случае целых чисел). Гарантируется, что существует такая последовательность результатов партий, при которой счёт A:B будет достигнут.

Формат выходных данных

В первой строке выведите общее количество сыгранных партий. Во второй — наименьшее возможное количество ничьих в матче. В третьей — наибольшее возможное количество ничьих в матче.

Примеры

стандартный ввод	стандартный вывод
1.5 0.5	2 1 1
0 0	0 0 0

2.2.2 Идея алгоритма

Число ходов - сумма целых частей a + b и +1(если a заканчивается на .5)

Минимальное количество ничей - 0, если оба целые, иначе 1

[1] Источник задачи A и всех ниже задач: https://rucode.net/

Максимальное - b/0.5

2.2.3 Решение задачи

Фрагмент кода решения задачи A представлено ниже в листинге 1.

Листинг 1 – Задача A. Анализ хода матча

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

#define ff first
#define ss second
#define pb push_back

#ifndef LOCAL
        #define cerr if(0)cerr
#endif

using namespace std;

typedef long long ll;
typedef long double ld;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

void solve();

int main()
{
        ios_base::sync_with_stdio(0);
```

```
        cin.tie(0);
        cout.tie(0);

#ifdef FILES
        freopen("input.in", "r", stdin);
        freopen("output.out", "w", stdout);
#endif

        int t = 1;
        //cin >> t;

        while(t--) solve();

        return 0;
}

void solve()
{
        double a, b;
        cin >> a >> b;

        int ans = (int)a + (int)b;
        int n1 = 0, n2 = 0;

        if((int)a != a)
                n1++;
        ans += n1;
```

```
        n2 = min(a, b)*2;

        cout << ans << "\n" << n1 << "\n" << n2;
}
```

2.2.4 Результат решения задачи

С 1-ой попытки программа прошла все тесты.

Результат: Решена.

2.3 Задача C. Выделяем римские анаграммы

2.3.1 Формулировка задачи

Имя входного файла: стандартный ввод

Имя выходного файла: стандартный вывод

Ограничение по времени: 1 секунда

Ограничение по памяти: 512 мегабайт

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).

Формат входных данных

Первая строка входных данных содержит одно целое число T — количество тестовых примеров (1 <= T <= 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 <= |s| <= 15).

Формат выходных данных

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

Примеры

стандартный ввод	стандартный вывод
2 IC XV	1 1

2.3.2 Идея алгоритма

Решение с помощью перебора. Идея - записать все варианты в условия и перебрать всевозможные положения букв, проверить на корректность с помощью большого количества условий. Если получалась корректная римская цифра — добавлял в сумму.

2.3.3 Решение и результат задачи

Программа получила вердикт WA 4 из-за опечатки в одном из условий. Со второй попытки задача была решена.

Результат: Accept.

2.4 Задача J. Играем в угадайку?

2.4.1 Формулировка задачи

Имя входного файла: стандартный ввод

Имя выходного файла: стандартный вывод

Ограничение по времени: 1 секунда

Ограничение по памяти: 512 мегабайт

Это интерактивная задача Программа жюри загадала целое число N от 1 до 10^6 , и ваша задача — отгадать его. Для этого вы можете работать со специальным регистром X. В начале игры X = N. Вы можете задавать запросы вида ? d, где d — целое число между 0 и 10^6 . Программа жюри заменяет текущее значение X значением X + d. Если X + d > 2^20, то вы проиграли. В противном случае программа жюри отвечает 1, если X + d — квадрат целого числа, и 0 в противном случае. Вы можете задать не более 2023 запросов. Если вы считаете, что у вас хватает информации, чтобы назвать N, вы сообщаете ответ программе жюри в формате ! N. Если вы угадали, решение зачтено. Это действие не считается запросом. Гарантируется, что интерактор не является адаптивным, то есть значение N определяется перед началом взаимодействия и не меняется в процессе.

Протокол взаимодействия

Входные данные состоят из одного целого числа W (1 <= W <= 1000) — суммарный вес библиотеки Леонардо в килограммах. Взаимодействие начинает ваша программа, выводя запрос. Запрос имеет вид ? d (0 <= d <= 10^6). В ответ на это программа жюри выведет 0 или 1 в зависимости от нового значения регистра X (1, если для некоторого целого q q · q = X, и 0 в противном случае). После чего вы задаёте следующий вопрос и так далее. Если вы хотите вывести ответ, выводите его в виде ! N. Не забывайте сбрасывать буфер вывода после каждого запроса или вывода ответа, иначе ваша программа может получить вердикт Idleness Limit.

Пример

стандартный ввод	стандартный вывод
1 0 1 1	? 2 ? 8 ? 8 ? 75 ! 7

2.4.2 Идея алгоритма

Пытаемся высчитать расстояние от начальной позиции до ближайших квадратов впереди числа. Если получается найти число, расстояние от которого до следующих квадратов совпадает, то оно является ответом. Можно заметить что разница между двумя соседними квадратами это всегда нечётное число. После нахождения первого квадрата можно увеличить число на 1, а после увеличивать на 2. Так мы гарантированно найдём следующий квадрат.

2.4.3 Решение задачи

Фрагмент кода решения задачи J представлено ниже в листинге 2.

Листинг 2 – Задача J. Играем в угадайку?

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

#define ff first
```

```
#define ss second
#define pb push_back

#ifndef LOCAL
        #define cerr if(0)cerr
#endif

using namespace std;

typedef long long ll;
typedef long double ld;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

void solve();

int main()
{
#ifdef FILES
        freopen("input.in", "r", stdin);
        freopen("output.out", "w", stdout);
#endif

        int t = 1;
        //cin >> t;

        while(t--) solve();
```

```
        return 0;
}

int check(int x) {
        cout << "? " << x << endl;
        cin>> x;
        return x;
}

void solve()
{
        vector<int> tmp;

        int cur = 0;

        for (;; cur = cur + 1 + tmp.size()) {
                int x = check(1 + tmp.size());
                if (x == 1) {
                        tmp.push_back(cur);
                        if (tmp.size() == 2) break;
                        check(cur);

                        if (cur & 1) {
                                cur += cur;
                        } else {
                                cur += cur;
```

```
                                check(1); cur++;
                        }
                }
        }
        int d = tmp[1] - tmp[0];
        int ans = ((d - 1) / 2 ) * ((d - 1) / 2) - tmp[0] - 1;
        cout << "! " << ans;
}
```

2.4.4 Результат решения задачи

После исправление некоторых багов, задача была решена.

Результат: Решена.

2.5 Задача L. Крутые номера

2.5.1 Формулировка задачи

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 512 мегабайт

Автомобильные номерные знаки в Берляндии имеют следующий формат: letter letter digit digit digit letter letter digit digit Здесь letter обозначает строчную латинскую букву, а digit — десятичную цифру. Допустима любая комбинация букв и цифр в рамках этих ограничений. «Крутыми» номерами в Берляндии считаются номера, в которых и цифровая часть, и буквенная являются палиндромами. Например, номер ab123ba21 — крутой, так как и abba,

и 12321 — палиндромы, а номер aa111aa22 — нет, так как число 11122 палиндромом не является. Дорожная инспекция Байтландии засекла нарушителя на крутом автомобиле с крутым номером. Номер записался на камеру, однако есть проблема: из-за высокой скорости распознать символ на некоторых местах не получается. Соответствующие буквы (или цифры) в строке номера заменены звёздочкой (*), одна звёздочка соответствует одной цифре или букве в зависимости от номера позиции. Вам дана распознанная строка. Найдите количество возможных крутых номеров, которые могли быть на фотографии.

Формат входных данных

Входные данные состоят из одной строки длины 9. Символы на позициях 0,1,5,6 — или строчные латинские буквы, или '*'; символы на остальных позициях — или цифры, или '*'.

Формат выходных данных

Выведите одно целое число — количество крутых номеров, которые соответствуют данным с фотографии.

Примеры

стандартный ввод	стандартный вывод
a*****b**	0
*********	676000
aa777aa77	1

2.5.2 Идея алгоритма

Проверяем на корректность. Если все ок, то начинаем считать варианты: Задача была поделена на две подзадачи. В первой подсчитывалось число полиндромов из циферных символов. Во второй подзадаче подсчитывалось число полиндромов из буквенных символов.

1. Если центральный элемент не зафиксирован — умножаем ответ на 10(т.к. на этой позиции число);
2. Если парный элемент зафиксирован, а текущий символ — нет, то ответ не увеличивается(умножается на 1);
3. Если парный элемент и текущий символ не зафиксированы, умножаем ответ на 26, если должна стоять буква, и на 10, если должно быть число.

2.5.3 Решение задачи

Фрагмент кода решения задачи L представлено ниже в листинге 3.

Листинг 3 – Задача L. Крутые номера

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

#define ff first
#define ss second
#define pb push_back

#ifndef LOCAL
        #define cerr if(0)cerr
#endif

using namespace std;

typedef long long ll;
typedef long double ld;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;
```

```
void solve();

int main()
{
        ios_base::sync_with_stdio(0);
        cin.tie(0);
        cout.tie(0);

#ifdef FILES
        freopen("input.in", "r", stdin);
        freopen("output.out", "w", stdout);
#endif

        int t = 1;
        //cin >> t;

        while(t--) solve();

        return 0;
}

long long f(string s) //let
{
        long long ans = 1;
```

```
        if(s[0] == '*' && s[3] == '*') ans *= 26;
        else if ((s[0] != '*' && s[3] == '*') || (s[0] == '*' && s[3]
        else if(s[0] != '*' && s[3] != '*' && s[0] != s[3]) ans *= 0;

        if(s[1] == '*' && s[2] == '*') ans *= 26;
        else if ((s[1] != '*' && s[2] == '*') || (s[1] == '*' && s[2]
        else if(s[1] != '*' && s[2] != '*' && s[1] != s[2]) ans *= 0;

        return ans;

}

long long g(string s) //num
{
        long long ans = 1;

        if(s[0] == '*' && s[4] == '*') ans *= 10;
        else if ((s[0] != '*' && s[4] == '*') || (s[0] == '*' && s[4]
        else if(s[0] != '*' && s[4] != '*' && s[0] != s[4]) ans *= 0;

        if(s[1] == '*' && s[3] == '*') ans *= 10;
        else if ((s[1] != '*' && s[3] == '*') || (s[1] == '*' && s[3]
        else if(s[1] != '*' && s[3] != '*' && s[1] != s[3]) ans *= 0;

        if(s[2] == '*') ans *= 10;
```

```
        return ans;

}

void solve()
{
        string s;
        cin >> s;
        string a = s.substr(0, 1) + s.substr(1, 1) + s.substr(5, 1) +
        string b = s.substr(2, 1) + s.substr(3, 1) + s.substr(4, 1) +
        cout << f(a) * g(b);
}
```

2.5.4 Результат решения задачи

Со 1-ой попытки программа прошла все тесты.

Результат: Решена.

Заключение

Нашей командой были успешно решены 7 задач из 12 и занято 34 место из 227.

Решение 8-ой задачи было придумано, но не хватило времени на реализацию. По итогу был получен дипллом призёра 2 степени.

Список литературы

1. Котельников, И. А. LaTeX по-русски [Электронный ресурс]. / И. А. Котельников, П. З. Чеботаев. — 3-е изд., перераб. и доп. —

Новосибирск : Сибирский хронограф, 2004. — 496 с. — URL: https://www.researchgate.net/publication/235255954_LaTeX_po-russki

2. Вики-конспекты // Wiki URL: https://neerc.ifmo.ru/wiki/ (дата обращения: 27.11.2022)

3. Каталог алгоритмов // Codeforces URL: https://codeforces.com/catalog (дата обращения: 27.11.2022)