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:
Смирнов Евгений Русланович

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 B и изучить теоретический материал по алгоритмическому программированию;

2. Используя полученные теоритические знания, выполнить задания чемпионата.

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). Мной было принято участие во направлении АП.

В течение 5 дней проводились онлайн интенсивы по АП. В установленную дату, 21.05.2023, был проведен чемпионат по АП. Одной из площадок для его проведения был Петрозаводский государственный университет, где мы с командой из 3-х человек приняли участие. Чемпионат проходил в течение 5 астрономических часов.

Наша команда «Радикальные псевдокодеры» организовала работу по достижению целей и выполнению задач, разделив рабочие обязанности между всеми членами команды:

1. Поиск и 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 A;

2. Дивизион B;

3. Дивизион C;

4. Дивизион D;

5. Дивизион E;

6. Дивизион F.

Сложность алгоритмов, проходимых в рамках дивизиона, по убывающей (дивизион A - 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. Мной был выбран дистанционный вариант.

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

2 Чемпионат

2.1 Формат чемпионата

Чемпионат проводился в разнличных площадках городов России, одной из площадок был Петрозаводский Государственный университет, где мы приняли участие. Участвовали в чемпионате дивизиона A-B.

Чемпионат проходил 21.05.2023 10:00-15:00.

Все программы нашей команды написанны на языке C++.

2.2 Задача C. Выделяем римские анаграммы

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

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

2.2.2 Идея алгоритма

Идея - перебрать все числа от 1 до 3999 в римской записи, отсортировать все буквы в каждой записи в лексикографическом порядке и записать количество вхождений каждой отсортированной записи в таблицу. В качестве ответа на каждый запрос выводить количество вхождений данного запроса в таблицу.

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>
using namespace std;

#define ll long long
#define pll pair <ll, ll>

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

```
map <vector <ll>, ll> ans;
map <char, ll> s = {
    {'I', 0},
    {'V', 1},
    {'X', 2},
    {'L', 3},
    {'C', 4},
    {'D', 5},
    {'M', 6},
};
map <ll, vector <ll>> ss = {
    {1, {1, 0, 0}},
    {2, {2, 0, 0}},
    {3, {3, 0, 0}},
    {4, {1, 1, 0}},
    {5, {0, 1, 0}},
    {6, {1, 1, 0}},
    {7, {2, 1, 0}},
    {8, {3, 1, 0}},
    {9, {1, 0, 1}},
};

for (ll i = 1; i < 4000; i++) {
    vector <ll> k(7);
    ll n = i;
    ll d = 0;
```

```
    while (n) {
        ll c = n % 10;
        n /= 10;
        if (c == 0) {
            d += 2;
            continue;
        }
        for (ll j = 0; j < ss[c].size() && d + j < 7; j++) {
            k[j + d] += ss[c][j];
        }
        d += 2;
    }

    ans[k]++;
}

ll t; cin >> t;
while (t--) {
    vector <ll> k(7);

    string str; cin >> str;
    for (auto c : str) {
        k[s[c]]++;
    }

    cout << ans[k] << "\n";
}
```

```
}
```

2.2.4 Решение и результат задачи

Программа прошла все тесты.

Результат: Решена.

2.3 Задача E. Делаем XORтировку

2.3.1 Формулировка задачи

Имя входного файла: стандартный ввод

Имя выходного файла: стандартный вывод

Ограничение по времени: 2 секунды

Ограничение по памяти: 512 мегабайт

Байтика придумала новый способ сортировки целых чисел, который она назвала eXperimental sOrting, или XORтировка. Пусть есть последовательность $a_1, a_2, ..., a_N$, тогда Байтика ищет такое целое число X, что $a1 \oplus X <= a2 \oplus X <= ... <= aN \oplus X$, где $\oplus$ обозначает побитовое исключающее ИЛИ, то есть всё, что ей надо — заменить a_i на $a_i \oplus x$, и массив отсортирован. Ваша задача — выполнять запросы на XORтировку. Изначально вам дан массив из целых неотрицательных чисел $a1, ..., aN$; вам надо ответить на Q запросов вида p_i, v_i, которые обозначают, что элемент a_{pi} становится равным v_i в результате i-го запроса. Ваша задача —- найти $Q + 1$ целое число $c_0, c_1, ..., c_Q, c_i$ — наименьшее неотрицательное целое число X такое, что массив $a_1 \oplus X, ..., a_N \oplus X$ (со значениями элементов после первых i запросов) будет отсортирован. Если такого X нет, выведите вместо соответствующего значения X число -1.

Формат входных данных

Первая строка входных данных содержит одно целое число $N(1 <= N <= 10^6)$ — количество элементов в массиве a. Вторая строка содержит N целых чисел $a_1, a_2, ..., a_N(0 <= a_i < 2^{30})$ — начальные значения элементов массива. Третья строка содержит целое число $Q(0 <= Q <= 10^6)$ — количество запросов. Каждая из последующих Q строк содержит по два числа $p_i(1 <= p_i <= N)v_i(0 <= v_i < 2^{30})$, задающих запрос на присваивание элементу a_{pi} значения v_i.

Формат выходных данных

Выведите Q + 1 строку, i-я из которых содержит одно целое число c_{i-1}. c_i должно соответствовать наименьшему целому положительному X, которое XORтирует массив после того, как изменения 1, 2, . . . , i были применены (или -1, если такого X нет).

Примеры

стандартный ввод	стандартный вывод
3 0 1 4 3 2 7 3 3 1 4	0 2 -1 4

2.3.2 Идея алгоритма

Заметим, что число будет больше предыдущего в том случае, если самый первый из различающихся битов будет больше. Таким образом, если предущее число имеет старший бит, то в X этот бит должен быть равен 1, иначе 0. Если в какой-то момент находится противоречие (нужный бит уже уста-

новлен и должен быть изменён), то ответ X. Также можно заметить, что при изменении числа следует проверять изменение битов только двух соседних от числа чисел, чтобы не пересчитывать всё заного.

2.3.3 Решение задачи

Фрагмент кода решения задачи E представлено ниже в листинге 2.

Листинг 2 – Задача E. XORтировка

```
#include <bits/stdc++.h>
using namespace std;

#define pll pair <int, int>
#define double long double

int arr[33][2];
int a[1000006];

void add(int a, int b)
{
    for(int i = 30; i >= 0; i--)
        if(((a & (1 << i)) >> i) != ((b & (1 << i)) >> i))
        {
            if(((a & (1 << i)) >> i) > ((b & (1 << i)) >> i))
                arr[i][1] ++;
            else arr[i][0] ++;
            break;
        }
}
```

```
void del(int a, int b)
{
    for(int i = 30; i >= 0; i--)
        if(((a & (1 << i)) >> i) != ((b & (1 << i)) >> i))
        {
            if(((a & (1 << i)) >> i) > ((b & (1 << i)) >> i))
                arr[i][1] --;
            else arr[i][0] --;
            break;
        }
}

int get()
{
    int x = 0;
    for(int i = 30; i >= 0; i--)
    {
        if(arr[i][0] != 0 && arr[i][1] != 0) return -1;
        else if(arr[i][1] != 0) x |= (1 << i);
    }
    return x;
}

void solve() {

    int n;
    cin >> n;
```

```
    for(int i = 0; i < n; i++)
    {
        cin >> a[i];
        if(i) add(a[i - 1], a[i]);
    }
    cout << get() << '\n';
    int q;
    cin >> q;
    while(q--)
    {
        int p, v;
        cin >> p >> v;
        int i = p - 1;
        if(p != 1) del(a[i - 1], a[i]);
        if(p != n) del(a[i], a[i + 1]);
        a[p - 1] = v;
        if(p != 1) add(a[i - 1], a[i]);
        if(p != n) add(a[i], a[i + 1]);
        cout << get() << '\n';
    }
}

signed main(){

    cout << fixed << setprecision(10);

    // freopen("input.txt", "r", stdin);
```

```
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    int t = 1;
    // cin >> t;
    for (int i = 1; i <= t; i++){
        solve();
        cout << '\n';
    }

}
```

2.3.4 Результат решения задачи

Программа прошла все тесты.

Результат: Решена.

2.4 Задача H. «Железные» секреты

2.4.1 Формулировка задачи

Имя входного файла: стандартный ввод

Имя выходного файла: стандартный вывод

Ограничение по времени: 4 секунды

Ограничение по памяти: 512 мегабайт

Компания Thinkbox планирует построить новую фабрику в джунглях, ко-

торая будет выпускать новый продукт компании — беспроводные клавиатуры с миниджойстиком, подсветкой клавиатуры и полугодовым временем работы между зарядками. Успех этого продукта гарантирует компании Thinkbox длительное доминирование на рынке. Чтобы избежать утечек информации к конкурентам, фабрика будет расположена под лесом и будет иметь форму квадрата. Лес на данной территории может быть представлен как матрица $R \times C$, для каждой клетки которой определён вид деревьев, доминирующих в данной клетке. Виды деревьев заданы строчными латинскими буквами. Территория фабрики должна быть квадратом $F \times F$, где $1 <= F <= min(R, C)$, стороны которого проходят точно по границам между клетками леса. Подлесное расположение фабрики обозначает, что фабрика должна быть максимально безопасно расположена. Сейчас конкуренты могут легко сделать фотографию участка леса, под которым расположена компания, используя дроны. Поэтому CEO компании хочет выбрать конфигурацию типов деревьев над фабрикой таким образом, что в лесу будет как минимум K квадратов с точно такой же стороной и одинаковым расположением доминирующих деревьев (иначе говоря, для каждого из K квадратов существует параллельный перенос, который при совмещении этих квадратов даёт полное совпадение всех обозначающих типы букв). Квадраты при этом могут перекрываться. Подлесное расположение фабрики обозначает также, что фабрика должна быть построена «на халяву», но если такое невозможно — то с минимумом дополнительных затрат. Налог на постройку фабрики не зависит от длины стороны квадрата, то есть большая длина F обозначает меньшие относительные затраты на квадратный метр. Вам дана карта леса и значение коэффициента секретности K. Найдите максимальное значение стороны квадрата F для постройки фабрики, или -1, если требование секретности выполнить не удастся.

Формат входных данных

Первая строка входных данных содержит три целых числа R, C и K ($1 <= R, C <= 600, 2 <= K <= R \cdot C$) — количество строк и столбцов в матрице, задающей лес, и коэффициент секретности (число одинаково выглядящих сверху квадратов). Далее следуют R строк, каждая из которых содержит C строчных латинских букв — карту леса. Клетки с одним и тем же доминирующим типом растительности обозначены одинаковыми буквами, клетки с разыми доминирующими типами— разными.

Формат выходных данных

Выведите одно целое число — максимальную длину квадрата F. Если требование секретности выполнить не удастся и K одинаково выглядящих квадратов не существует, выведите -1.

Примеры

стандартный ввод	стандартный вывод
4 5 2 baacd bbaab bbbaa bbbbb	3
3 3 5 cbb dxb xxx	-1

2.4.2 Идея алгоритма

Для быстрого определения совпадения квадратов можно воспользоваться хешированием входной матрицы по строкам и столбцам. Тогда можно будет

за линейное время переходить от одного положения к соседнему.

2.4.3 Решение задачи

Фрагмент кода решения задачи H представлено ниже в листинге 3.

Листинг 3 – Задача H. Железные секреты

```
#include <bits/stdc++.h>
using namespace std;

#define int long long
#define double long double

#define htype pair<int, int>

htype operator + (htype a, htype b){
    return {a.first + b.first, a.second + b.second};
}

htype operator - (htype a, htype b){
    return {a.first - b.first, a.second - b.second};
}

htype operator * (htype a, htype b){
    return {a.first * b.first, a.second * b.second};
}

htype operator % (htype a, htype b){
    return {a.first % b.first, a.second % b.second};
```

```
}

const int SZ = 1 << 10;

htype mod = {1e9 + 7, 1e9 + 9};
htype base = {29, 31};
htype h[SZ][SZ];
htype pw[SZ * SZ];
htype rw[SZ * SZ];

int f_pow(int a, int p, int mod){
    if (p == 0){
        return 1;
    }
    if (p & 1){
        return a * f_pow(a, p - 1, mod) % mod;
    }
    return f_pow(a * a % mod, p >> 1, mod);
}

char a[SZ][SZ];
int n, m, k;

htype get(int i, int l, int r){
    return (h[i][r] - h[i][l - 1] + mod) * pw[SZ - r] % mod;
}
```

```
bool f(int x){
    unordered_map<int, int> mp;
    for (int j = 1; j <= m - x + 1; j++){
        htype cur = {0, 0};
        for (int i = 1; i <= x; i++){
            cur = (cur + get(i, j, j + x - 1) * pw[(i - 1) * x]) % mod
            // cout << ((get(i, j, j + x - 1)) % mod).first << ' ';
        }
        mp[cur.first + (cur.second << 32)]++;
        // cout << cur.second << ' ';
        for (int i = x + 1; i <= n; i++){
            cur = (cur - get(i - x, j, j + x - 1) + mod) % mod;
            cur = (cur * rw[x]) % mod;
            // cout << cur.first << ' ';
            cur = (cur + get(i, j, j + x - 1) * pw[x * (x - 1)]) % mod
            mp[cur.first + (cur.second << 32)]++;
            // cout << cur.second << ' ';
        }
        // cout << endl;
    }

    for (auto e : mp){
        if (e.second >= k){
            return true;
        }
    }
```

```
    return false;
}

void solve() {

    pw[0] = {1, 1};
    rw[0] = {1, 1};
    for (int i = 1; i < SZ * SZ; i++){
        pw[i] = pw[i - 1] * base % mod;
        rw[i].first = f_pow(pw[i].first, mod.first - 2, mod.first);
        rw[i].second = f_pow(pw[i].second, mod.second - 2, mod.second)
        // rw[0] = f_pow(pw[i], mod - 2, mod);
    }

    cin >> n >> m >> k;
    for (int i = 1; i <= n; i++){
        for (int j = 1; j <= m; j++){
            cin >> a[i][j];
            h[i][j] = (h[i][j - 1] + make_pair(a[i][j] - 'a' + 1, a[i]
            // cout << get(i, j, j).first << ' ';
        }
        // cout << endl;
        // cout << h[i][1].first << ' ' << get(i, 1, 1).first << endl;
    }

    int l = 1;
    int r = min(n, m) + 1;
```

```
    if (!f(1)){
        cout << -1;
        return;
    }

    while (l < r){
        int mid = (l + r) >> 1;
        if (f(mid)){
            l = mid + 1;
        }else{
            r = mid;
        }
    }

    cout << l - 1;

}

signed main(){

    cout << fixed << setprecision(10);

   // freopen("input.txt", "r", stdin);

    ios_base::sync_with_stdio(0);
    cin.tie(0);
```

```
    cout.tie(0);

    int t = 1;
    // cin >> t;
    for (int i = 1; i <= t; i++){
        solve();
        cout << '\n';
    }

}
```

2.4.4 Результат решения задачи

В рамках контеста задача решена не была, но была впоследствии дорешена вне соревнования.

Результат: Не решена.

2.5 Задача J. Играем в угадайку?

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

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 $\cdot$ 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

2.5.2 Идея алгоритма

пытаемся найти ближайшие два квадрата, следующие после этого числа. А затем, найдя разницу между этими двумя квадратами, находим исходное число.

2.5.3 Решение задачи

Фрагмент кода решения задачи J представлено ниже в листинге 4.

Листинг 4 – Задача J. Играем в угадайку?

```
#include <bits/stdc++.h>
using namespace std;

#define int long long
#define double long double

bool q(int x){
```

```
    cout << "? " << x << endl;
    int ans;
    cin >> ans;
    return ans;
}

void pr(int x){
    cout << "! " << x << endl;
    exit(0);
}

void solve() {

    int d = 0;
    if (!q(0)){
        d++;
        while (!q(1)){
            d++;
        }
    }

    int buf = d;

    d++;
    if (!q(d + 1)){
        while (!q(1)){
            d++;
```

```
        }
        d++;
    }

    int x = d / 2 + 1;
    pr((x-1)*(x-1) - buf);

}

signed main(){

    cout << fixed << setprecision(10);

    // freopen("input.txt", "r", stdin);

    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    int t = 1;
    // cin >> t;
    for (int i = 1; i <= t; i++){
        solve();
        cout << '\n';
    }

}
```

2.5.4 Результат решения задачи

Программа прошла все тесты.

Результат: Решена.

Заключение

Нашей командой были успешно решены 4 задачи из 12 и занято 9 место из 49.

В ходе практики мной были приобретены знания алгоритмов и опыт работы с языком C++, а также опыт работы в команде.

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)