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

# Решение олимпиадных задач Дивизиона A/B на чемпионате Rucode

Выполнил:
студент группы 22307

Анисимов Н.Н. _____
*подпись*

Место прохождения практики:
Кафедра информатики и математического обеспечения

Период прохождения практики:
29.05.23-11.06.23

Руководитель:
О.Ю. Богоявленская, к.т.н., доцент

_____
*подпись*

Итоговая оценка

_____
*оценка*

Петрозаводск — 2023

# Содержание

# Введение

Цель практики: Участие в фестивале RuCode для получения и применения теоретических знаний в области алгоритмического программирования.

Задачи учебной практики:

1. Тренировка навыков олимпиадного программирования;
2. Изучение алгоритмов и подходов в олимпиадном программировании;
3. Получение диплома в чемпионате.

План прохождения практики:
В рамках Всероссийский учебный фестиваль по искусственному интеллекту и программированию RuCode проводились интенсивы и чемпионат по двум площадкам: искусственный интеллект и алгоритмическое программирование (далее АП). Мной было принято участие во второй площадке, АП.

В течение 5 дней проводились интенсивы по АП. В установленную дату был проведен чемпионат по АП в разных точках страны. Одной из точек был Петрозаводский государственный университет, в стенах которого мы с командой из 3-х человек и приняли участие. Чемпионат проходил в течение 5 часов.

Наша команда «Радикальные псевдокодеры» организовала работу по достижению целей и выполнению задач.

# 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

Я выбрал дивизион C, как наиболее мне подходящий. Возможность проходить интенсивы в течение 5 дней была как дистанционная, так и очная, в аудиториях ПетрГУ. Мной был выбран очный вариант.

# 2 Чемпионат

## 2.1 Формат чемпионата

Чемпионат проводился в разных точках России, одной из точек был Петрозаводский Государственный университет, где мы приняли участие. Проанализировав наши зная и опыт было принято решения участвовать в более сложном дивизионе, а именно в дивизионе A\B.

Чемпионат проходил 21.05.2023 10:00-15:00.

Ниже представлен список задач, решения к которым были разработаны мной.

## 2.2 Задача D. Графопостроитель

### 2.2.1 Формулировка задачи

Input file: standard input

Output file: standard output

Time limit: 2 seconds

Memory limit: 512 mebibytes

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.

### Input

Первая строка входных данных содержит одно целое число N — количество шагов графопостроителя ($1 \le N \le 2 \cdot 10^5$). 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 ($0 \le x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2} \le 10^9$ , 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.

**Output**

Выведите одно целое число - количество побед в сезоне с r гонками, требуемое для того, чтобы побить рекорд Аскари.

### 2.2.2 Идея алгоритма

Задача является усложненной версией задачи о динамической связности. Подобная задача решается с помощью системы непересекающихся множеств (СНМ). Для этой задачи необходимо еще поддерживать количество забавных фигур. Заметим, что фигура является забавной тогда и только тогда, когда в ней существует Эйлеров путь или цикл. В СНМ при каждом обновлении будем поддерживать степени вершин и, исходя из четности, пересчитывать количество забавных фигур.

### 2.2.3 Решение задачи

Последнее решение задачи D представлено ниже в листинге 1.

Листинг 1 – Задача D. Графопостроитель

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

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

int ans = 0;

struct obj
{
```

```
    size_t operator()(const pll a) const
    {
        return a.first * 1e9 + a.second;
    }
};

unordered_map<pll, int, obj> cnt, arr;
unordered_map<pll, pll, obj> mp;

pll get(pll a)
{
    if(mp[a] == a) return a;
    return (mp[a] = get(mp[a]));
}

void uni(pll a, pll b)
{
    int tmp = 0;
    if(cnt[a] % 2 == 0 && cnt[b] % 2 == 0)
    {
        tmp = arr[get(a)] + arr[get(b)] + 2;
    }
    else if(cnt[a] % 2 != 0 && cnt[b] % 2 != 0)
    {
        tmp = arr[get(a)] + arr[get(b)] - 2;
    }
    else{
```

```
        tmp = arr[get(a)] + arr[get(b)];
    }
    if((arr[get(a)] == 2 || arr[get(a)] == 0) && cnt[a] != 0) ans--;
    if((arr[get(b)] == 2 || arr[get(b)] == 0) && cnt[b] != 0) ans--;
    if(tmp == 2 || tmp == 0) ans++;
    a = get(a);
    b = get(b);
    if(rand() % 2 == 0)
    {
        mp[a] = b;
        arr[b] = tmp;
    }
    else
    {
        mp[b] = a;
        arr[a] = tmp;
    }
}
void solve() {

    int n;
    cin >> n;
    for(int i = 0; i < n; i++)
    {
        int a, b, c, d;
        cin >> a >> b >> c >> d;
        if(!mp.count({a, b}))
```

```
{
    mp[{a, b}] = {a, b};
}
if(!mp.count({c, d}))
{
    mp[{c, d}] = {c, d};
}
if(get({a, b}) != get({c, d}))
    uni({a, b}, {c, d});
else
{
    int tmp = 0;
    if(cnt[{a, b}] % 2 == 0 && cnt[{c, d}] % 2 == 0)
    {
        tmp = arr[get({a, b})] + 2;
    }
    else if(cnt[{a, b}] % 2 != 0 && cnt[{c, d}] % 2 != 0)
    {
        tmp = arr[get({a, b})] - 2;
    }
    else{
        tmp = arr[get({a, b})];
    }
    if((arr[get({a, b})] == 2 || arr[get({a, b})] == 0) && cnt
    if(tmp == 2 || tmp == 0) ans++;
    arr[get({a, b})] = tmp;
}
```

```
        cnt[{a, b}]++;
        cnt[{c, d}]++;
        cout << ans << endl;
    }

}

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';
    }

}
```

### 2.2.4 Результат решения задачи

Со 3-ей попытки программа прошла все тесты.

**Результат:** Success.

## 2.3 Задача E. Делаем XORтировку

### 2.3.1 Формулировка задачи

Input file: standard input

Output file: standard output

Time limit: 2 seconds

Memory limit: 512 mebibytes

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 $a_1 \oplus X \leq a_2 \oplus X \leq ... \leq a_N \oplus X$, 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 $a_1, ..., a_N$; вам надо ответить на Q запросов вида $p_i$, $v_i$, которые обозначают, что элемент $a_{p_i}$ становится равным $v_i$ в результате i-го запроса. Ваша задача —- найти Q + 1 целое число $c_0, c_1, ..., c_Q$, где $c_i$ — 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.

**Input**

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

**Output**

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).

### 2.3.2 Идея алгоритма

Как и во многих задачах на xor в этой задаче необходимо рассмотреть биты числа незавесимо. Рассмотрим какой то бит. Если это первый отличающийся бит у двух соседних чисел, то для правильной сортировки необходимо, чтобы у раньшестоящего был ноль, а у следующего единица. В таком случае в числе X должен стоять ноль в этом бите. Если же ситуация обратная, то единица. Таким образом можно вычислить X, или понять, что такого не существует (при противоречии). При дальнейших изменениях заметим, что запрос влияет только на отношения числа с двумя соседними. Будем поддерживавать массив arr[i][b], где будет хранится количество раз, когда нужно в числе X в бите i поставить значение b.

### 2.3.3 Решение задачи

Последнее решение задачи B представлено ниже в листинге 2.

Листинг 2 – Задача E. Делаем XORтировку

```
#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

С 2-ой попытки программа прошла все тесты.

**Результат:** Success.

# 3 Дневник практики

| Период | Выполняемая работа |
|---|---|
| 15.05 - 19.05 | Интенсивы RuCode |
| 21.05 | Чемпионат |
| 20.05 - 11.06 | Подготовка отчета о результатах прохождения практики. Представление их руководителю. |

# Заключение

В Чемпионате нашей командой были успешно решены 4 задачи из 12 и занято 9 место из 49. В ходе практики нашей командой был получен диплом призера третьей степени. В ходе интенсивов был изучен и отработан новый материал по алгоритмическому программированию.

## 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)

# Приложение А. Диплом призера третей степени