Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

по «Алгоритмам и структурам данных»

Базовые задачи / Timus

Выполнил:

Студент группы P32121

Гиниятуллин Арслан Рафаилович

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2023

Задачи с **Яндекс Контеста**, handle = **giniiatullinarslan**

Задача A: Агроном-любитель

```
void solve() {
    int n;
    cin >> n;
    int eq_in_a_row = 1, cur = -1;
    int flower, l = 0, l_mn = 0, r_mx = n - 1, mx = 1, i;
    for (i = 0; i < n; ++i) {
        cin >> flower;
        if (flower == cur) {
            eq_in_a_row++;
        } else {
            cur = flower;
            eq_in_a_row = 1;
        }
        if (eq_in_a_row == 3) {
            if (i - l > mx) {
                mx = i - l, l_mn = l, r_mx = i - 1;
            }
            l = i - 1;
            cur = flower;
            eq_in_a_row = 2;
        }
    }
    if (i - l > mx) {
        l_mn = l, r_mx = i - 1;
    }
    cout << l_mn + 1 << ' ' << r_mx + 1 << '\n';
}
```

Описание решения:

Давайте пройдемся по массиву, считая кол-во подряд идущих одинаковых цветков, обновляя текущий цветок. В процессе прохода будем проверять на нужное нам условие: есть ли три одинаковых цвета, идущих подряд. Если они встретились, то пересчитываем максимум и границы, тем самым поддерживая инвариант, что l_mn и r_mx – два указателя на участок с самой большой длиной без 3 идущих подряд одинаковых цветков (берем жадно границу с двумя подряд идущими одинаковыми цветками, если можно)

Асимптотика: O(n), просто цикл по массиву длины n с простыми операциями (присваивание, сравнение и тп)

Задача B: Зоопарк Глеба

```
void solve() {
    string s;
    cin >> s;
    stack<char> st;
    for (auto &i: s) {
        if (!st.empty() && tolower(i) == tolower(st.top())) {
            st.pop();
        } else {
            st.push(i);
        }
    }
    if (!st.empty()) {
        cout << "Impossible\n";
        return;
    }
    int lower_cnt = 1;
    unordered_map<char, stack<int>> mp;
    for (auto &i: s) {
        if (islower(i)) {
            mp[i].push(lower_cnt);
            ++lower_cnt;
        }
    }
    cout << "Possible\n";
    for (auto &i: s) {
        if (isupper(i)) {
            cout << mp[tolower(i)].top() << ' ';
            mp[tolower(i)].pop();
        }

    }
}
```

Описание решения:

Заметим, что перед нами задача о правильной скобочной последовательности, где буквы – это скобки. Достаточно проверить, что последовательность правильная, складывая все буквы в стек, проверяя на соответствие (первая буква ‘X’ – открывающаяся скобка, вторая – закрывающаяся). Если после прохода стек пуст, то ищем решение: находим для каждого животного по обходу – его ловушку.

Асимптотика: O(n), проверка на псп – линия, get/set в unordered_map – амортизационно O(1) – получим амортизационно O(n) в цикле

Задача C: Конфигурационный файл

```
inline bool is_open_bracket(string &c) {
    return c == "{";
}

inline bool is_close_bracket(string &c) {
    return c == "}";
}

void parse_key_value_from_str(string &s, string &key, string &value) {
    bool was_eq_sign = false;
    for (char i: s) {
        if (i == '=') {
            was_eq_sign = true;
            continue;
        }
        if (was_eq_sign) { value += i; }
        else { key += i; }
    }
}

bool is_number(const string &str) {
    for (int i = 0; i < str.size(); ++i) {
        if (str[i] == '-' && i == 0) continue;
        if (isdigit(str[i]) == 0) return false;
    }
    return true;
}

void solve() {
    string s;
    unordered_map<string, vector<int>> mpst(1);
    vector<unordered_set<string>> re(1);
    while (cin >> s) {
        if (is_open_bracket(s)) {
            re.emplace_back();
        } else if (is_close_bracket(s)) {
            for (auto &i: re.back()) {
                mpst[i].pop_back();
            }
            re.pop_back();
        } else {
            string key, value;
            parse_key_value_from_str(s, key, value);
            if (is_number(value)) {
                int val = stoi(value);
                if (re.back().count(key)) {
                    mpst[key].back() = val;
                } else {
                    mpst[key].push_back(val);
                }
            } else {
                if (re.back().count(key)) {
                    mpst[key].back() = mpst[value].empty() ? 0 :
mpst[value].back();
                } else {
                    mpst[key].push_back(mpst[value].empty() ? 0 :
mpst[value].back());
                }
                cout << mpst[key].back() << '\n';
            }
```

```
            re.back().insert(key);
        }
    }
}
```

Описание решения:

Давайте будем хранить состояние для каждого блока в unordered_map <string, vector<int>> (get/set ~ O(1)). Соответственно для каждой переменной (ключа в мапе) – пишем значение в vector по ключу. Для актуальности мапа – будем восстанавливать прежние состояния благодаря vector<unordered_set<string>>. В каждом сете будем хранить переменные, которые изменяются в scope текущего блока.

Асимптотика: ~O(n*s), где s – длина строки, а n – их количество. Работа с unordered структурами ведется за ~ O(1).

Задача D: Профессор Хаос

```
void solve() {
    int a, b, c, d;
    long long k;
    cin >> a >> b >> c >> d >> k;
    vector<int> cycle = {a};
    long long q = k;
    int res = a;
    while (q--) {
        res *= b;
        if (res <= c) {
            cout << "0\n";
            return;
        }
        res -= c;
        if (res >= d) {
            cout << d << '\n';
            return;
        }
        if (res == a) {
            cout << cycle[k % (cycle.size() + 1)] << '\n';
            return;
        }
        cycle.push_back(res);
    }
    cout << res << '\n';
}
```

Описание решения:

Заметим, что вскоре количество бактерий на начало дня будет повторяться: a = min(max(a * b – c, 0), d). a всегда не больше d.

Тогда будем хранить значение количества бактерий на начало каждого дня, пока не найдем цикл (не найдется день, в котором количество бактерий такой, что встречалось ранее). Узнав период – с легкостью находим ответ.

Асимптотика: O(D)