Факультет программной инженерии и компьютерной техники

Алгоритмы и структуры данныхка

Yandex Contest

Введение в алгоритмы

Преподаватель: Косяков Михаил Сергеевич
Выполнил: Ле Чонг Дат
Группа: P3231

2021 г.

Часть I

A. Агроном-любитель

1 Решение

Вызов mx_l и mx_r - это две конечные точки самой длинной подпоследовательности.

Изначально $mx\_l = mx\_r = 1$, принимая позицию i, если a_i, a_{i-1}, a_{i-2} не являются 3 одинаковыми числами, мы обновим $mx\_r = i$ и самая длинная подпоследовательность. Если $i \geq, 3$ и $a_i == a_{i-1} == a_{i-2}$, очевидно, что mx_l должен $\geq i-1$, поэтому мы обновим $mx\_l = i-1$ и $mx\_r = i$, а затем рассмотрим $i+1$.

2 Реализация

```
#include <bits/stdc++.h>
using namespace std;
const int N = 200010;
int n;
int a[N];
void opt(int i, int cur, int &mx, int &mx_l, int &mx_r) {
    if (cur > mx) {
        mx = cur;
        mx_r = i;
        mx_l = i - cur + 1;
    }
}
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    cin >> n;
    for(int i = 1; i <= n; ++i)
        cin >> a[i];
    int mx_l = 1, mx_r = 1, mx = 1, cur = 0;
    for(int i = 1; i <= n; ++i) {
        if (a[i] != a[i-1] or (i > 1 && a[i] != a[i-2])) {
            cur ++;
            opt(i, cur, mx, mx_l, mx_r);
            continue;
        }
        cur = 2;
        opt(i, 2, mx, mx_l, mx_r);
    }
    cout << mx_l << ' ' << mx_r;

}
```

Часть II

B. Зоопарк Глеба

1 Решение

Чтобы не перекрывать каждое (животное, ловушка) должна состоять из двух букв, стоящих рядом друг с другом (первая и последняя буквы также являются двумя соседними буквами).

Алгоритм упрощен из приведенного выше наблюдения. Мы будем использовать двухстороннюю очередь для удаления букв соответственно, вызывая pop_front(), pop_back(), если есть 2 элемента в начале двухсторонней очереди или в конце двухсторонней очереди, или 1 элемент в начале и 1 элемент в конце двухсторонней очереди. можно удалить. Ответ «невозможно», если конец двухсторонней очереди не пуст.

2 Реализация

```
#include <bits/stdc++.h>
using namespace std;
string s;
vector<pair<int, int>> pairs;
bool is_lowercase(char c) {
    return 'a' <= c && c <= 'z';
}
bool is_uppercase(char c) {
    return 'A' <= c && c <= 'Z';
}
char to_lowercase(char c) {
    return char(c - 'A' + 'a');
}
bool catchable(pair<char, int> a, pair<char, int> b) {
    if (is_lowercase(a.first) and is_lowercase(b.first)) return false;
    if (is_uppercase(a.first) and is_uppercase(b.first)) return false;
    if (is_lowercase(a.first))
        swap(a, b);
    if (to_lowercase(a.first) == b.first) {
        pairs.emplace_back(a.second, b.second);
        return true;
    }
    return false;
}
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    cin >> s;
    deque<pair<char, int>> dq;
```

```
    int trap = 0, animal = 0;
    for(int i = 0; i < s.size(); ++i) {
        if (is_uppercase(s[i])) {
            dq.emplace_back(s[i], ++trap);
            continue;
        }
        dq.emplace_back(s[i], ++animal);
    }
    while (!dq.empty()) {
        if (catchable(dq[0], dq[1])) {
            dq.pop_front();
            dq.pop_front();
            continue;
        }
        int n = dq.size();
        if (catchable(dq[n-2], dq[n-1])) {
            dq.pop_back();
            dq.pop_back();
            continue;
        }
        if (catchable(dq[0], dq[n-1])) {
            dq.pop_front();
            dq.pop_back();
            continue;
        }
        break;
    }
    if (not dq.empty()) cout << "Impossible";
    else {
        cout << "Possible\n";
        sort(pairs.begin(), pairs.end());
        for(auto pair: pairs) cout << pair.second << ' ';
    }
}
```

Часть III

C. Конфигурационный файл

1 Решение

В этой задаче мы воспользуемся рекурсией. Вызовите свою рекурсивную функцию $process\_block(vars)$, где $vars_i$ - стек, содержащий значения переменной i (извне в текущий блок). Прочтите каждую строку в этом блоке одну за другой.

- Если встречается 1 символ "{ что означает открытие другого вложенного блока, мы вызываем функцию $process\_block(vars)$ для обработки вложенного блока.

- Если мы встречаем символ «}», обозначающий конец текущего блока, мы pop все новые элементы, добавленные в стек в текущем блоке, и возвращаемся.
- В случае присвоения <v1> = <num> мы проверим, содержит ли $vars[v1]$ какое-либо значение, в противном случае $v1$ - это новая переменная, и нам нужно добавить 0 в stack $vars[v1]$, иначе нам просто нужно изменить верхнее значение стека $vars[v1]$.
- В случае присвоения <v1> = <v2> мы сначала проверим значение <v2>, как указано выше. Как только у нас есть значение <v2>, нам просто нужно присвоить <v1> = <num> значение <num> = <v2>.

2 Реализация

```
#include <bits/stdc++.h>
using namespace std;
const string delimiter = "=";
bool is_num(string &s) {
    for(auto x: s)
        if (x != '-' and (x < '0' or x > '9')) return false;
    return true;
}
long long to_num(string &s) {
    long long ans = 0;
    for(auto x: (s[0] == '-' ? s.substr(1, s.size()) : s)) ans = ans * 10 + x - '0';
    return (s[0] == '-' ? -ans : ans);
}
void process_block(map<string, stack<long long>> &vars) {
    string s;
        vector<string> popout;
    while (getline(cin, s)) {
            if (s == "}") {
                for(auto var: popout) vars[var].pop();
                return;
        }
        if (s == "{") {
            process_block(vars);
            continue;
        }

        auto var1 = s.substr(0, s.find(delimiter));
        auto var2 = s.substr(s.find(delimiter) + 1, s.size());
        if (is_num(var2)) {
            vars[var1].emplace(to_num(var2));
                popout.emplace_back(var1);
            continue;
        }
            if (not vars.count(var2) or vars[var2].empty()) {
                    vars[var2].emplace(0);
```

```
                popout.emplace_back(var2);
        }
        vars[var1].emplace(vars[var2].top());
            popout.emplace_back(var1);
        cout << vars[var1].top() << '\n';
    }
}
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    map<string, stack<long long>> vars;
    process_block(vars);
}
```

Часть IV
D.Профессор Хаос

1 Решение

- Через несколько дней, если количество вируса <= 0, завершите эксперимент, ответ будет 0.
- Через несколько дней, если количество вирусов> = d, то в оставшиеся дни количество вирусов по-прежнему будет $geqd$, тогда ответ будет d.
- В случае $b = c$ ответ будет a (поскольку вирусная нагрузка не меняется).
- Если нет, это означает, что ответ будет в диапазоне 0 $rightarrow$ 1000 ($0\ leqd\ leq1000$), поэтому даже если $k\ leq10^{18}$, но мы обязательно найдем результаты в течение $leq1000$ дней.

Алгоритм просто проверяет ежедневно и проверяет количество вирусов каждый день, чтобы найти ответ.

2 Реализация

```
#include <bits/stdc++.h>
using namespace std;
long long a, b, c, d, k;
int main() {
    cin >> a >> b >> c >> d >> k;
    long long prv = a;
    for(int i = 1; i <= k; ++i) {
        a = a * b - c;
        if (a <= 0) {
            cout << 0;
            return 0;
```

```
        }
        if (a >= d) {
            cout << d;
            return 0;
        }
        if (a == prv) {
            cout << a;
            return 0;
        }
    }
    cout << a;
}
```

Часть V
Вывод

Я изучил конструктивные алгоритмы, сделал некоторые основные наблюдения, хорошо использовал рекурсивный алгоритм.