Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №3**

по «Алгоритмам и структурам данных»

Timus

Выполнил:

Студент группы P32092

Голиков А.С.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2023

Задача №1521 «Военные учения 2»

Пояснение к примененному алгоритму:

аведем массив индикаторов, где на i-том месте будем хранить 1, если солтата еще не отправили куда-то там и 0 если отправили

Задача свелась к циклическому поиску k-той единицы, причем как только мы нашли какую-то, нужно узнать, какую искать теперь.

Если справа от текущего места >= k единиц, то давайте искать там. Если меньше, то нужно искать в левой половине дополняющее количество

Эту штуку вполне удобно умеет делать ДО, давайте просто его построим и будем наслаждаться операциями за логарифм.

Асимптотика - O(nlogn)

Код решения:

#include <iostream>

#include <cmath>

#include <vector>

using namespace std;

int find\_k(const vector<int> &a, int k, int v, int l, int r) {

if (r - l == 1) {

return l;

}

if (k > a[v]) k = (k - 1) % (a[v]) + 1;

int mid = (l + r) / 2;

if (a[2 \* v] >= k) {

return find\_k(a, k, 2 \* v, l, mid);

} else {

return find\_k(a, k - a[2 \* v], 2 \* v + 1, mid, r);

}

}

void change(vector<int> \*a, int v, int val) {

(\*a)[v] = val;

while (v /= 2) {

(\*a)[v] = (\*a)[v \* 2] + (\*a)[v \* 2 + 1];

}

}

int sum(const vector<int> &a, int v, int l, int r, int lv, int rv) {

if (l >= rv || lv >= r)

return 0;

if (lv >= l && rv <= r)

return a[v];

int m = (lv + rv) / 2;

int sl = sum(a, 2 \* v, l, r, lv, m);

int sr = sum(a, 2 \* v + 1, l, r, m, rv);

return sl + sr;

}

int main() {

int n, k;

cin >> n >> k;

int p = (int) pow(2, (int) ceil(log2(n)));

vector<int> a(2 \* p);

for (int i = 0; i < n; ++i) {

a[p + i] = 1;

}

for (int i = p - 1; i > 0; --i) {

a[i] = a[2 \* i] + a[2 \* i + 1];

}

int num = 0;

for (int i = 0; i < n; ++i) {

int sl = sum(a, 1, p, num, p, 2 \* p);

int sr = a[1] - sl;

if (sr >= k) {

num = find\_k(a, sl + k, 1, p, 2 \* p);

} else {

num = find\_k(a, k - sr, 1, p, 2 \* p);

}

cout << num - p + 1 << " ";

change(&a, num, 0);

}

return 0;

}

Задача №1650 «Миллиардеры»

Пояснение к примененному алгоритму:

unordered\_map<string, int\_fast64\_t> billionaires;

unordered\_map<string, string> bill\_city;

unordered\_map<string, int\_fast64\_t> cities;

set<pair<int\_fast64\_t, string>, greater<>> fortunes;

Будем для каждого миллиардера помнить, в каком он сейчас городе и сколько у него денег

Так же, для каждого города будем знать, сколько в нем денег всего

Будем хранить сет пар состояние-город чтобы в каждый момент знать, кто самый богатый

Также, нужно будет смотреть, что самый богатый город сейчас только один. Для этого будем бежать от начала сета, пока счетчик < 1.

И все..

Асимптотика - O(mlogm), ге m = n + k

Код решения:

#include <iostream>

#include <map>

#include <unordered\_map>

#include <vector>

#include <set>

using namespace std;

unordered\_map<string, int\_fast64\_t> billionaires;

unordered\_map<string, string> bill\_city;

unordered\_map<string, int\_fast64\_t> cities;

set<pair<int\_fast64\_t, string>, greater<>> fortunes;

vector<string> split(const string &stringToBeSplitted, const string &delimeter) {

vector<string> splittedString;

unsigned long startIndex = 0;

unsigned long endIndex;

while ((endIndex = stringToBeSplitted.find(delimeter, startIndex)) < stringToBeSplitted.size()) {

string val = stringToBeSplitted.substr(startIndex, endIndex - startIndex);

splittedString.push\_back(val);

startIndex = endIndex + delimeter.size();

}

if (startIndex < stringToBeSplitted.size()) {

string val = stringToBeSplitted.substr(startIndex);

splittedString.push\_back(val);

}

return splittedString;

}

int\_fast64\_t parse\_int\_fast64\_t(const string &s) {

int\_fast64\_t ans = 0;

for (char i: s) {

ans \*= 10;

ans += i - '0';

}

return ans;

}

int main() {

ios::sync\_with\_stdio(false);

cin.tie(nullptr);

cout.tie(nullptr);

int n;

cin >> n;

int counter;

map<string, int\_fast64\_t> ans;

for (int i = 0; i < n; ++i) {

string input;

getline(cin >> ws, input);

vector<string> splitted = split(input, " ");

string name = splitted[0];

string city = splitted[1];

int\_fast64\_t fortune = parse\_int\_fast64\_t(splitted[2]);

billionaires[name] = fortune;

bill\_city[name] = city;

cities[city] += fortune;

}

for (const auto &c: cities) {

fortunes.emplace(c.second, c.first);

}

counter = 0;

for (const auto &t: fortunes) {

if (t.first == fortunes.begin()->first) {

++counter;

if (counter > 1) {

break;

}

} else {

break;

}

}

int m, k;

cin >> m >> k;

vector<pair<int, pair<string, string>>> moves(k);

for (int i = 0; i < k; ++i) {

string input;

getline(cin >> ws, input);

vector<string> splitted = split(input, " ");

int day = (int) parse\_int\_fast64\_t(splitted[0]);

string name = splitted[1];

string city = splitted[2];

moves[i] = {day, {name, city}};

}

int prev = 0;

for (int i = 0; i < moves.size();) {

int day = moves[i].first;

if (counter == 1) {

ans[fortunes.begin()->second] += day - prev;

}

while (i < moves.size() && moves[i].first == day) {

string bil\_name = moves[i].second.first;

string dest\_city = moves[i].second.second;

int\_fast64\_t bil\_fort = billionaires[bil\_name];

string curr\_city = bill\_city[bil\_name];

fortunes.erase({cities[curr\_city], curr\_city});

cities[curr\_city] -= bil\_fort;

if (cities[curr\_city] != 0) {

fortunes.emplace(cities[curr\_city], curr\_city);

}

if (cities.find(dest\_city) == cities.end()) {

cities[dest\_city] = bil\_fort;

} else {

fortunes.erase({cities[dest\_city], dest\_city});

cities[dest\_city] += bil\_fort;

}

fortunes.emplace(cities[dest\_city], dest\_city);

bill\_city[bil\_name] = dest\_city;

++i;

}

counter = 0;

for (const auto &t: fortunes) {

if (t.first == fortunes.begin()->first) {

++counter;

if (counter > 1) {

break;

}

} else {

break;

}

}

prev = day;

}

if (counter == 1) {

ans[fortunes.begin()->second] += m - prev;

}

string out;

for (

const auto &a: ans) {

out += a.first + " " + to\_string(a.second) + "\n";

}

cout << out;

return 0;

}