Чернівецький національний університет ім. Юрія Федьковича

Факультет математики та інформатики

Кафедра математичного моделювання

Теорія алгоритмів

Виконав: Апатьєв Нікіта Дмитрович

Чернівці 2022

**Варіант № 8**

Рок-група ЧЕРВ2 дасть концерт у Чернівцях у червні цього року. *N* фанів U2 (*N*<=200) стоять у черзі й хочуть купити білети. Кожен хоче купити один білет, у той же час касир може продати не більше двох білетів одній людині.

Касир витрачає час *ti* для продажу одного білета *i*-му фану і час *ri* для продажу двох білетів *i*-му фану. Тому, якщо це економить час, двоє фанів, що стоять в черзі поруч (*i*-й та (i+1)-й) можуть домовитися, і один з них ((i+1)-й) піде з черги, а інший (*i*-й) купить два білети.

Визначити, за який мінімальний час можна обслужити усю чергу фанів та який варіант (сам бере білет чи об’єднується з сусідом) дасть цей найкращий варіант.

Приклад.

|  |  |
| --- | --- |
| **Вхідні дані** | **Вихідні дані** |
| 7 | 14 |
| 5 4 3 2 1 4 4 | 1 2 2 2 |
| 7 3 4 2 2 4 |  |

#include <iostream>

#include <vector>

using namespace std;

int main(int argc, char const \*argv[]) {

    vector<vector<int>> table(3);

    vector<int> way;

    int n, el; cin >> n;

    for (int i = 0; i < n; i++) {

        cin >> el;

        table[0].push\_back(el);

        table[1].push\_back(0);

    }

    for (int i = 0; i < n - 1; i++) {

        cin >> el;

        table[2].push\_back(el);

    }

    table[2][n-1] = table[1][0] = INT\_MAX;

    for (int i = 1; i < n; i++) {

        int m = min(table[0][i-1], table[1][i-1]);

        table[0][i] += m;

        table[1][i] = table[2][i-1];

        table[2][i] += m;

    }

    cout << endl << min(table[0][n-1], table[1][n-1]) << endl;

    for (int i = 0; i < n; i++) {

        way.push\_back(int(table[0][i] > table[1][i]) + 1);

    }

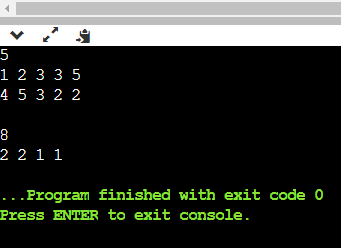
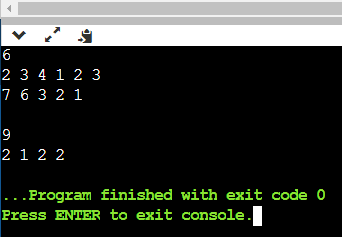
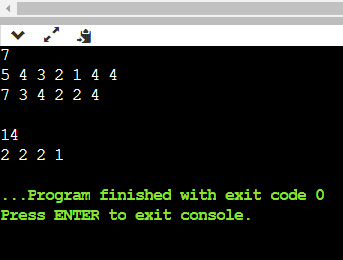
    for (int i = n - 1, s = 0; s != n; i--, s += way[i]) {

        cout << way[i] << ' ';

    }

    return 0;

}



На екран монітора виводяться *n*(1≤*n*≤100000) відрізків, що розміщені на одній [горизонталь](https://moodle.chnu.edu.ua/mod/glossary/showentry.php?eid=12780&displayformat=dictionary)ній прямій. Відрізки задаються двома координатами *X\_LEFTі* та *X\_RIGHTі* (*і*=1,2,…,*n*). Відрізки можуть перекриватися. Знайти сумарну довжину видимих відрізків на екрані.

#include <iostream>

#include <cmath>

#include <vector>

#include <algorithm>

using namespace std;

struct Line {

    float left, right;

    Line() : left(0), right(0) {}

    Line(float x\_left, float x\_right) : left(x\_left), right(x\_right) {}

    float length() {

        return abs(right - left);

    }

    friend ostream& operator<<(ostream &os, Line l) {

        return os << "(" << l.left << ", " << l.right << ")";

    }

    friend istream& operator>>(istream &is, Line l) {

        return is >> l.left >> l.right;

    }

};

void erase(vector<Line> &lines) {

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

        for (int j = 0; j < lines.size(); j++) {

            if (lines[i].right >= lines[j].right && lines[i].left <= lines[j].left && i != j) {

                lines.erase(lines.begin() + j);

                j--;

            }

        }

    }

}

int compare(const void\* a, const void\* b) {

    const Line\* x = (Line\*) a;

    const Line\* y = (Line\*) b;

    if (x->right == y->right) return 0;

    return x->right > y->right ? 1 : -1;

}

int main(int argc, char const \*argv[]) {

    vector<Line> lines = {Line(-5, -2), Line(-3,  0), Line(-2, -1),

                          Line(1,  3),  Line(2,  4),  Line(4,  7)};

    // vector<Line> lines = {{-2, 0}, {-1.5, -0.5}, {-1, 0.5}, {0.5, 1.5}, {1, 3}, {1.5, 2}};

    erase(lines);

    qsort(&lines[0], lines.size(), sizeof(Line), compare);

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

        cout << lines[i] << endl;

    }

    float len = lines[0].length();

    for (int i = 0; i < lines.size() - 1; i++) {

        if (lines[i].right <= lines[i+1].right && lines[i].right >= lines[i+1].left) {

            len += Line(lines[i].right, lines[i+1].right).length();

        } else {

            len += lines[i+1].length();

        }

    }

    cout << len << endl;

    return 0;

}

