Лабораторная работа № 8 по курсу дискретного анализа: жадные алгоритмы

Выполнил студент группы 80-308 Иванов Андрей

Условие

Вариант №2: Выбор отрезков
На координатной прямой даны несколько отрезков с координатами [Li, Ri]. Необходимо выбрать минимальное количество отрезков, которые бы полностью покрыли интервал [0, M].

Формат ввода

На первой строчке располагается число N, за которым следует N строк на каждой из которой находится пара чисел Li, Ri ; последняя строка содержит в себе число M.

Формат вывода

На первой строке число K выбранных отрезков, за которым следует K строк, содержа-щих в себе выбранные отрезки в том же порядке, в котом они встретились во входных данных. Если покрыть интервал невозможно, нужно распечатать число 0. какой-либо из них.

Метод решения

Сортируем набор отрезков по левой границе. Заводим переменную currentEnd, для от-слеживания, насколько интервал уже покрыт. Проходим по отсортированным отрезкам, выбираем отрезок, который начинается до currentEnd и имеет наибольшую правую гра-ницу, пока не покрыли отрезок, либо не обнаружили, что покрыть невозможно. Сортировка работает за O(n*log(n))
Поиск покрытия работает за O(n)

Описание программы

```
#include <iostream>
#include <vector>
#include
<algorithm>
#include <climits>

using namespace std;

struct Segment {
    int lef, right, index
;};
```


int

```
main() {int N;cin>>N;

vector<Segment>
segments(N);for (int i
= 0; i < N; ++i) {
    cin >> segments[i].left >> segments[i
    ].right;segments[i].index = i + 1;
}

int M;
cin >> M;

sort(segments.begin(), segments.end(), [](const Segment& a,
    const Segmereturn a.left < b.left;
});

vector<Segment> result;
int currentEnd = 0, idx = 0;

while (currentEnd
    < M) { bool
    found = false
    ;Segment
    bestSegment;
    bestSegment.right = INT MIN;



    while (idx < N && segments[idx].left <=
        currentEnd) {if (segments[idx].right >=
        bestSegment.right) {
            found = true;
            bestSegment =
```

```
            segments[idx];}
          ±
        +i
        dx
        ;}

        if (!found) {
            cout << 0
            << endl;
            return 0;
        }

        result.push_back(
        bestSegment);currentEnd
        = bestSegment.right;
```



```
    }

    sort(result.begin(), result.end(), [](const Segment& a, const
        Segment&return a.index < b.index;
    });

    cout << result.size() << endl;
    for (const auto& segment : result) {
        cout << segment.left << "␣" << segment.right
    << endl;}

    r
eturn
0;}
```

Дневник отладки

Была ошибка компиляции из-за того, что забыл написать include <iostream>

Тест производительности

Сортировка работает за
O(n*log(n)) Поиск
покрытия работает за O(n)

N	Время работы алгоритма, мс
10000 100000 1000000	1 15 177

Ниже приведена программа , использовавшаяся для определения времени работы алгоритма:

```
void
    t
    e
    st
    ()
    { i
    n
    t
    m
    ,
    n
    ;
    cin >>
    m >> n;
    srand(
    time
    (0));
    vector<Segment> segments;
    for (int i = 0; i < n; ++i ) {
        int right = rand() % (m +
        1); int left = rand() % (
        right + 1);
        segments.push back({lef , right
    , i});}
```



```
    auto start = chrono::high resolution clock:
    :now();fun1(segments, m, n);
    auto end = chrono::high resolution clock::now();

    auto duration = chrono::duration cast<chrono::milliseconds>(end
    — start

    cout << "fun1␣execution␣time:␣" << duration << "␣ms"
<< endl;}
```

Выводы

В ходе выполнения данной лабораторной работы были изучены понятие жадных ал-горитмов и решена конкретная задача с помощью жадного алгоритма. Жадные алго-ритмы на каждом шаге делают локально оптимальный выбор, такой подход способен эффективно решать некоторые задачи.