Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №8 по курсу «Дискретный анализ»

Студент: А. О. Знай
Преподаватель: С. А. Михайлова
Группа: М8О-301Б-21

Дата: 19.10.2023
Оценка:
Подпись:

Москва, 2023

Лабораторная работа №5

Задача: Разработать жадный алгоритм решения задачи, определяемой своим вариантом.

Вариант: На координатной прямой даны несколько отрезков с координатами [Li, Ri]. Необходимо выбрать минимальное количество отрезков, которые бы полностью покрыли интервал [0, M].

Формат входных данных: 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, содержащих в себе выбранные отрезки в том же порядке, в котом они встретились во входных данных. Если покрыть интервал невозможно, нужно распечатать число 0.

1 Описание

Как сказано в [1]: «Жадный алгоритм — алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.» В данной задаче можно поступить таким образом: сначала нужно отсортировать все отрезки по левой границе. Далее, мы будем смотреть все отрезки, левые границы которых не заходят вправо за ту часть, которую мы покрыли, и выбирать из них тот, который заканчивается правее всего. После этого обновлять покрытую часть - она станет правой границей выбранного отрезка. Если такой отрезок найти не удаётся - значит покрытие невозможно. Получается, мы в данный момент выбираем самое оптимальное решение - отрезок, который заканчивается правее всего. Чтобы доказать оптимальность данного алгоритма, надо сравнить его решение с неким оптимальным решением. Допустим, на каком-то этапе отрезок, выбранный жадным алгоритмом не совпадает с отрезком из оптимального. Тогда в оптимальном решении будет другой отрезок, покрывающий данную точку, и мы можем заменить его на отрезок из жадного решения. Решение останется решением, ведь все точки до этого были покрыты. Продолжая таким образом, мы дойдём до конца отрезка. Сложность жадного алгоритма - O(nlogn).

[1]

Код

```cpp
#include <string>
#include <iostream>
#include <vector>
#include <stack>
#include <memory>
#include <chrono>
#include <map>
#include <algorithm>

using namespace std;

bool compare(const pair<int, int>& left, const pair<int, int>& right) {
    return left.first < right.first;
}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);

    int N, M, cols;
    vector<pair<int, int> > segments;
    map<pair<int, int>, int> map;

    cin >> N;
    int l, r;
    for (int i = 0; i < N; i++) {
        cin >> l >> r;
        map.insert({ make_pair(l, r), i });
        segments.push_back(make_pair(l, r));
    }
    cin >> M;

    sort(begin(segments), end(segments), compare);

    int curLeft = 0;
    vector<pair<int, int> > answer;
    bool isFindLeft = true;
    while (isFindLeft) {

        if (curLeft >= M) {
            break;
        }

        isFindLeft = false;

        for (int i = segments.size() - 1; i >= 0; i--) {
            if (segments[i].first <= curLeft && segments[i].second > curLeft) {
```

```
                isFindLeft = true;
                int maxRight = segments[i].second;
                int maxInd = i;
                for (int j = i-1; j >= 0; j--) {
                    if (segments[j].second > maxRight) {
                        maxRight = segments[j].second;
                        maxInd = j;
                    }
                }
                answer.push_back(make_pair(segments[maxInd].first, maxRight));
                curLeft = maxRight;
                break;
            }
        }
    }
    if (isFindLeft) {
        cout << answer.size() << '\n';
        vector<pair<int, int> > answerOrdered(segments.size());
        for (int i = 0; i < answer.size(); i++) {
            answerOrdered[map.at(answer[i])] = make_pair(answer[i].first, answer[i].
                second);
        }
        for (int i = 0; i < answerOrdered.size(); i++) {
            if (answerOrdered[i].first != 0 || answerOrdered[i].second != 0) {
                cout << answerOrdered[i].first << ' ' << answerOrdered[i].second << '\n'
                    ;
            }
        }
    }
    else {
        cout << 0;
    }
}
```

2 Консоль

```
artemznaj@MacBook-Air-Artem lab8 % ./a.out
8
-3 3
6 13
7 10
-2 0
-1 4
4 6
3 5
5 7
10
3
6 13
-1 4
4 6
artemznaj@MacBook-Air-Artem lab8 % ./a.out
3
-1 0
-5 -3
2 5
1
0%
```

3 Тест производительности

Тест производительности представляет из себя следующее: сравнение жадного алгоритма и наивного. Размер тестов - 15 и 30 отрезков.

```
artemznaj@MacBook-Air-Artem lab8 % ./a* <tests/1.in
Наивный алгоритм: 3.802ms
artemznaj@MacBook-Air-Artem lab8 % ./s* <tests/1.in
Жадный алгоритм: 0.020ms
artemznaj@MacBook-Air-Artem lab8 % ./a* <tests/1.in
Наивный алгоритм: 63489.423ms
artemznaj@MacBook-Air-Artem lab8 % ./s* <tests/1.in
Жадный алгоритм: 0.031ms
```

Как видно, наивный алгоритм, который основан на переборе, работает намного дольше даже на маленьких тестах, так как его сложность $O(2^n)$.

4 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», я научился решать задачи с использованием жадных алгоритмов. У данных алгоритмов есть недостатки: выбирать всегда наилучший вариант не всегда приводит к оптимальному решению. Прежде чем применять такой метод, следует удостовериться в его эффективности.

Список литературы

[1] Жадный алгоритм https://ru.wikipedia.org/

[2] Лекции Н.К.Макарова, Московский авиационный институт.