Московский авиационный институт
(национальный исследовательский университет)

Институт информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №8 по курсу дискретного анализа

Студент: А. В. Синявский
Преподаватель: Н. А. Зацепин
Группа: М8О-308Б-18
Дата:
Оценка:
Подпись:

Москва, 2020

Лабораторная работа № 8 по курсу дискретного анализа

Выполнил студент группы М80-308Б-18 МАИ *Синявский Андрей.*

Условие

Разработать жадный алгоритм решения задачи, определяемой своим вариантом. Реализовать программу на языке C или C++, соответсвующую построенному алгоритму.

Вариант №2

На координатной прямой даны несколько отрезков с координатами $[L_i,\ R_i]$. Необходимо выбрать минимальное количество отрезков, которые бы полностью покрыли интервал [0, M].

Формат входных данных

На первой строке располагается число N, за которым следует N строк на каждой из которой находится пара чисел $L_i,\ R_i$; последняя строка содержит в себе число M.

Формат результата

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.

Метод решения

Из заданного набора отрезков на каждом шаге выбираем самый выгодный, т.е. с наибольшим значением правой границы, при этом удовлетворяющий следующему условию: его левая граница не больше текущей, а его правая - больше текущей. Если не нашлось подходящего отрезка - значит покрытие невозможно.

Описание программы

Программа по сути состоит из одной функции main, вне её лежит только воспомогательный класс отрезка, являющийся по сути структурой, хранящей начало, конец и порядковый номер. Ещё за пределами основной функции лежат две функции-компаратора, подающиеся в std::sort в качестве аргумента, чтобы сортировать объекты

класса по нужному полю (comp1 для сортировки по правой границе, comp2 - по порядковому номеру) Весь остальной код, лежащий в мейне, опишу по блокам с указанием строк

Блок	Предназначение
15-16	оптимизация ввода-вывода
18-28	объявление переменных, ввод отрезков и границы
30-41	выбор отрезка, и добавление его в вектор ответа
43-51	вывод ответа в требуемом по заданию формате

Дневник отладки

При создании этой таблицы была использована история посылок.

Время	ведрикт	Описание
2020/10/07 18:50:32	Неправильный ответ	неверно разработал алгоритм, сортил не по тому краю
2020/10/07 19:43:06	Превышено реальное время работы	программа зациклилась на выборе подходящего отрезка
2020/10/03 10:46:102	Ожидает подтверждения	ура

листинг

main.cpp

```
#include <iostream>
#include <vector>
#include <algorithm>
#include "interval.h"

bool comp1(interval &a, interval& b) {
    return a.finish > b.finish;
}

bool comp2(interval &a, interval& b) {
    return a.id < b.id;
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    unsigned long N;
    double L, R, M, cur = 0;
    std::cin >> N;
    std::vector<interval> intervals;
    std::vector<interval> answer;
    for (unsigned long i = 0; i < N; ++i) {
        std::cin >> L >> R;
        intervals.emplace_back(interval(L, R));
```

```
        intervals.back().id = i;
    }
    std::cin >> M;

    std::sort(intervals.begin(), intervals.end(), comp1);
    while(true) {
        unsigned long i = 0;
        for (;i < intervals.size(); ++i) {
            if ((intervals[i].start <= cur) && (intervals[i].finish > cur))
                answer.push_back(intervals[i]);
                cur = intervals[i].finish;
                break;
            }
        }
        if ((i == intervals.size()) || cur >= M) break;
    }

    if (cur < M) {
        std::cout << 0 << '\n';
        return 0;
    }
    std::sort(answer.begin(), answer.end(), comp2);
    std::cout << answer.size() << '\n';
    for (auto & interval : answer) {
        std::cout << interval.start << ' ' << interval.finish << '\n';
    }

    return 0;
}
```

interval.h

```
#ifndef DA_LAB8_INTERVAL_H
#define DA_LAB8_INTERVAL_H

class interval {
public:
    interval() = default;
    interval(double L, double R): start(L), finish(R) {}
    ~interval() = default;

    double start;
```

```
    double finish;
    unsigned long id=0;
};

#endif
```

Недочёты

При поиске нового отрезка программа каждый раз пробегается по всему списку. хранение отрезков в двусвязном списке после сортировки могло бы ускорить программу, так как использованные элементы можно было бы удалять за константное время.

Выводы

Проделав данную работу, я изучил жадный подход при разработке алгоритмов, и надеюсь, это знание пригодится мне при написании алгоритмов в будущем.