Лабораторная работа №8 по курсу дискретного анализа: Жадные алгоритмы

Выполнил студент группы М8О-307Б-20 Мерц Савелий Павлович

Условие

Разработать жадный алгоритм решения задачи, определяемой своим вариантом.

Доказать его корректность, оценить скорость и объем затрачиваемой оперативной памяти.

Вариант:

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

Метод решения:

Написал функцию, принимающая значение левой границы интересующего интервала. Изначально эта граница всегда равна 0. Сама функция ищет отрезок максимальной длины, содержащий эту левую границу, записывает его индекс в ответ и вызывает саму себя с правой границей найденного отрезка. Если уже покрыли интервал или максимум не обновляется с предыдущей итерации, то выходим из функции и выводим ответ.

Тест программы:

Генерировал отрезки с рандомной длиной и левым значением, так чтобы точно покрывали интервал, то есть был отрезок, который полностью покрывал интервал.

10000	**96**
100000	**488**
1000000	**4830**
10000000	**49054**

Описание программы:

Программа написана в одном файле.

Дневник отладки:

Изначально принудительно удалял записанный в ответ отрезок, что давала ошибку в индексах которые я запоминаю. Конечно можно это обойти скопирую входные отрезки, но можно просто не удалять и основываясь только на индексах не записывать повторно тот же отрезок в ответ.

Выводы

Я познакомился с жадным алгоритмами, которы как и динамическое программирование, стараются оптимизировать решение задачи. Жадные алгоритмы принимают локальные решения, допуская, что решение будет оптимальным. Однако далеко не всегда надо быть жадиной и использовать жадные алгоритмы в программировании, хотя это и первое, что приходит в голову после наивного способа решения задачи. Для решения данной задачи подходит отлично. Сложность алгоритма по времени от O(n), если есть отрезок, который полностью интервал покрывает, и до O(n^2), если нет такого отрезка. Сложность алгоритма по памяти O(n).