Лабораторная работа № 8 по курсу дискретного анализа: Жадные алгоритмы

Выполнил студент группы 8О-207Б-17 МАИ *Ваньков Денис.*

Условие

Разработать алгоритм решения задачи, определяемой своим вариантом; Доказать его корректность, оценить время работы алгоритма и объем затраченной оперативной памяти.

Реализовать программу на C или $C++$, реализующую построенный алгоритм. Формат входных и выходных данных описан в варианте задания.

Вариант задания: 2. Выбор отрезков

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 строчке 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 выходных данных: 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

Жадный алгоритм — это алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, считая, что конечное решение окажется также оптимальным. Соответсвенно на каждом шаге алгоритм будет работать корректно.

Описание программы

На вход подается N отрезков, для их хранения я использовал массив собственной структуры данных (*ARRAYS*), состоящей из 2 значений *start* и *end* типа *int*. Затем я сортирую по невозрастанию данный массив по концам этих отрезков, с помощью быстрой сортировки. Далее алгоритм заключается в следующем: В качестве начала создается переменная *time* которой присваивается значение конца отрезка с наименьшим значением *end*, и добавляется в вектор *std::vector* результата *res*. Затем, на каждом шаге алгоритм проверяет начало следующего отрезка, если он находится левее или на границе с предыдущим, то с помощью функции *FindMaxEnd* ищется отрезок с максимальным покрытием, удовлетворяющий условию что его начало не превосходит конца предыдущего. Затем переменной *time* присваивается значение конца найденного отрезка и он

добавляется в вектор результата (*res*). Как только переменная *time* становится больше, чем M - значение конца интервала, алгоритм прекращает работу. Если в конце алгоритма было не найдено такого покрытия, то наш вектор результата остается пустым.

Дневник отладки

В процессе отладке долго не удавалась отследить почему алгоритм проскакивает действительно большие по покрытию интервалы, однако я нашел и исправил данную ошибку.

Выводы

Эта лабораторная работа вызвала у меня некие трудности, так как было не понятно как можно найти минимальное число отрезков, покрывающих данный интервал. Задачу о максимальном покрытии я решил быстро и оставалось только понять как модифицировать ее и реализовать. Данный алгоритм работает корректно, так как: Пусть X_k - число непокрытых элементов после k-го шага. M - размер минимального покрытия. $X_k + 1 \leq X_k - X_k/M = X_k(1 - 1/M)$. 1. Пусть X непокрытых покрываются минимум M' подмножествами. 2. Алгоритм за шаг покроет не меньше X/M'. 3. Иначе, оптимальное покрытие должно покрывать меньше $M' * X/M'$ элементов. 4. Т.к $M' \leq M$, получаем доказательство. Моя программа хранит $\sim O(N)$ данных и имеет $O(N * log(N))$ временную из-за сортировки и функции нахождения максимального покрытия интервала.