Лабораторная работа № 8 Жадные алгоритмы

Выполнил студент группы 08-208 МАИ *Скворцов Александр.*

Условие

1. Разрабтать жадный алгоритм решения задачи, определяемой своим вариантом. Доказать его корректность, оценить скорость и объём затрачиваемой оперативной памяти.

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 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

Чтобы решить с помощью жадных алгоритмов, необходимо:

1. Привести задачу оптимизации к виду, когда после сделанного выбора остаётся решить только одну подзадачу.

2. Доказать, что всегда существует такое оптимиальное решение исходной задачи, которое можно получить путём жадного выбора, так что такой выбор всегда допустим.

3. Показать, что после жадного выбор остаётся подзадача, обладающая тем свойством, что объединение оптимального решения подзадачи со сделанным жадным выбором приводит к оптимальному решению исходной задачи.

Тогда алгоритм решения выглядит следующим образом:

1. Выбираем отрезок, который начинается раньше покрываемого отрезка и заканчивается дальше всех аналогичных. Это жадный выбор.

2. Переходим к рассмотрению непокрытой части.

Доказательство корректности:

- Пусть для покрытия дан отрезок S_{ij} с началом f_i и концом f_j. Отсортируем все отрезки по их началу.

- выберем такой отрезок a_k, что $f_k = max\{f_n | s_n < f_i\}$. Тогда это наш жадный выбор, не существует отрезка, который больше выбранного, и задача сводится к нахождению решения подзадачи S_{kj}.

- Такой выбор всегда принадлежит итоговому решению, так как если он принадлежит, то утверждение верно. В противном случае выберем из решения A отрезок, покрывающий начало, и заменем его на a_k, но так как a_k содержит выбранный, то он пересекается с оставшейся частью решения, а значит новое A также является ответом.

Описание программы

Программа состоит всего из одного файла с единственной функцией main(). Определена дополнительная структура TSegment, описывающая отрезок: его начало s, конец f и номер при вводе.

Дневник отладки

№	Ответ чекера	Причина ошибки
1	Wrong answer at test 03.t	первый отрезок по-умолчанию являлся покрытием
2	Wrong answer at test 04.t	программа некорректно обрабатывала случай, когда отрезок не покрывылся с правой стороны

Тест производительности

В программе используются только 2 вектора, причем первый служит для ввода данных, а длина второго не превышает длину первого. Т.е. итоговая оценка используемой памяти $O(n)$.

Ход работы программы можно разделить на три основные части: сортировка начального вектора отрезков по их началу за $O(n \log n)$, выбор покрытия за $O(n)$ (рассматриваем все отрезки по одному разу за константное время) и сортировка покрытия по номеру за $O(n \log n)$. Таким образом итоговая оценка составляет $O(n \log n)$.

Количество отрезков	Время работы
40000	105
80000	194
160000	384
320000	769
640000	1516

Как видно из таблицы, время работы на выбранном количестве элементов оказалось сублинейным, но это не протеворечит верхней оценке $O(n \log n)$. К сожалению, на большем количестве элементов мой компьютер начинает работать медленно.

Выводы

Жадные алгоритмы, как и динамическое программирование, — не конкретный алгоритм, а метод их разработки. Используются они в задачах оптимизации, когда локальный оптимальный выбор на каждом этапе, может привести к общему глобальному оптимальному решению. Однако даже если такую структуру найти не удается, решение, найденое жадным алгоритмом, может стремится к требуемому, а может и вообще не существовать.

Сложность программирования, как мне показалось, невысокая, поэтому их часто используют вместо динамического программирования, если это возможно.