Лабораторная работа № 8 по курсу дискретного анализа: Жадные алгоритмы

Выполнил студент группы 08-207 МАИ *Днепров Иван.*

Условие

1. Разрабтать жадныи алгоритм решения задачи, определяемои своим вариантом. Доказать его корректность, оценить скорость и объем затрачиваемои оперативнои памяти. Реализовать программу на языке C или C++, соответствующую построенному алгоритму. Формат входных и выходных данных описан в варианте задания.

2. Вариант 5. Оптимальная сортировка чисел.

3. Дана последовательность длины N из целых чисел 1, 2, 3. Необходимо наити минимальное количество обменов элементов последовательности, в результате которых последовательность стала бы отсортированнои. Входные данные: число N на первои строке и N чисел на второи строке. Выходные данные: минимальное количество обменов.

Метод решения

Суть алгоритма заключается в том, что в начеле я нахожу число чисел, не превышающее данное и записываю его в вектор count. Для этого я считаю сколько чисел каждого типа во входном векторе для сортировки numbers, а за тем, начиная с конца, в каждую поледующую ячейку записываю суммы предыдущих, а в нулевую – ноль.

После чего можно приступак к самой сортировке. Благодаря вектору count теперь я могу легко понять, стоит ли конуретное число на своем месте или требеутся перестановка. В цикле я пробегаю по всем элементам и если требуется перестановка, оцениваю опять же при помощи count, какое число должно стоять на этом месте и нахожу его из промежутка, в котором стоит текущее число, после чего делаю swap и увеличиваю счётчик перестановок.

После того, как я пробежал все элементы, возвращаю счётчик перестановок.

В случае, если порадок подобран таким образом, чтобы алгоритм алгоритм работал наиболее медленно, то искать число дла замены текущего числа мы будем за n итераций, где n – количество чисел равных текущему. И в худшем случае таких перестановок придётся совершить k - 1, где k – общее количество чисел для сортировки. Тогда общая сложность получается $k \cdot k/3$, Следовательно сложность алгоритма квадратичная, хоть коэффицент и меньше эдиницы.

Доказать корректность работы моего алгоритма довольно просто, из определения он перастваляет только те числа, которые стоят на противоположных позициях, то есть

после пересановки числа сразу принимают необходимую позицию и для этой пары дальнейшие перстановки не требуются. В связи с чем алгоритм осуществаляет оптимальное количество перестановок и верно осуществляет сортировку.

Описание программы

В отличие от прошлой лабораторной я решил обособить основные действия в отдельную функцию, которую назвал SortWithSwapsCounting. Как лего догадаться по названию, она сортирует вектор и считает количество перестановок. А в main я только зачитываю входные параметры, вызываю SortWithSwapsCounting и вывожу полученное число.

Выводы

Хоть в этом алгоритме оптимизированно число сравнений и перестановок за счёт вектора count , сложность всё равно получилась $O(n^2)$. Да, это сильно больше линейной сложноси алгоритмов сортировки, которых мы проходили в прошлом семестре, за то в случае, если swap элементов занимает очень много времени, эта сортировка будет работать быстрее, чем сортировки за линию, и это доказывает, что она тоже имеет право на существование.

В большинстве случаев её сложность будет ближе к линеарифмической, за счёт того, что средняя сложность поиска ближе к логорифму, чем к $O(n)$.

Памяти для моего алгоритма требуется линейное количество, потому что я не копирую вектор с числами для сортировки, а вектор для продсчёта занимает константную память.