МГТУ им. Баумана

Лабораторная работа №3

По курсу: "Анализ алгоритмов"

Сортировки

Работу выполнил: Подвашецкий Дмитрий, ИУ7-54Б

Преподаватели: Волкова Л.Л., Строганов Ю.В.

Москва, 2019

Оглавление

Введение

Алгоритм сортировки - это алгоритм, позволяющий упорядочить элементы в некотором списке. Сортировки - это основа, которую учат все, кто так или иначе хочет заниматься чем-либо связанным с программированием.

За все время было создано огромное множество различных алгоритмов сортировки, каждая из которых обладает какими-либо особенности. В данной лабораторной работе я постраюсь это продемонстрировать.

Задачами данной лабораторной работы являются:

1. выбор и изучение трех алгоритмов сортировки;
2. реализация выбранных алгоритмов;
3. теоретический анализ сложности;
4. эксперементальное подтверждение различий во временной эффективности алгоритмов;
5. описание и обоснование полученных результатов в отчете о выполненной лабораторной работе, выполненного как расчётно-пояснительная записка к работе.

1 Аналитическая часть

Для рассмотрения в этой лабораторной работе мною были выбраны алгоритмы:

1. быстрой сортировки;
2. сортироки пузырьком;
3. сортировки вставками.

1.1 Быстрая сортировка

Суть данного алгоритма заключается в выборе некоторого опортного элемента (обычно выбирают либо последний, либо средний) и дальнейшем разбиении списка на два подсписка: все элементы меньше опортного и все те, что больше опорного. Далее для каждого из двух подсписков рекурсивно применяется тот же алгоритм сортировки.

Обозначим:

qSort(list) - применение алгоритма быстрой сортировки к некоторому списку list.

$list = l_0, l_1, ..., l_n$

$listL = l_i : l_i <= l_0, i = 1..n$

$listR = l_i : l_i > l_0, i = 1..n$

Тогда алгоритм быстрой сортировки можно записать как:

$$qSort(list) = qSort(listL) + l_0 + qSort(listR) \tag{1.1}$$

1.2 Сортировка пузырльком

Данный алгоритм заключается в проходе списка слева направо до конца. Если текущий элемент больше следующего, то необходимо поменять их местами (для сортировки по возрастанию). Заметим, что после, этот процесс необходимо повторять до тех пор, пока массив не будет отсортирован. В случае если алгоритм никак не модифицирован, то необходимо повторить кол-во раз, равного длине массива.

1.3 Сортировка вставками

Суть этого алогоритма аключается в том что, на каждом шаге алгоритма мы берем один из элементов массива, находим позицию для вставки и вставляем.

Стоит отметить что массив из 1-го элемента считается отсортированным. [1] В ходе работы данного алгоритма, при обратоке i-го элемента можно быть уверенным в том, что левая часть является полностью остортированной.

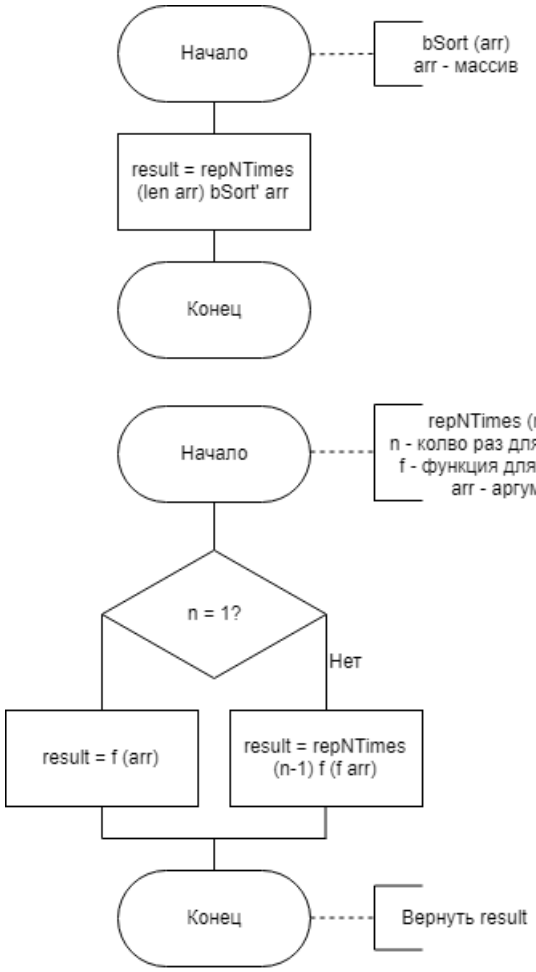
Вывод

В данном разделе мною было рассмотрены и вкратце описаны рассматриваемые мною алгоритмы.

2 Конструкторский раздел

2.1 Схемы алгоритмов и их анализ

2.1.1 Сортировка пузырьком



bSort - это функция 'надстройка', едиснтвенная её задача - вызов функции repNTimes, следовательно

$$F_{bSort} = F_{repNTimes} \tag{2.1}$$

Максимальная глубина рекурсии функции repNTimes = N, что равно длине массива

$$F_{repNTimes} = N(2 + F_{bSort'}) - 1 \tag{2.2}$$

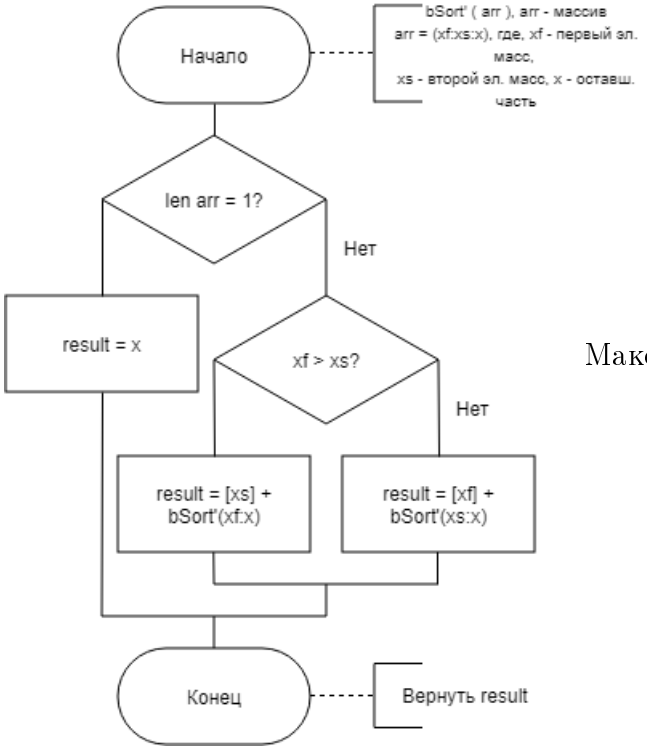


Рис 1. Схема алгоритма сортировки пузырьком.

Максимальная глубина рекурсии bSort' так же равна N

$$F_{bSort'} = 3N - 1 \qquad (2.3)$$

Анализ трудоемкости:

$$F_{bSort} = F_{repNTimes} = N(2 + F_{bSort'}) - 1 = N(2 + 3N - 1) - 1 = N + 3N^2 - 1 \quad (2.4)$$

Для данной реализации алгоритма сортировки пузрьком, нет различий в трудоемкости при обратке обратно отсортированного массива, прямого или случайного.

Общая сложность алгоритма для всех случаев: $O(N^2)$

Список литературы

1. В мире алгоритмов: Сортировка Вставками. [Электронный ресурс] Режим доступа: https://habr.com/ru/post/181271/ Последнняя дата обращения: 12.11.2019