Жуков Кирилл Б20-205

Лабораторная работа №1
По курсу «Языки программирования и методы программирования»
(информатика, 3 семестр)

Техническое задание

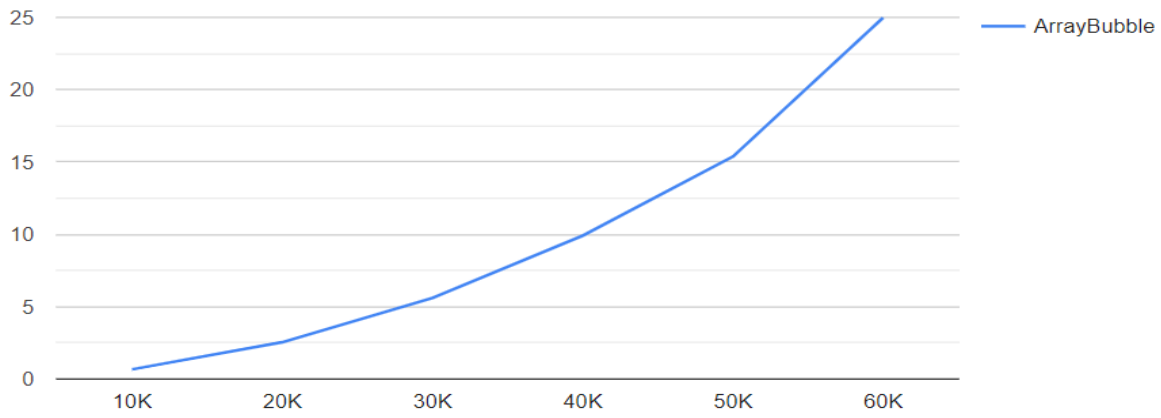
Введение/

На основе реализации класса Sequence <T>, полученной в рамках курса за 2-й семестр, была написана программа на C++, способная производить сортировку (достаточно произвольных) исходных данных с помощью одного из 3 алгоритмов сортировки, а также сравнивать скорость выполнения того или иного алгоритма.

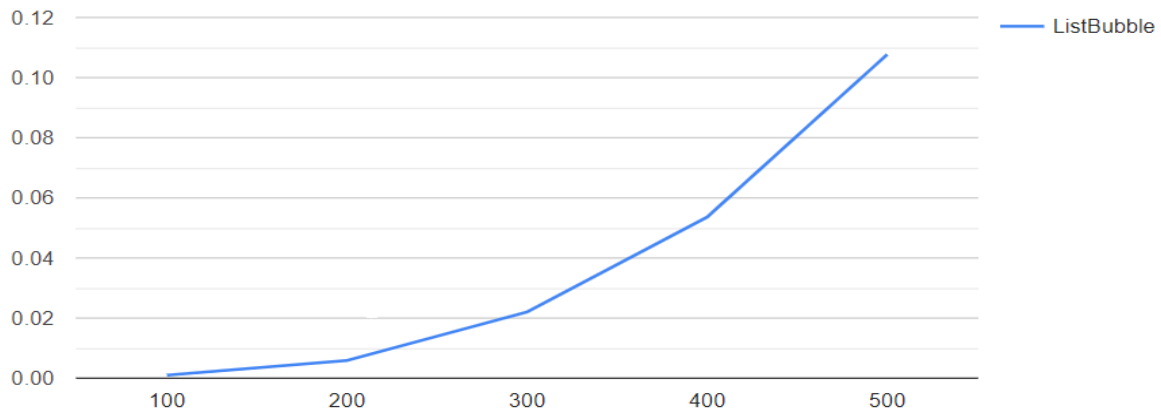
Алгоритмы сортировки.

В рамках лабораторной работы были реализованы следующие алгоритмы:

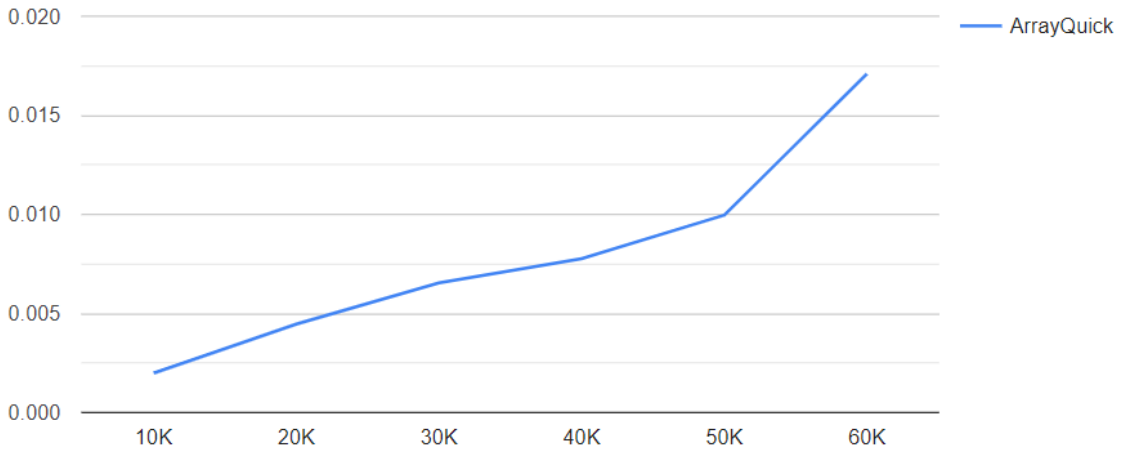
1. “Bubble Sort” (BubbleSort()). Сортировка пузырьком – простейший способ отсортировать исходные данные. Алгоритм проходит несколько раз по массиву, каждый раз меняя соседние элементы, если они стоят в неправильном порядке.
Для массива:



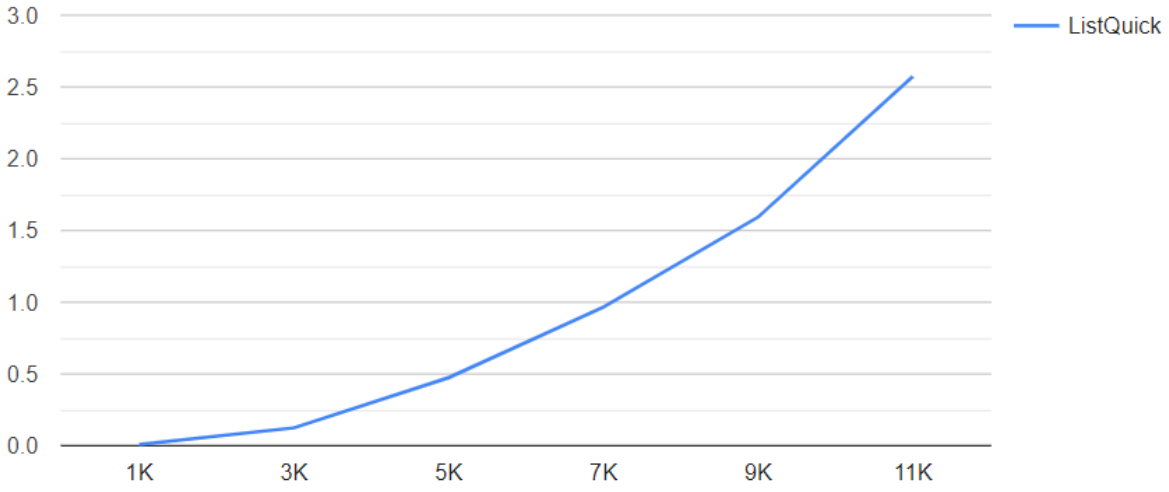
Для связного списка:



2. “Quick Sort” (QuickSort()). Быстрая сортировка, за опорный элемент берется крайний левый элемент участка. Сначала массив разбивается на 3 непрерывные части: элементы, меньшие чем опорный, равные опорному и большие опорного. После чего рекурсивно сортируются участки с элементами меньшими, и большими опорного.
Для массива:

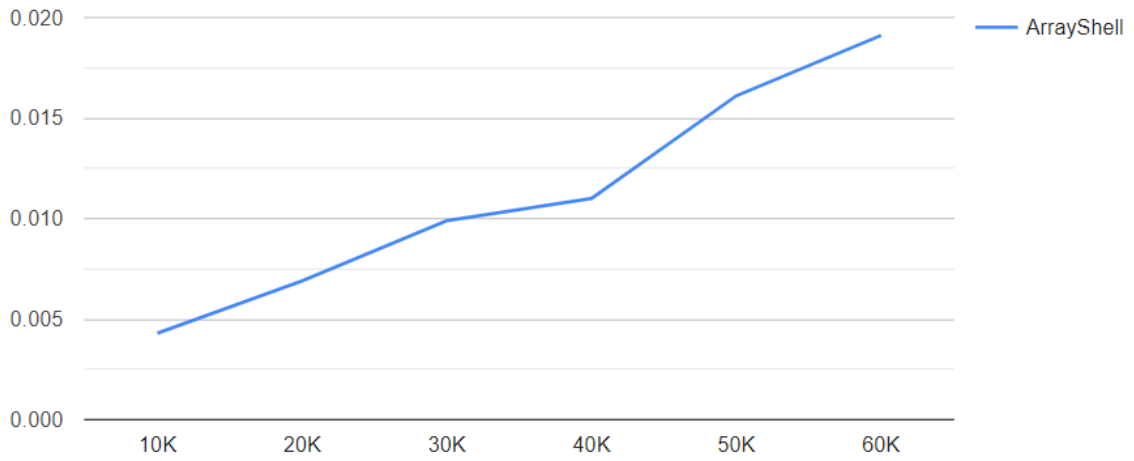


Для связного списка:

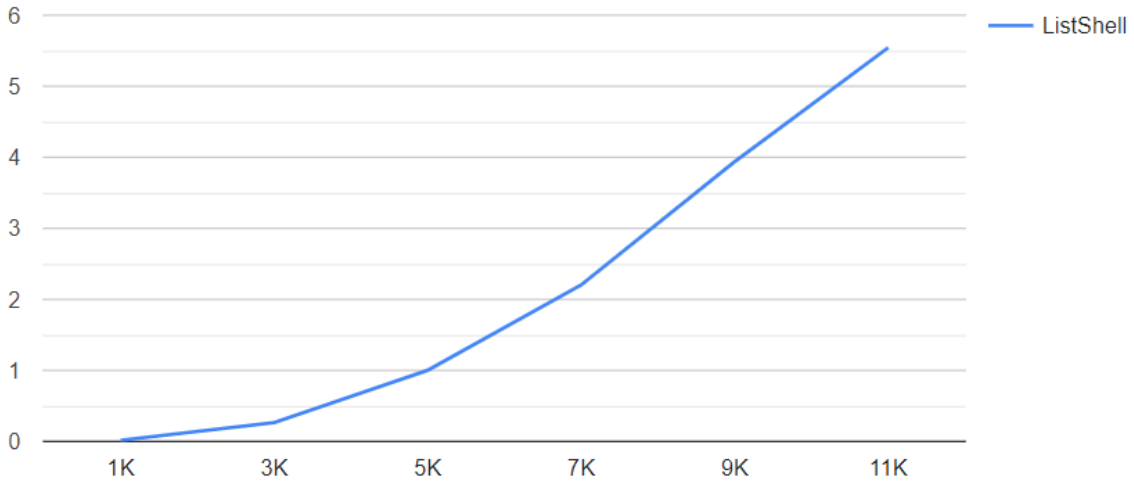


3. “Shell Sort”(ShellSort()):Идея метода заключается в сравнение разделенных на группы элементов последовательности, находящихся друг от друга на некотором расстоянии. Изначально это расстояние равно d или N/2, где N — общее число элементов. На первом шаге каждая группа включает в себя два элемента расположенных друг от друга на расстоянии N/2; они сравниваются между собой, и, в случае необходимости, меняются местами. На последующих шагах также происходят проверка и обмен, но расстояние d сокращается на d/2, и количество групп, соответственно, уменьшается. Постепенно расстояние между элементами уменьшается, и на d=1 проход по массиву происходит в последний раз.

Для массива:



Для связного списка:



Все алгоритмы сортировки принимают в качестве параметра способ сравнения элементов. В программе реализованы два способа сравнения: меньше или равно (cmp1()), меньше(cmp2()) и больше (cmp3()). Эти функции позволяют соответственно сортировать массивы по возрастанию или по убыванию.

Также присутствует консольный интерфейс, позволяющий проверить корректность работы алгоритмов на последовательностях, любой длины.
Присутствует возможность создания произвольной последовательности фиксированной длины и отсортированной последовательности фиксированной длины.
Консольное приложение также предоставляет возможность узнать время работы алгоритмов на тех или иных входных данных

Про сложность изученных алгоритмов.
1.
Алгоритм сортировки пузырьком имеет сложность O(n²), где n – количество элементов. Из формулы видно, что сложность сортировки пузырьком квадратично зависит от количества сортируемых элементов.

2 .Сложность быстрой сортировки в среднем – O(n*log (n)) (при случайном выборе опорного элемента),где n – количество элементов.

3. Сложность сортировки Шелла: O($n \log^2 n$)
*****Еще пару слов про сортировку Шелла: Сортировка Шелла является попыткой улучшить простейшую сортировку вставками.**

Алгоритм	Временная сложность			Пространственная сложность
	Наилучший случай	Средний случай	Наихудший случай	Наихудший случай
Быстрая сортировка	O(n log(n))	O(n log(n))	O(n^2)	O(log(n))
Сортировка слиянием	O(n log(n))	O(n log(n))	O(n log(n))	O(n)
Timsort	O(n)	O(n log(n))	O(n log(n))	O(n)
Пирамидальная сортировка	O(n log(n))	O(n log(n))	O(n log(n))	O(1)
Сортировка пузырьком	O(n)	O(n^2)	O(n^2)	O(1)
Сортировка вставками	O(n)	O(n^2)	O(n^2)	O(1)
Сортировка выбором	O(n^2)	O(n^2)	O(n^2)	O(1)
Сортировка Шелла	O(n)	O((nlog(n))^2)	O((nlog(n))^2)	O(1)
Блочная сортировка	O(n+k)	O(n+k)	O(n^2)	O(n)
Поразрядная сортировка	O(nk)	O(nk)	O(nk)	O(n+k)