Лабораторная работа 1. Поиск и сортировка в линейной статической структуре данных

Часть 1. Алгоритмы поиска

Задание 1. Задан одномерный массив целых чисел. Организовать бинарный и интерполяционный поиск введенного с клавиатуры значения.
Примечание: Для организации бинарного и интерполяционного поиска следует массив отсортировать.
Задание 2. Составить программу поиска введенного с клавиатуры слова на основе алгоритма Кнута-Мориса-Пратта и Боуэра-Мура (два отдельных метода). Промежуточные результаты сдвига слова относительного текста вывести на экран.
Дополнительное задание.
1. Линейным поиском найти второй по максимальности элемент массива.

Часть 2. Сортировка данных
В соответствии со своим вариантом (Вариант задания определяется так: (n mod 16 +1), где n– порядковый номер студента в списке) необходимо реализовать 3 алгоритма сортировки и провести экспериментальное исследование их эффективности. Распределение алгоритмов по вариантом приведено в табл. 1.
Таблица 1. Распределение заданий по вариантам

№ варианта	Алгоритмы, не использующие операцию сравнения	Квадратичные алгоритмы сортировки	«Быстрые» алгоритмы сортировки
1	Counting Sort	Bubble Sort	MergeSort
2	Radix Sort	Selection Sort	HeapSort
3	Counting Sort	Insertion Sort	QuickSort
4	Radix Sort	Shell Sort	MergeSort
5	Counting Sort	Bubble Sort	HeapSort
6	Radix Sort	Selection Sort	QuickSort
7	Counting Sort	Insertion Sort	MergeSort
8	Radix Sort	Shell Sort	HeapSort
9	Counting Sort	Bubble Sort	QuickSort
10	Radix Sort	Selection Sort	MergeSort
11	Counting Sort	Insertion Sort	HeapSort
12	Radix Sort	Shell Sort	QuickSort
13	Counting Sort	Bubble Sort	MergeSort
14	Radix Sort	Selection Sort	HeapSort
15	Counting Sort	Shell Sort	QuickSort
16	Radix Sort	Insertion Sort	MergeSort

1. Сортировка подсчётом (англ. counting sort)
2. Поразрядная сортировка (англ. radix sort)
3. Сортировка пузырьком / Bubble sort
4. Сортировка вставками / Insertion sort
5. Сортировка Шелла / Shell sort
6. Сортировка выбором / Selection sort
7. Пирамидальная сортировка / Heapsort
8. Быстрая сортировка / Quicksort
9. Сортировка слиянием / Merge sort

Экспериментальное исследование
- Необходимо измерить время работы каждого алгоритма при различном количестве элементов в массиве — **заполните таблицу 2 для каждого алгоритма**
- По результатам экспериментов определите какой алгоритм работает быстрее и почему – В экспериментах используйте массивы с целочисленными элементами типа

В отчете должны быть следующие разделы:

- **Описание алгоритмов** — общая характеристика алгоритмов (привести код), их свойства (in place, stable), вычислительная сложность и сложность по памяти **организация экспериментов** — в этом разделе следует указать конфигурацию машины,
- на которой вы проводили эксперименты (модель процессора, объем памяти; версию операционной системы и ключи компиляции программы)
- **Результаты экспериментов** — таблицы и выводы о эффективности алгоритмов

Таблица 2. Результаты экспериментов

№	**Количество элементов в массиве**	**Время выполнения алгоритма, с**
1	10 000	
2	150 000	
3	1000 000	

Контрольные вопросы

На защите отчета вы должны ответить на следующие вопросы:

1) Что такое вычислительная сложность алгоритма (computational complexity)?
2) Что означают записи $f(n) = O(g(n))$, $f(n) = \Theta(g(n))$, $f(n) = \Omega(g(n))$?
3) Какой алгоритм сортировки называется устойчивым (stable)?
4) Какая вычислительная сложность в худшем случае у алгоритмов, которые Вы реализовали?
5) Какие алгоритмы сортировки с вычислительной сложностью $O(n\log n)$ для худшего случая вам известны?
6) Известны ли вам алгоритмы сортировки работающие быстрее $O(n\log n)$ для худшего случая?