ЛАБОРАТОРНЫЕ РАБОТЫ
по курсу «Алгоритмы и анализ сложности»

1. Внутренние сортировки

Требуется реализовать 5 алгоритмов:

- рекуррентное слияние,
- сортировку Шелла,
- пирамидальную сортировку,
- быструю сортировку с одним рекурсивным вызовом,
- цифровую сортировку целых неотрицательных чисел.

Программа должна последовательно выполнять все 5 сортировок для одинаковых массивов целых неотрицательных чисел по заданной длине и типу массива. Типы массивов:

- упорядоченные по возрастанию,
- упорядоченные по убыванию,
- случайные.

Каждая сортировка должна завершаться вычислением времени работы и проверкой упорядоченности полученного массива.

Для получения более высокой оценки нужно провести сравнение быстродействия всех сортировок при n = 50000, 100000, 500000, 1000000, 5000000, 10000000 и построить графики зависимости времени работы от числа точек (в Excel).

2. Внешняя сортировка

Требуется реализовать многофазную сортировку с 2 входными файлами. На входе задаются:

- N – общее число сортируемых значений,
- L – длина начальных серий.

Вначале N генерируемых случайных значений записываются в некоторый начальный файл. Из этого файла последовательно считываются группы по L значений (последняя может быть короче). В каждой группе значения упорядочиваются с помощью любой внутренней сортировки. Получаемые серии распределяются по входным файлам с добавлением нужного числа пустых серий.

Числа в файлах должны быть представлены во внутренней кодировке (файлы не текстовые). Необходимо проверить упорядоченность получаемого файла и выдать распределение начальных и пустых серий по входным файлам на каждом проходе.

3. Хеш-таблицы

Требуется построить хеш-таблицу, для поиска в которой используется метод открытой адресации (размещение и поиск элементов – обязательно, удаление – желательно). Длина таблицы q – простое число в диапазоне 10-20 тысяч. Таблица строится для набора случайных символьных строк длиной 1-20 символов и хранит номера или адреса этих строк. Хеш-функция для строки S длины L:

$f(S) = ((\ldots(S[1] * 31 + S[2]) * 31 + \ldots + S[L-1]) * 31 + S[L]) \bmod q.$

Необходимо вычислить среднюю трудоемкость поиска при различной заполненности таблицы (например, 25, 50, 75, 90 и 99%). Для этого нужно сначала разместить в таблице нужное число строк, а потом для каждой строки подсчитать число шагов, выполняемых при ее поиске. Все вычисления провести для трех вариантов: линейные пробы, квадратичные пробы и двойное хеширование.

4. Красно-черные или 2-3-деревья

Требуется написать программу построения красно-черного или 2-3-дерева с возможностью добавления и поиска элементов (без удаления). Элементы – целые числа в диапазоне 0…99. Вход программы: N – число начальных элементов, V – вариант генерации начальных

элементов (0 – случайные, 1 – упорядоченные). По данным элементам строится начальное дерево, к которому можно добавлять новые значения и проводить поиск.
Если текущее число вершин не превышает 50, то текущее дерево должно отображаться на экране (возможный, но не лучший вариант – вывод в файл после каждого изменения текущего дерева). Это можно реализовать с помощью простой рекурсивной функции даже в консольном режиме. Параметрами такой функции будут номер строки и номера левой и правой позиции на экране, определяющие поле, в котором выводится нужное значение (дуги дерева рисовать не нужно). После поиска необходимо сообщать о его результате и/или отмечать найденное значение.
Упрощенный вариант: построение случайного бинарного дерева с возможностью добавления, поиска и удаления элементов. Вход и выход – как указано выше.

5. Поиск цепочек символов

Требуется реализовать 2 алгоритма поиска в строке по образцу:

- алгоритм Бойера-Мура,
- алгоритм Кнута-Морриса-Пратта или алгоритм Рабина-Карпа.

Входные данные:

- B, E - 2 символа, определяющие нижнюю и верхнюю границу диапазона символов анализируемой строки (они могут и совпадать),
- N – длина анализируемой строки (N<2000),
- строка-образец.

По значениям B, E, N генерируется строка, в которой с помощью обоих алгоритмов ищутся все вхождения подстроки-образца. Необходимо выдать на экран входную строку и информацию о найденных подстроках (отметить начальные символы или указать их номера в строке), а также общее число сравнений символов строки и образца для обоих алгоритмов.

6. Задача коммивояжера

Требуется реализовать 2 алгоритма:

- переборный алгоритм с отсечениями (метод ветвей и границ),
- приближенный алгоритм обхода на основе минимального остова.

Полный граф для задачи коммивояжера определяется набором случайных вершин в единичном квадрате [0…1, 0…1], веса ребер равны расстояниям между соответствующими вершинами.
Проверить работу обоих алгоритмов на графах с 10, 15, 20 вершинами, а также приближенного алгоритма на графах с 50, 100, 500, 1000 вершин. Для каждого графа и алгоритма вычислять время работы, веса минимального остова и маршрутов, а также оценку качества – отношение веса маршрута к весу остова. Построить таблицы или графики зависимости данных оценок от числа точек.