**Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики**

**Алгоритмы и структуры данных**

**Лабораторная работа №1**

**Введение**

Выполнил:

Бараканов Жаргал Мырзабекович

Факультет ИКТ

Группа K3121

Преподаватель:

Харьковская Татьяна Александровна

**Санкт-Петербург**

**21.10.2021**

**Задание 1.**

Пункт 1.

Используя псевдокод процедур Merge и Merge-sort из презентации к Лекции 2 (страницы 6-7), напишите программу сортировки слиянием на Python и проверьте сортировку, создав несколько рандомных массивов, подходящих под параметры:

* Формат входного файла (input.txt). В первой строке входного файла содержится число n (1 ≤ n ≤ 2 · 104) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 109;
* Формат выходного файла (output.txt). Одна строка выходного файла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел;
* Ограничение по времени. 2сек;
* Ограничение по памяти. 256 мб.

*Решение:*

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Пункт 2.

Для проверки можно выбрать наихудший случай, когда сортируется массив размера 1000, 104, 105 чисел порядка 109, отсортированных в обратном порядке; наилучший, когда массив уже отсортирован, и средний. Сравните, например, с сортировкой вставкой на этих же данных.

*Результат:*

Изображение выглядит как текст

Автоматически созданное описание

Сортировка вставкой работает быстрее только в случае уже отсортированного массива. При сортировки слиянием оказалось, что средний – рандомный массив сортируется почти столько же, сколько и полностью не отсортированный.

Пункт 3.

Перепишите процедуру Merge так, чтобы в ней не использовались сигнальные значения. Сигналом к остановке должен служить тот факт, что все элементы массива L или R скопированы обратно в массив A, после чего в этот массив копируются элементы, оставшиеся в непустом массиве.

*Решение:*

Изображение выглядит как текст

Автоматически созданное описание

Данное условие обеспечивает желаемый результат. Если индекс i станет не меньше n1, то в массив пойдут элементы из правой части и наоборот.

**Задание 3.**

Инверсией в последовательности чисел A называется такая ситуация, когда i < j, а Ai > Aj . Количество инверсий в последовательности в некотором роде определяет, насколько близка данная последовательность к отсортированной. Например, в сортированном массиве число инверсий равно 0, а в массиве, сортированном наоборот - каждые два элемента будут составлять инверсию (всего n(n − 1)/2).

Дан массив целых чисел. Ваша задача — подсчитать число инверсий в нем.

Подсказка: чтобы сделать это быстрее, можно воспользоваться модификацией сортировки слиянием.

* Формат входного файла (input.txt). В первой строке входного файла содержится число n (1 ≤ n ≤ 105) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 109;
* Формат выходного файла (output.txt). В выходной файл надо вывести число инверсий в массиве;
* Ограничение по времени. 2сек;
* Ограничение по памяти. 256 мб.

*Решение:*

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Задание 4.**

В этой задаче вы реализуете алгоритм бинарного поиска, который позволяет очень эффективно искать (даже в огромных) списках при условии, что список отсортирован. Цель - реализация алгоритма двоичного (бинарного) поиска.

* Формат входного файла (input.txt). В первой строке входного файла содержится число n (1 ≤ n ≤ 105) — число элементов в массиве, и последовательность a0 < a1 < ... < an−1 из n различных положительных целых чисел в порядке возрастания, 1 ≤ ai ≤ 109 для всех 0 ≤ i < n. Следующая строка содержит число k, 1 ≤ k ≤ 105 и k положительных целых чисел b0, ...bk−1, 1 ≤ bj ≤ 109 для всех 0 ≤ j < k;
* Формат выходного файла (output.txt). Для всех i от 0 до k − 1 вывести индекс 0 ≤ j ≤ n − 1, такой что ai = bj или -1, если такого числа в массиве нет;
* Ограничение по времени. 2сек;
* Ограничение по памяти. 256 мб.

*Решение:*

Изображение выглядит как текст

Автоматически созданное описание

**Задание 5.**

Ваша цель - использовать метод "Разделяй и властвуй" для разработки алгоритма проверки, содержится ли во входной последовательности элемент, который встречается больше половины раз, за время O(n log n);

* Формат входного файла (input.txt). В первой строке входного файла содержится число n (1 ≤ n ≤ 105) — число элементов в массиве. Во второй строке находятся n положительных целых чисел, по модулю не превосходящих 109, 0 ≤ ai ≤ 109;
* Формат выходного файла (output.txt). Выведите 1, если во входной последовательности есть элемент, который встречается строго больше половины раз; в противном случае – 0;
* Ограничение по времени. 2сек;
* Ограничение по памяти. 256 мб.

*Решение:*

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Задание 7.**

Можно найти максимальный подмассив за линейное время, воспользовавшись следующими идеями. Начните с левого конца массива и двигайтесь вправо, отслеживая найденный к данному моменту максимальный подмассив. Зная максимальный подмассив массива A[1..j], распространите ответ на поиск максимального подмассива, заканчивающегося индексом j + 1, воспользовавшись следующим наблюдением: максимальный подмассив массива A[1..j + 1] представляет собой либо максимальный подмассив массива A[1..j], либо подмассив A[i..j + 1] для некоторого 1 ≤ i ≤ j + 1. Определите максимальный подмассив вида A[i..j + 1] за константное время, зная максимальный подмассив, заканчивающийся индексом j.

*Решение:*

Изображение выглядит как текст

Автоматически созданное описание