Отчёт

По лабораторной работе №1 “Поиск и сортировка в линейной статической структуре данных”

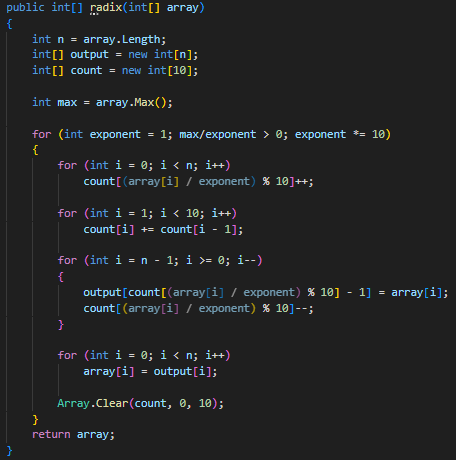
Подготовил: Красуцкий Владимир

Уфа

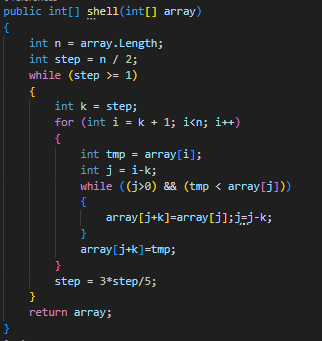
2025

Цель работы: проверка времени выполнения различных алгоритмов относительно друг друга.

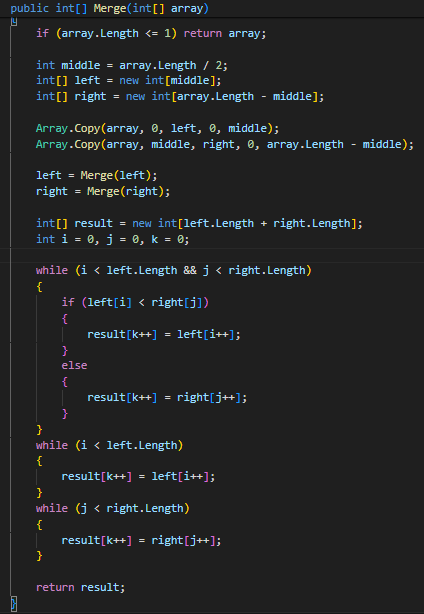
Radix Sort (поразрядная сортировка) – это алгоритм сортировки, который сортирует числа, обрабатывая их поразрядно, начиная с младших разрядов и двигаясь к старшим. Он использует дополнительную память, а его стабильность зависит от вспомогательной сортировки. Его сложность равна O(nk), где k – количество разрядов.



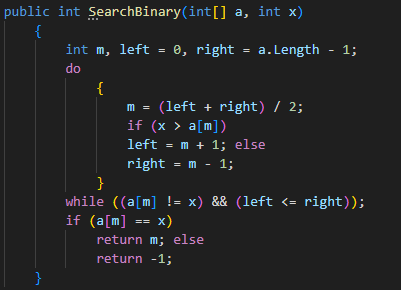
Shell Sort (сортировка Шелла) – это алгоритм сортировки, который сравнивает на начальных стадиях значения, расположенные достаточно далеко друг от друга в упорядочиваемом наборе данных. Такая сортировка обычно быстрее, нежели чем если используется сравнение соседних элементов. Эффективность объясняется тем, что сдвигаемые элементы быстро попадают на нужные места. Он не требует дополнительной памяти, является нестабильным. Сложность зависит от массива, в лучшем случае O(n logn), средний O(n^1,25), Худший O(n^2).



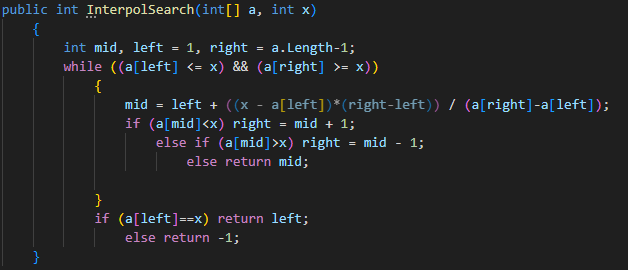
Merge Sort (Сортировка слиянием) – это алгоритм сортировки, использующий принцип разделяй и властвуй. Он рекурсивно делит массив на две половины, а затем сливает в один отсортированный. Он использует дополнительную память, при этом является стабильным. Сложность его всегда равна O(n logn).



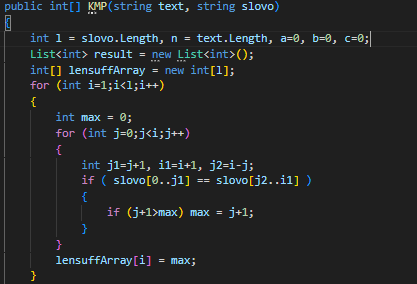
Binary Search (Бинарный поиск) – это алгоритм поиска в отсортированном массиве, что действует по принципу разделяй и властвуй. Он делит множество на 2 части таким образом, что искомый элемент попадает в одну из этих частей. После происходит разделение уже этой части и так до нахождения искомого значения. Он не использует дополнительную память, не является стабильным. Временная сложность зависит от массива – в лучшем случае О(1), в остальных же O(log n).

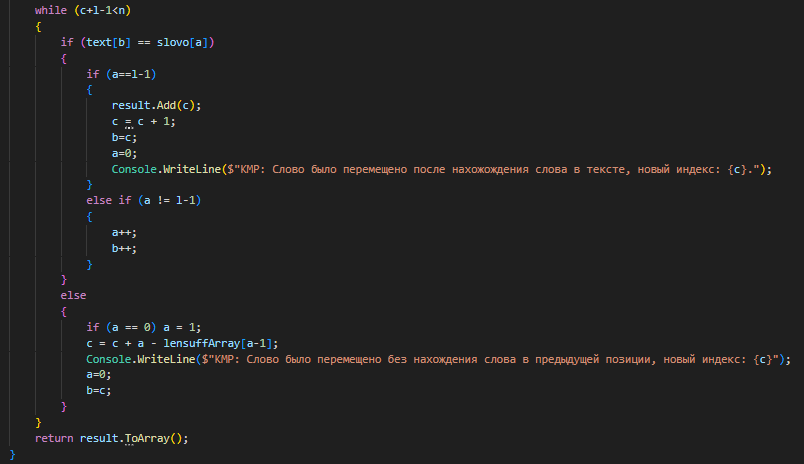


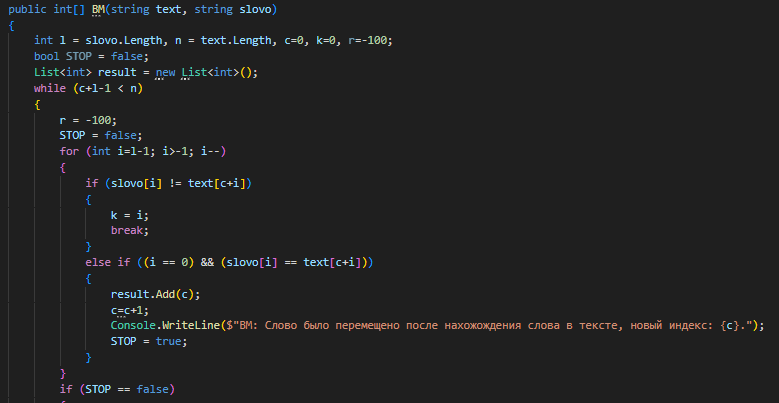
Interpolation Search (Интерполяционный поиск) – это алгоритм поиска в отсортированном массиве, что действует как бинарный поиск, однако учитывает распределение данных и делает предположение о местоположении искомого значения, то есть поиск как таковой середины происходит по формуле. Он не требует дополнительной памяти, является не стабильным. Временная сложность делится в зависимости от массива – в лучшем случае О(1), в среднем О(log log n), в худшем О(n). Относительно бинарного поиска этот алгоритм работает лучше, когда данные распределены равномерно.

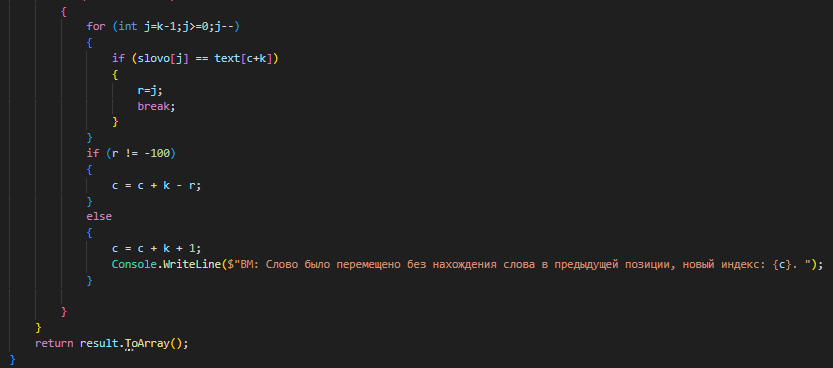


KMP (Алгоритм Кнута-Мориса-Пратта) – это алгоритм поиска слова в тексте, в основе которого лежит идея вычисления информации о том, сколько символов можно пропустить при несовпадении. Сначала вычисляется префикс-функция по положению символа в слове, что не совпал в тексте. Затем же вычисляется то количество символов, которое можно пропустить. Его сложность равна O(n+m), где n – длина текста, а m – длина искомого слова.





Boyer-Moore (Алгоритм Боуера-Мура) – это алгоритм поиска слова в тексте, что основывается на том, присутствует ли не совпавший символ из текста в слове. Если нет, то слово встаёт на позицию после данного символа. Если да, то слово переставляется так, чтобы не совпавший символ встал в ряд с самым правым символом текста, что является идентичным. Временная сложность в худшем случае равна O(n+m), в лучшем O(n). 



Эксперименты

Эксперименты проводились на процессоре i5-3230M с объёмом памяти 16 гигобайт, версия ОС: Windows 10, ключи компиляции установлены не были.

Результаты экспериментов приведены в таблицах ниже.

Бинарный поиск

|  |  |  |
| --- | --- | --- |
| № | Количество элементов в массиве | Время выполнения алгоритма, с |
| 1 | 10 000 | 0, 0002375 |
| 2 | 150 000 | 1,3E-06 |
| 3 | 1 000 000 | 1,2E-06 |

Интерполяционный поиск

|  |  |  |
| --- | --- | --- |
| № | Количество элементов в массиве | Время выполнения алгоритма, с |
| 1 | 10 000 | 0, 0002505 |
| 2 | 150 000 | 4E-07 |
| 3 | 1 000 000 | 2E-07 |

Radix sort

|  |  |  |
| --- | --- | --- |
| № | Количество элементов в массиве | Время выполнения алгоритма, с |
| 1 | 10 000 | 0,0118461 |
| 2 | 150 000 | 0,0285914 |
| 3 | 1 000 000 | 0,1790614 |

Shell sort

|  |  |  |
| --- | --- | --- |
| № | Количество элементов в массиве | Время выполнения алгоритма, с |
| 1 | 10 000 | 0, 0013622 |
| 2 | 150 000 | 0, 0249714 |
| 3 | 1 000 000 | 0, 2150707 |

Merge sort

|  |  |  |
| --- | --- | --- |
| № | Количество элементов в массиве | Время выполнения алгоритма, с |
| 1 | 10 000 | 0, 0028544 |
| 2 | 150 000 | 0, 0451059 |
| 3 | 1 000 000 | 0, 2612198 |

Алгоритм Кнута, Мориса и Пратта

|  |  |  |
| --- | --- | --- |
| № | Количество элементов в массиве | Время выполнения алгоритма, с |
| 1 | 100 | 0,001184 |
| 2 | 1 500 | 0,0012064 |
| 3 | 10 000 | 0,0013792 |

Алгоритм Боуера и Мура

|  |  |  |
| --- | --- | --- |
| № | Количество элементов в массиве | Время выполнения алгоритма, с |
| 1 | 100 | 0,0004843 |
| 2 | 1 500 | 0,0004796 |
| 3 | 10 000 | 0,000887 |

Как мы видим по результатам, немалое значение имеет наполнение самого массива, так как даже массивы большего наполнения могут выполнять быстрее. Однако в тестах между разными алгоритмами с одним количеством элементов использовались одинаковые массивы, поэтому мы можем сравнить их. Как мы видим, интерполяционный поиск в большинстве случаев оказывается быстрее бинарного, однако лишь за счёт того, что массив случайно заполняется примерно равномерно. Результаты же сравнения сортировочных алгоритмов показывает, что на малых массивах быстрее всех себя показывает Shell sort, а с крупными быстрее всего справляется Merge sort. Однако последний при этом выполняется при помощи рекурсии, что в свою очередь увеличивает требуемую память. Radix sort показала наихудший результат, однако это и не удивительно – свою максимальную эффективность он показывает лишь на числах небольших разрядов. В сравнении поиска в тексте наилучшие результаты показал алгоритм Боуера и Мура.