Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

по курсу “Логика и основа алгоритмизации в ИЗ”

на тему “ Оценка времени выполнения программ”

Выполнил студент группы 21ВВ3:

Ванюшин И.А.

Приняли:

Митрохин М.А., Юрова О.В.

Пенза 2022

**Название**

Оценка времени выполнения программ

**Цель работы:** Оценить время выполнения программы, выполняя лабораторные указания 1-2.

**Лабораторное задание:**

Задание 1:

1. Вычислить порядок сложности программы (*О*-символику).
2. Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 100, 200, 400, 1000, 2000, 4000, 10000.
3. Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.

Задание 2:

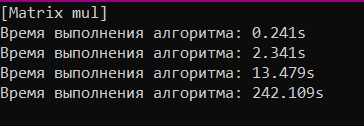
1. Оценить время работы каждого из реализованных алгоритмов на случайном наборе значений массива.
2. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.
3. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.
4. Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.
5. Оценить время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

**Перемножение двух матриц**

Сложность выполнения алгоритма О(n3)

Алгоритм содержит вложенные циклы, которые выполняют полный проход от 0 до n.

Таким образом, в сложность добавляется n от корневого цикла, n от вложенного в него цикла и n от вложенного цикла в последний.



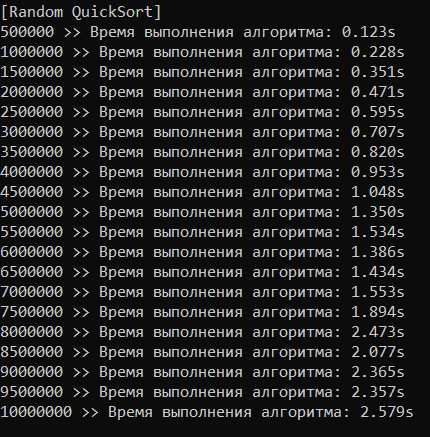
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Размер матрицы | 100 | 200 | 400 | 1000 | 2000 | 4000 | 10000 |
| Время выполнения алгоритма, c | 0.221 | 2.206 | 15.838 | 233.680 | ~1869.440 | ~14955.520 | ~233680.000 |

**Быстрая сортировка(QuickSort)**

Для массива случайных чисел

Сложность алгоритма в таком массиве О(nlog n) и стремится к линейной.

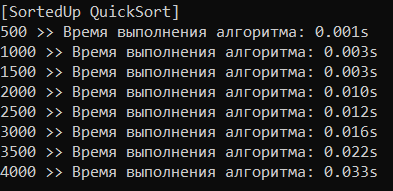
Сортируются огромные объемы данных за очень короткое время.



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | 1000000 | 2000000 | 3000000 | 4000000 | 5000000 | 6000000 | 7000000 | 8000000 | 9000000 |
| Время сортировки | 0.306 | 0.446 | 0.627 | 0.847 | 1.054 | 1.281 | 1.471 | 1.705 | 1.934 |

Для отсортированного по возрастанию массива чисел

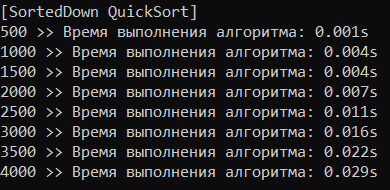
Сложность алгоритма в таком массиве ухудшается до О(n2). Помимо этого, глубина рекурсии в неоптимизированном алгоритме сортировки становится большой уже на малых значениях объема данных.



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | 500 | 1000 | 1500 | 2000 | 2500 | 3000 | 3500 | 4000 |
| Время сортировки | 0.001 | 0.002 | 0.004 | 0.007 | 0.009 | 0.015 | 0.020 | 0.028 |

Для отсортированного по убыванию массива чисел

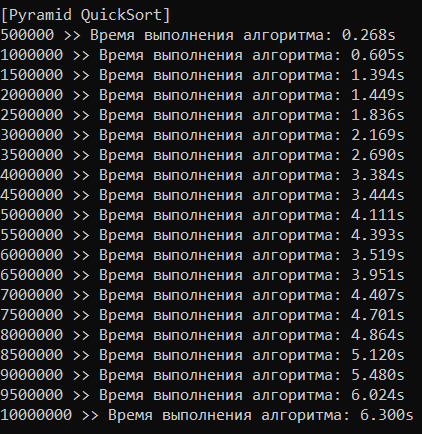
Сложность сортировки идентична для отсортированных по убыванию и по возрастанию массивов чисел.



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | 500 | 1000 | 1500 | 2000 | 2500 | 3000 | 3500 | 4000 |
| Время сортировки | 0.001 | 0.002 | 0.004 | 0.006 | 0.010 | 0.014 | 0.020 | 0.026 |

Для массива чисел, с отсортированной по возрастанию половиной и отсортированной по убыванию половиной

Сложность алгоритма в данном случае составляет О(nlog n), однако время выполнения сортировки увеличилось примерно в 2 раза относительно сортировки массива случайных чисел. Превышение глубины рекурсии на больших объемах данных не наблюдалось.

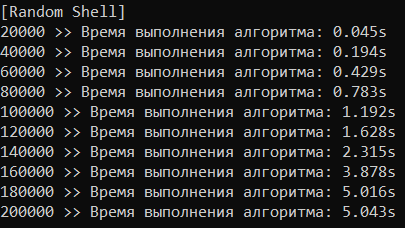


|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | 1000000 | 2000000 | 3000000 | 4000000 | 5000000 | 6000000 | 7000000 | 8000000 | 9000000 |
| Время сортировки | 0.592 | 1.018 | 1.562 | 2.187 | 2.869 | 3.376 | 4.089 | 4.792 | 5.287 |

**Сортировка Шелла**

Для массива случайных чисел

Сложность алгоритма в таком массиве составляет О(nlog n).  
Однако QuickSort намного быстрее сортирует бо́льшие объемы данных.

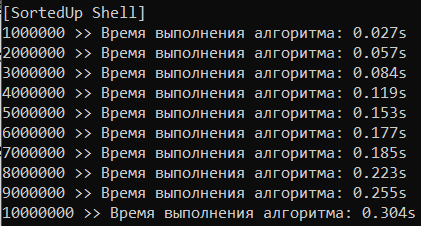


|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | 20000 | 40000 | 60000 | 80000 | 100000 | 120000 | 140000 | 160000 | 180000 |
| Время сортировки | 0.049 | 0.232 | 0.560 | 0.908 | 1.090 | 1.568 | 2.128 | 2.811 | 3.560 |

Для отсортированного по возрастанию массива чисел

Сложность алгоритма в таком массиве стремится к линейной, и составляет О(n).

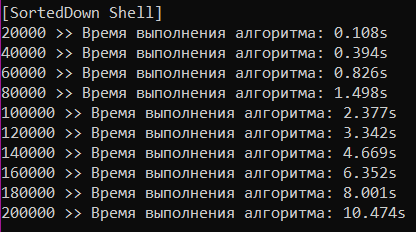
Это делает сортировку Шелла более выгодной для сортировки возрастающего массива чисел.



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | 1000000 | 2000000 | 3000000 | 4000000 | 5000000 | 6000000 | 7000000 | 8000000 | 9000000 | 10000000 |
| Время сортировки | 0.0032 | 0.0058 | 0.102 | 0.117 | 0.166 | 0.205 | 0.226 | 0.278 | 0.313 | 0.352 |

Для отсортированного по убыванию массива чисел

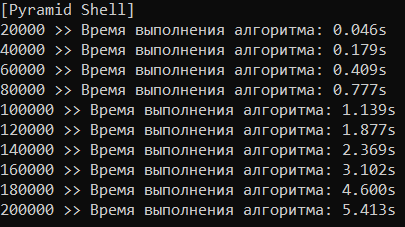
Сложность алгоритма в таком массиве ухудшается до О(n2). Однако по сравнению с Быстрой сортировкой глубина рекурсии не может стать слишком большой.



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | 20000 | 40000 | 60000 | 80000 | 100000 | 120000 | 140000 | 160000 | 180000 |
| Время сортировки | 0.109 | 0.446 | 0.868 | 1.688 | 2.591 | 3.686 | 4.865 | 6.752 | 8.463 |

Для массива чисел, с отсортированной по возрастанию половиной и отсортированной по убыванию половиной

Сложность алгоритма в данном случае составляет О(nlog n), однако время выполнения сортировки увеличилось примерно в 2 раза. Превышение глубины рекурсии на больших объемах данных не наблюдалось.

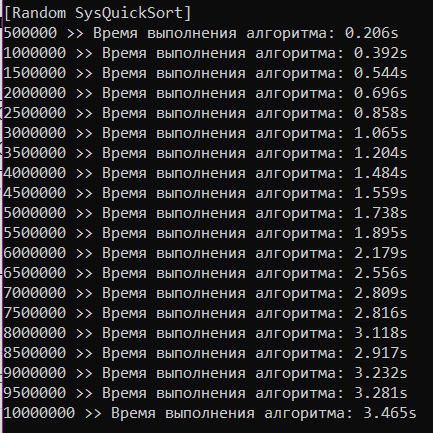


|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | 20000 | 40000 | 60000 | 80000 | 100000 | 120000 | 140000 | 160000 | 180000 |
| Время сортировки | 0.046 | 0.179 | 0.463 | 1.890 | 1.404 | 2.547 | 3.522 | 4.367 | 5.451 |

**Системная Быстрая сортировка(QSort)**

Для массива случайных чисел

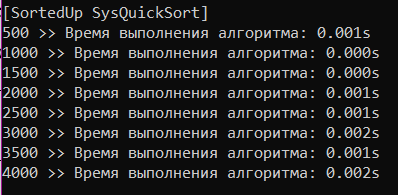
Сложность алгоритма в таком массиве О(nlog n), но в угоду множественным проверкам исключений, время сортировки значительно выше, чем у написанным алгоритмом.



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | 1000000 | 2000000 | 3000000 | 4000000 | 5000000 | 6000000 | 7000000 | 8000000 | 9000000 |
| Время сортировки | 0.306 | 0.446 | 0.627 | 0.847 | 1.054 | 1.281 | 1.471 | 1.705 | 1.934 |

Для отсортированного по возрастанию массива чисел

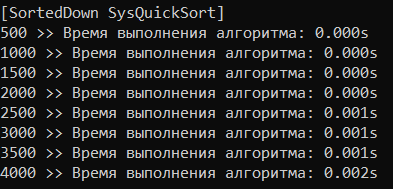
Сложность алгоритма в таком массиве становится О(n), поскольку системная быстрая сортировка проверяет массив на сортированность.



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | 500 | 1000 | 1500 | 2000 | 2500 | 3000 | 3500 | 4000 |
| Время сортировки | 0.001 | 0.000 | 0.000 | 0.001 | 0.002 | 0.001 | 0.001 | 0.001 |

Для отсортированного по убыванию массива чисел

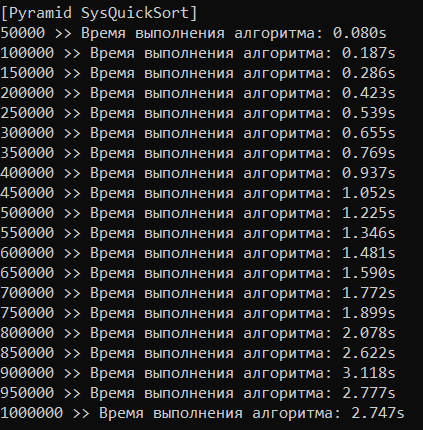
Сложность сортировки идентична для отсортированных по убыванию и по возрастанию массивов чисел.



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | 500 | 1000 | 1500 | 2000 | 2500 | 3000 | 3500 | 4000 |
| Время сортировки | 0.000 | 0.000 | 0.000 | 0.000 | 0.001 | 0.001 | 0.001 | 0.002 |

Для массива чисел, с отсортированной по возрастанию половиной и отсортированной по убыванию половиной

Сложность алгоритма в данном случае составляет О(nlog n), однако время выполнения сортировки увеличилось примерно в 2 раза.



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | 100000 | 200000 | 300000 | 400000 | 500000 | 600000 | 700000 | 800000 | 900000 |
| Время сортировки | 0.187 | 0.423 | 0.655 | 0.937 | 1.225 | 1.481 | 1.772 | 2.078 | 3.118 |

**Вывод**

Сортировка Шелла является более стабильной по сложности алгоритма к различным наборам данных чем Быстрая сортировка, следовательно, ее выгодно использовать на небольших объемах данных

Быстрая сортировка может обрабатывать большие объемы данных за небольшой промежуток времени, однако имеет большой ряд исключений и нестабильное время работы, зависящее от набора данных, следовательно, ее выгодно использовать на больших объемах данных при условии их неотсортированности.

Системная Быстрая сортировка имеет все те же преимущества, что имеет написанный алгоритм. Имеет увеличенное время работы алгоритма, однако все возможные исключения обрабатываются. Это оптимальный вариант как для малых, так и для больших наборов данных.

Были выполнены задания 1-2 лабораторной работы №2 и сравнены между собой время и результаты работы данных алгоритмов.