ОТЧЕТ

По лабораторной работе №3

По курсу «Анализ алгоритмов»

Тема: «Алгоритмы сортировки»

|  |  |
| --- | --- |
|  | **Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  «Московский государственный технический университет  имени Н.Э. Баумана  (национальный исследовательский университет)»  (МГТУ им. Н.Э. Баумана) |

Студент: Зейналов З. Г.

Группа: ИУ7-51

Преподаватель: Волкова Л.Л.

Москва, 2019г.

Оглавление

[Введение 3](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736334)

[Задачи работы: 4](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736335)

[1. Аналитическая часть 5](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736336)

[1.1 Описание алгоритмов 5](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736337)

[1.2 Алгоритм умножения Копперсмита-Винограда 6](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736338)

[2. Конструкторская часть 7](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736339)

[2.1 Разработка алгоритмов 8](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736340)

[2.2 Расчет трудоемкости 14](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736341)

[3 Технологическая часть 15](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736342)

[3.1 Требования к программному обеспечению 15](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736343)

[3.2 Средства реализации 15](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736344)

[3.3 Листинг кода 16](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736345)

[4 Экспериментальная часть 19](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736346)

[4.1 Примеры работы 19](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736347)

[4.2 Результаты тестирования 20](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736348)

[4.3 Постановка эксперимента по замеру времени 20](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736349)

[4.4 Сравнительный анализ на материале экспериментальных данных 21](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736350)

[Заключение 23](file:///C:\Users\зейнал\Desktop\отчет%20по%20АА%202.docx#_Toc22736351)

# Введение

**Алгоритм сортировки** — это [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) для упорядочивания элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма. Наша современная жизнь вплотную пересекается с большим объемом данных. Именно поэтому задача сортировки имеет высокое значение для упорядочивания данных и получения результата в удобном виде.

**Целью** данной лабораторной работы является исследование существующих алгоритмов сортировки массива и исследовать их трудоемкость.

**Задачи работы:**

Задачами данной лабораторной работы являются:

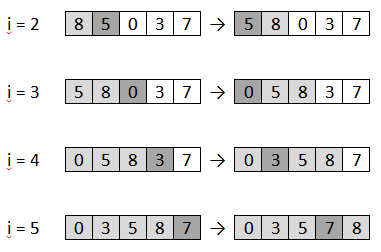
1. Реализовать 3 выбранных алгоритма сортировки
2. Рассчитать трудоемкость одного из алгоритмов, привести рассчитанное значение трудоемкости для двух оставшихся алгоритмов из литературы.
3. Провести экспериментальный анализ по замерам времени.
4. **Аналитическая часть**

В данном разделе будут представлены описания алгоритмов, формулы и оценки сложностей алгоритмов

### Описание алгоритмов

Алгоритм сортировки вставками:

Основная идея данного алгоритма заключается в том, что по левую сторону от рассматриваемого элемента, находится уже отсортированный массив. Для этого элемента выбирается место в этой части массива.



*Рисунок 1 - Демонстрация работы алгоритма*

Алгоритм «Гномья сортировка»

Гномья сортировка впервые была предложена 2 октября 2000 года Хамидом Сарбази-Азадом (Hamid Sarbazi-Azad). Он назвал ее «Глупая сортировка, простейший алгоритм сортировки всего с одним циклом…». И действительно, глупый этот метод или нет, но в нем задействован, никак в большинстве сортировок – два или более циклов, а только один. Позже, голландский ученый Дик Грун, на страницах одной из своих книг, привел для гномьей сортировки следующую аналогию:

«Гномья сортировка основана на технике, используемой обыкновенным голландским садовым гномом. Вот как садовый гном сортирует ряд цветочных горшков. По существу, он смотрит на два соседних цветочных горшка, если они находятся в правильном порядке, то он переходит на один горшок вперед, иначе он меняет их местами и возвращается на один горшок назад. Граничные условия: если позади нет ни одного горшка – он шагает вперед, а если нет следующего горшка, тогда он закончил». [1]

Т.е алгоритм работы заключается в том, что у нас существует некий указатель, способный ходить как в одну сторону, так в другую. Таким образом, если мы возьмем 2 элемента массива А: A[i - 1] и A[i], происходит их сравнение, в случае, если они стоят на своих местах (упорядочены), то счетчик i увеличиваем, в обратном случае меняем их местами и уменьшаем счетчик, затем проделываем то же самое.

Алгоритм быстрой сортировки

QuickSort является существенно улучшенным вариантом алгоритма сортировки с помощью прямого обмена (его варианты известны как «Пузырьковая сортировка» и «Шейкерная сортировка»), известного в том числе своей низкой эффективностью. Принципиальное отличие состоит в том, что в первую очередь производятся перестановки на наибольшем возможном расстоянии и после каждого прохода элементы делятся на две независимые группы. Любопытный факт: улучшение самого неэффективного прямого метода сортировки дало в результате один из наиболее эффективных улучшенных методов.

Общая идея алгоритма состоит в следующем :

* Выбрать из массива элемент, называемый опорным. Это может быть любой из элементов массива. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность.
* Сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующих друг за другом: «элементы меньшие опорного», «равные» и «большие».
* Для отрезков «меньших» и «больших» значений выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы [2].

Мы же в нашей реализации будем использовать 2 разделения на «большие либо равные» и на «меньшие».

### Описание модели вычислений

Выберем следующую модель вычислений:

1. Трудоемкость базовых операций  
   Операции - имеют стоимость 1.
2. Трудоемкость условного перехода  
   Условный переход имеет стоимость 0, при этом оцениваем расчет условия:

Трудоемкость цикла   
,  
где N – число повторений цикла

## Конструкторская часть

В данном разделе будут размещены схемы алгоритмов и сравнительный анализ рекурсивной и не рекурсивной реализаций.

### Разработка алгоритмов

На рисунках 1 – 11 приведены схемы алгоритмов, демонстрирующие работу







* 1. **Расчет трудоемкости**

Проведем расчёт трудоемкости для алгоритма сортировки вставками:

* Лучший случай, когда массив уже отсортирован
* Худший случай

Алгоритм Гномьей сортировки  
Алгоритм концептуально простой, не требует вложенных циклов. Время работы . На практике алгоритм может работать так же быстро, как и сортировка вставками.

Алгоритм быстрой сортировки

В наиболее сбалансированном варианте при каждой операции разделения массив делится на две одинаковые (плюс-минус один элемент) части, следовательно, максимальная глубина рекурсии, при которой размеры обрабатываемых под массивов достигнут 1, составит. В результате количество сравнений, совершаемых быстрой сортировкой, было бы равно значению рекурсивного выражения , что даёт общую сложность алгоритма .

## Технологическая часть

В данном разделе будут приведены Требования к программному обеспечению, средства реализации, листинг кода и примеры тестирования.

### Требования к программному обеспечению

На вход подаются 2 целочисленные матрицы, на выходе должен возвращаться результат их умножения либо сообщение о невозможности их умножения.

### Средства реализации

В качестве языка программирования был выбран С++ в связи с его широким функционалом и быстротой работы, а так же благодаря привычному для меня синтаксису и семантики языка. Среда разработки - Qt. Время работы процессора замеряется с помощью функции std::chrono().

### Листинг кода

Ниже будут представлены реализации алгоритмов

Листинг 1. Реализация алгоритма сортировки вставками

void arraySort::**insertionSort**()

{

*for* (size\_t i = 0; i < *this*->arraySize - 1; i++)

{

int key = i + 1;

int tmp = *this*->Array[key];

size\_t j = i + 1;

*while* (j > 0 && tmp < *this*->Array[j - 1])

{

*this*->Array[j] = *this*->Array[j - 1];

key = j - 1;

j = j - 1;

}

*this*->Array[key] = tmp;

}

}

void arraySort::**GnomeSort**(size\_t i, size\_t j)

{

*if* (i > *this*->arraySize || j > *this*->arraySize)

{

*return*;

}

*while* (i < *this*->arraySize)

{

*if* (*this*->Array[i - 1] <= *this*->Array[i])

{

i = j;

j++;

}

*else*

{

int t = *this*->Array[i];

*this*->Array[i] = *this*->Array[i - 1];

*this*->Array[i - 1] = t;

i--;

*if* (i == 0)

{

i = j;

j++;

}

}

}

}

Листинг 2. Реализация алгоритма Гномьей сортировки

void arraySort::**quickSort**(size\_t first, size\_t last)

{

int mid, count;

int f = first, l = last;

mid = *this*->Array[(f + l) / 2]; *//вычисление* *опорного* *элемента*

*do*

{

*while* (*this*->Array[f] < mid)

f++;

*while* (*this*->Array[l] > mid)

l--;

*if* (f<=l) *//перестановка* *элементов*

{

count = *this*->Array[f];

*this*->Array[f] = *this*->Array[l];

*this*->Array[l] = count;

f++;

l--;

}

} *while* (f < l);

*if* (first < l) quickSort(first, l);

*if* (f < last) quickSort(f, last);

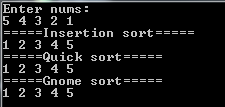
}

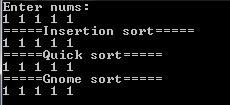
Листинг 3. Реализация алгоритма быстрой сортировки

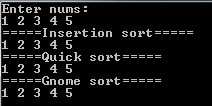
## Экспериментальная часть

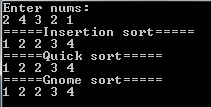
В данном разделе будут приведены примеры работы программы, постановка эксперимента и сравнительный анализ алгоритмов на основе экспериментальных данных.

### Примеры работы









Программа успешно прошла все тестовые случаи, все полученные результаты совпали с ожидаемыми.

### Постановка эксперимента по замеру времени

Для произведения замеров времени выполнения реализаций алгоритмов будет использована следующая формула , где t – время выполнения, N – количество замеров. Неоднократное измерение времени необходимо для построения более гладкого графика.

Количество замеров будет взято равным 100.

Тестирование будет проведено на одинаковых входных данных. 1) массивы размерностями от 100х100 до 800х800 с шагом 100.

### Сравнительный анализ на материале экспериментальных данных

Ниже приведены графики зависимости временных затрат ( в тиках процессора) от размеров входных данных.

Рисунок 13. Алгоритмы умножения матриц (четных размеров)

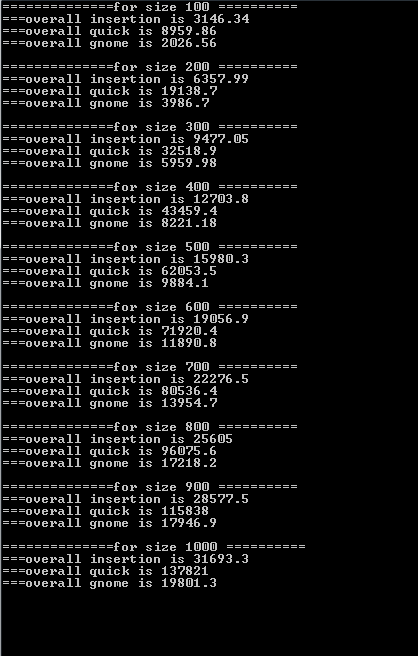
На Рисунке 12 видно, что оптимизированный алгоритм Винограда превосходит стандартный алгоритм на 30 % и неоптимизированный алгоритм почти на 45%

Рисунок 14. Алгоритмы умножения матриц (нечетных размеров)

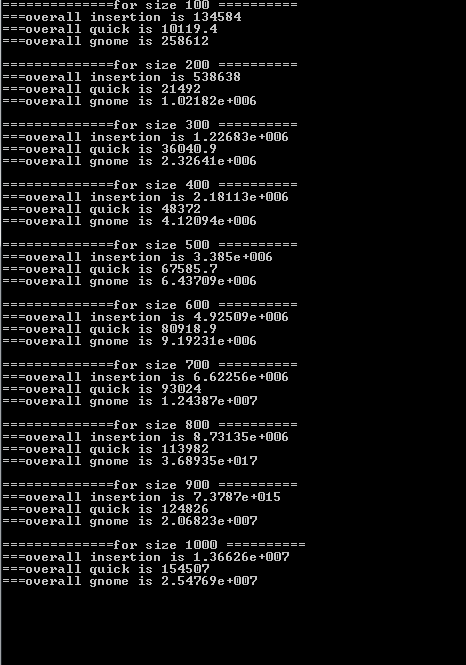
На рисунке 13 видно, что Оптимизированный алгоритм не потерял свое превосходство, стандартный алгоритм не изменил своего времени работы, а алгоритм Винограда стал работать медленнее .



Рандом



**Инкризинг**



**декризинг**

# Заключение

В ходе работы были изучены и реализованы алгоритмы стандартного умножения матриц, алгоритма Винограда и оптимизированного алгоритма Винограда. Был проведен сравнительный анализ перечисленных алгоритмов по трудоемкости и экспериментально выявлена временная разница работы алгоритмов. Классический алгоритм в неоптимизированном виде является более эффективным чем алгоритм винограда, однако после ряда оптимизаций, алгоритм Винограда становится значительно быстрее классического.

Литература   
<http://kvodo.ru/gnome-sorting.html> - здесь описание алгоритмов сортировки история и реализация

<https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0_%D0%B2%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0%D0%BC%D0%B8>

<https://ru.wikipedia.org/wiki/%D0%91%D1%8B%D1%81%D1%82%D1%80%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0#%D0%9E%D1%86%D0%B5%D0%BD%D0%BA%D0%B0_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B0>

<https://ru.wikipedia.org/wiki/%D0%93%D0%BD%D0%BE%D0%BC%D1%8C%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0>