Министерство науки и высшего образования 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)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»
ДИСЦИПЛИНА «Анализ алгоритмов»

Лабораторная работа № 3

Дисциплина Анализ алгоритмов
Тема Анализ алгоритмов сортировки массовов
Студент Боренко Анастасия
Группа ИУ7-52Б
Оценка (баллы) _____
Преподаватель Волкова Л.Л.

Москва — 2021г.

Содержание

Введение **3**

1 Аналитическая часть **4**

1.1 Сортировка пузырьком 4

1.2 Сортировка выбором 4

1.3 Сортировка вставками 4

1.4 Вывод аналитической части 5

2 Конструкторская часть **6**

2.1 Схемы алгоритмов 6

2.2 Трудоемкость алгоритмов 8

2.3 Сравнительный анализ трудоемкостей алгоритмов 9

2.4 Структуры данных 9

2.5 Тестирование 10

2.6 Вывод конструкторской части 10

3 Технологическая часть **11**

3.1 Требования к ПО 11

3.2 Выбор языка программирования 11

3.3 Структуры данных 11

3.4 Реализация алгоритмов 12

3.5 Тестирование 13

3.6 Вывод технологической части 13

4 Экспериментальная часть **14**

4.1 Примеры работы 14

4.2 Замеры времени 16
4.3 Сравнительный анализ алгоритмов 17
4.4 Вывод экспериментальной части 18

5 Заключение **19**

Список литературы **20**

Введение

В данной лабораторной работе рассматриваются алгоритмы, которые известны своей простотой и необходимостью, алгоритмы сортировки. Сортировать можно что угодно: числа в массиве, слова в словаре, вещи в магазине и т.д. Наиболее частые сферы применения сортировок в программировании

1. поиск в массиве по ключу
2. группировка элементов по одинаковым значениям какого-либо признака
3. сравнение двух и более массивов на наличие одинаковых элементов

Целью данной лабораторной являются изучение метода динамического программирования на материале алгоритмов сортировки. Задачами данной лабораторной являются:

1. изучение алгоритмов сортировки пузырьком, вставками, выбором;
2. определение трудоемкостей исследуемых алгоритмов;
3. реализация указанных алгоритмов;
4. сравнительный анализ исследуемых алгоритмов;
5. экспериментальное подтверждение различий в трудоемкости алгоритмов с указанием лучшего и худшего случаев;
6. описание и обоснование полученных результатов в отчете о выполненной лабораторной работе, выполненного как расчетно-пояснительная записка к работе.

1. Аналитическая часть

1.1 Сортировка пузырьком

Идея алгоритма заключается в следующем: шаг сортировки состоит в проходе от начала к концу по массиву. По пути просматриваются пары соседних элементов. Если элементы некоторой пары находятся в неправильном порядке, то меняем их местами. После нулевого прохода в начале массива окажется самый "легкий"элемент - "пузырек". Следующий проход делается со второго элемента, всплывает 2 "пузырек". И так повторяем, пока элементы не закончатся. После этого получаем отсортированный массив.

1.2 Сортировка выбором

Шаг сортировки - это поиск максимального элемента массива. В первый проход найденный максимум меняется местами с последним элементом. Во второй проход поиск производится до предпоследнего элемента, и найденный элемент меняется местами с предпоследним. И далее, каждый раз перебирается на один элемент меньше и результат ставится на место с индексом на единицу меньше. Эти действия повторяются до тех пор как массив не закончится.

1.3 Сортировка вставками

Массив перебирается от начала к концу и каждый элемент обрабатывается по очереди. Слева от очередного элемента будет находиться отсортированная часть алгоритма, очередной элемент добавляется в нее так, чтобы не нарушить отсортированность этой части. В отсортированной части массива ищется точка вставки для очередного

элемента. Сам элемент отправляется в буфер, в результате чего в массиве появляется свободная ячейка — это позволяет сдвинуть элементы и освободить точку вставки.

1.4 Вывод аналитической части

В данной работе стоит задача реализации следующих алгоритмов: изучение алгоритмов сортировки пузырьком, вставками, выбором. Необходимо сравнить алгоритмы умножения матриц по эффективности по времени.

2. Конструкторская часть

2.1 Схемы алгоритмов

2.1.1 Схема стандартного алгоритма умножения матриц

На рисунке 2.1 показана схема алгоритма сортировки пузырьком.

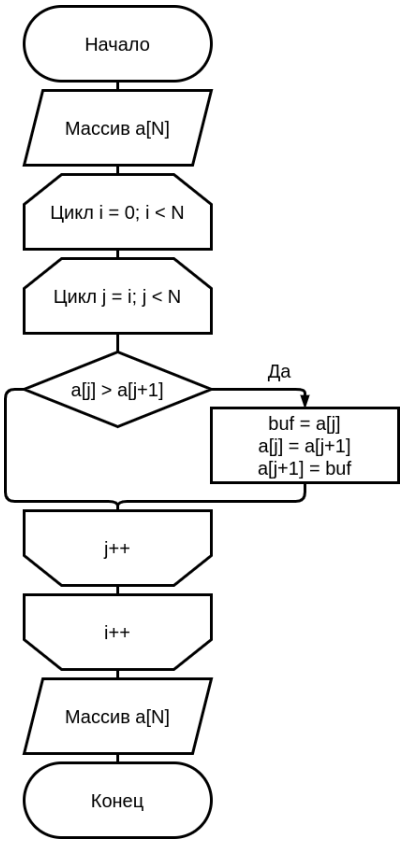

Рис. 2.1: Схема алгоритма сортировки пузырьком

На рисунке 2.2 показана схема алгоритма сортировки выбором.

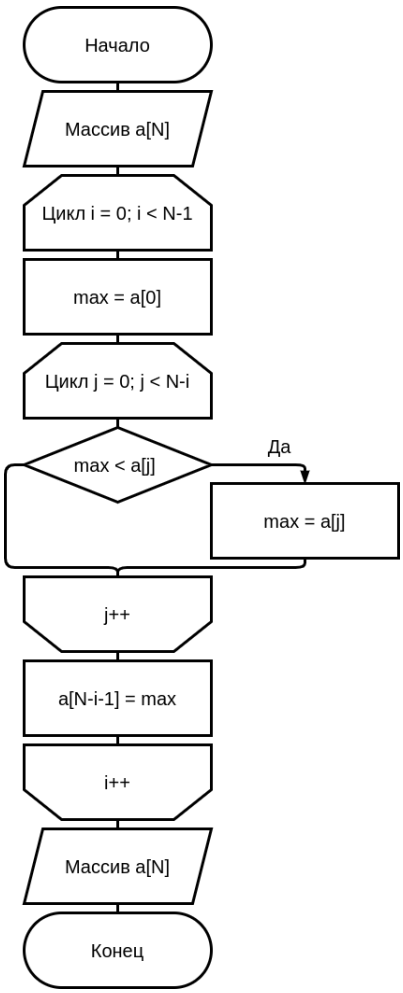

Рис. 2.2: Схема алгоритма сортировки выбором

На рисунке 2.3 показана схема алгоритма сортировки вставками.

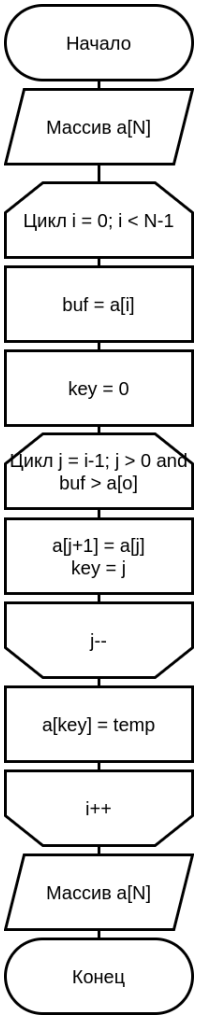

Рис. 2.3: Схема алгоритма сортировки вставками

2.2 Трудоемкость алгоритмов

2.2.1 Трудоемкость алгоритма сортировки пузырьком

Трудоемкость в лучшем случае $O(n^2)$:

$f = 2 + N \cdot (2 + 2 + N \cdot (2 + 3)) = 5 \cdot N^2 + 4 \cdot N + 2$

Трудоемкость в худшем случае $O(n^2)$:

$f = 2 + N \cdot (2 + 2 + N \cdot (2 + 3 + 9)) = 14 \cdot N^2 + 4 \cdot N + 2$

2.2.2 Трудоемкость алгоритма сортировки выбором

Трудоемкость в лучшем случае $O(n^2)$:

$f = 2 + N \cdot (2 + 2 + 4 + 2 + N \cdot (2 + 2)) = 4 \cdot N^2 + 10 \cdot N + 2$

Трудоемкость в худшем случае $O(n^2)$:

$f = 2 + N \cdot (2 + 2 + 4 + 2 + N \cdot (2 + 2 + 2) = 6 \cdot N^2 + 10 \cdot N + 2$

2.2.3 Трудоемкость алгоритма сортировки вставками

Трудоемкость в лучшем случае $O(n)$:

$f = 2 + N \cdot (2 + 3 + 2) = 7 \cdot N + 2$

Трудоемкость в худшем случае $O(n^2)$:

$f = 2 + N \cdot (2 + 3 + 2 + 2 + N \cdot (2 + 5) = 7 \cdot N^2 + 9 \cdot N + 2$

2.3 Сравнительный анализ трудоемкостей алгоритмов

Сравнив трудоемкости, можно сделать вывод, что в лучшем случае самым быстрым алгоритмом из рассматриваемых является алгоритм сортировки вставками, самый медленный - сортировка пузырьком. В худшем случае самый быстрый алгоритм - сортировка выбором, самый медленный - сортировка пузырьком.

2.4 Структуры данных

При реализации приведенных алгоритмов потребуется тип данных: массив.

2.5 Тестирование

2.5.1 Классы эквивалентности

Для алгоритма умножения матриц можно выделить следующие классы эквивалентности:

1. отсортированный по возрастанию массив
2. отсортированный по убыванию массив
3. случайный массив

2.5.2 Способы тестирования

При разработке программы удобно использовать следующие методы тестирования:

1. Модульные тесты
2. Функциональные тесты

2.6 Вывод конструкторской части

На основе данных, полученных в аналитическом разделе, были построены схемы используемых алгоритмов, выделены необходимые для реализации структуры данных и методы тестирования.

3. Технологическая часть

3.1 Требования к ПО

Требования к программному обеспечению:

1. Наличие меню для реализации выбора пользователя.
2. На вход подаются 2 матрица
3. Результат в зависимости от выбора пользователя:
   (a) отсортированный массив
   (b) замеры времени работы каждого из исследуемых алгоритмов

3.2 Выбор языка программирования

Был выбран язык Go, поскольку он удовлетворяет требованиям задания. Средой разработки выбрана Visual Studio Code.

3.3 Структуры данных

На листинге 3.1 представлено описание структуры массива.

Листинг 3.1: Структура массива

```
type vector_t struct{
  elem[] float32
  size int
}
```

3.4 Реализация алгоритмов

На листинге 3.2 представлена реализация алгоритма сортировки пузырьком.

Листинг 3.2: Реализация алгоритма сортировки пузырьком

```
func sort_bubble(array vector_t){
  for i:=1;i< array.size;i++{
    for j:=0;j < array.size - i;j++{
      if (array.elem[j + 1] < array.elem[j]){
        array.elem[j + 1], array.elem[j] = array.elem[j], array.elem[j + 1]
      }
    }
  }
}
```

На листинге 3.3 представлена реализация алгоритма сортировки выбором.

Листинг 3.3: Реализация алгоритма сортировки выбором

```
func sort_choice(array vector_t){
  for i:=0;i<array.size;i++{
    var min_i int = i

    for j:=i+1;j< array.size;j++{
      if(array.elem[j] <= array.elem[min_i]){
        min_i = j
      }
    }

    if(min_i !=i){
      array.elem[i], array.elem[min_i] = array.elem[min_i], array.elem[i]
    }
  }
}
```

На листинге 3.4 представлена реализация алгоритма сортировки вставками.

Листинг 3.4: Реализация алгоритма сортировки вставками

```
func sort_insert(array vector_t){
```

```
2   for i:=0;i<array.size;i++{
3     for j:= i;j > 0 && array.elem[j - 1] > array.elem[j];j--{
4       array.elem[j], array.elem[j - 1] = array.elem[j - 1], array.elem[j]
5     }
6   }
7 }
```

3.5 Тестирование

Модульные тесты

Таблица 3.1 – Тесты

№	Ввод	Вывод
1	1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10
2	10 9 8 7 6 5 4 3 2 1	1 2 3 4 5 6 7 8 9 10
3	1 4 2 7 10 5 9 3 8 6	1 2 3 4 5 6 7 8 9 10

3.6 Вывод технологической части

Были реализованы исследуемые алгоритмы, программа прошла тесты и удовлетворяет требованиям.

4. Экспериментальная часть

Оценка качества работы алгоритмов. Экспериментальное сравнение работы различных алгоритмов умножения матриц (зависимость времени выполнения от размерности матриц).

4.1 Примеры работы

На рисунках 4.1, 4.2, 4.3, 4.4, **??** показаны примеры работы.

```
                                                    MENU
        1.Manual testing
        2.Auto   testing
        Another choice is exit
1
Введите кол-во элементов: 10
Сгенерировать автоматически (1 - да, иначе - нет): 1


Введенный вектор:
Вектор [10]: 81 87 47 59 81 18 25 40 56 0

Результат сортировки пузырьком
Вектор [10]: 0 18 25 40 47 56 59 81 81 87

Результат сортировки вставками
Вектор [10]: 0 18 25 40 47 56 59 81 81 87

Результат сортировки выбором
Вектор [10]: 0 18 25 40 47 56 59 81 81 87
```

Рис. 4.1: Ручное тестирование: тест 1

```
                                                    MENU
        1.Manual testing
        2.Auto   testing
        Another choice is exit
1
Введите кол-во элементов: 10
Сгенерировать автоматически (1 - да, иначе - нет): 2
1 2 3 4 5 6 7 8 9 10

Введенный вектор:
Вектор [10]: 1 2 3 4 5 6 7 8 9 10

Результат сортировки пузырьком
Вектор [10]: 1 2 3 4 5 6 7 8 9 10

Результат сортировки вставками
Вектор [10]: 1 2 3 4 5 6 7 8 9 10

Результат сортировки выбором
Вектор [10]: 1 2 3 4 5 6 7 8 9 10
```

Рис. 4.2: Ручное тестирование: тест 2

```
                                                    MENU
        1.Manual testing
        2.Auto   testing
        Another choice is exit
1
Введите кол-во элементов: 10
Сгенерировать автоматически (1 - да, иначе - нет): 2
10 9 8 7 6 5 4 3 2 1

Введенный вектор:
Вектор [10]: 10 9 8 7 6 5 4 3 2 1

Результат сортировки пузырьком
Вектор [10]: 1 2 3 4 5 6 7 8 9 10

Результат сортировки вставками
Вектор [10]: 1 2 3 4 5 6 7 8 9 10

Результат сортировки выбором
Вектор [10]: 1 2 3 4 5 6 7 8 9 10
```

Рис. 4.3: Ручное тестирование: тест 3

```
                                    MENU
        1.Manual testing
        2.Auto   testing
        Another choice is exit
2
```

Длина	Пузырек			Вставки			Выбор		
	Сорт.мас.	Рев.мас.	Случ.мас.	Сорт.мас.	Рев.мас.	Случ.мас.	Сорт.мас.	Рев.мас.	Случ.мас.
100	8.058	10.858	9.870	1.194	11.527	7.980	13.803	13.592	13.592
200	30.584	43.015	59.479	2.219	42.019	25.252	45.586	47.246	51.703
300	61.789	93.633	152.868	1.661	94.169	52.925	98.629	100.954	114.772
400	107.337	171.254	255.958	2.177	169.601	90.660	170.760	173.767	198.360
500	167.297	262.905	370.636	2.757	267.706	140.483	262.756	270.617	301.413
600	239.705	378.091	524.190	2.923	382.799	194.270	379.318	388.728	429.846
700	323.615	513.112	686.957	2.689	513.659	253.153	508.726	521.604	572.319
800	419.755	673.690	855.826	2.972	674.651	338.073	675.799	682.956	736.832
900	529.077	847.265	1122.038	3.912	850.585	428.721	838.701	862.348	929.849
1000	647.144	1048.045	1324.571	4.184	1051.924	552.871	1037.872	1060.927	1140.303
1100	785.846	1274.001	1545.600	4.064	1304.461	674.602	1257.566	1286.826	1368.554
1200	930.754	1517.546	1884.788	4.486	1513.149	773.534	1495.610	1522.649	1627.841
1300	1088.599	1780.985	2176.689	4.583	1779.194	885.963	1743.549	1788.707	1898.314
1400	1268.485	2060.611	2496.537	4.776	2067.419	1019.679	2035.279	2075.916	2207.517
1500	1453.166	2369.067	2800.868	5.642	2369.207	1230.692	2331.465	2396.592	2548.044
1600	1853.952	2826.935	3251.986	5.946	2701.472	1364.365	2646.748	2707.327	2873.041
1700	1862.670	3051.830	3640.188	5.895	3036.905	1568.001	2974.353	3037.253	3210.430
1800	2090.658	3414.676	4108.832	6.038	3414.046	1696.835	3334.142	3413.555	3606.188
1900	2331.844	3812.733	4501.008	7.071	3797.205	1936.651	3720.101	3803.230	4019.092
2000	2570.320	4212.681	5044.008	6.651	4214.815	2149.102	4127.731	4224.770	4459.516

Рис. 4.4: Автотестирование

4.2 Замеры времени

На рисунках 4.5, 4.6, и 4.7 показаны графические результаты сравнения исследуемых алгоритмов по времени.

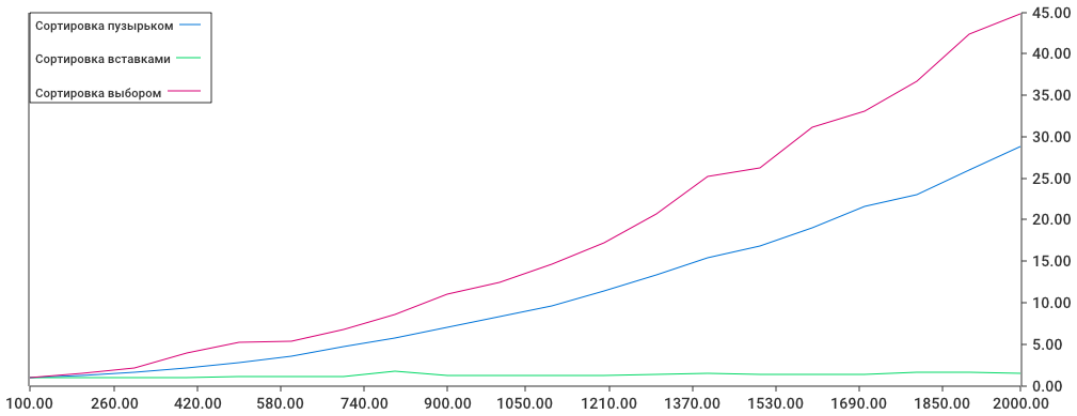

Рис. 4.5: Сравнение 3 исследуемых алгоритмов по времени (массив отсортирован)

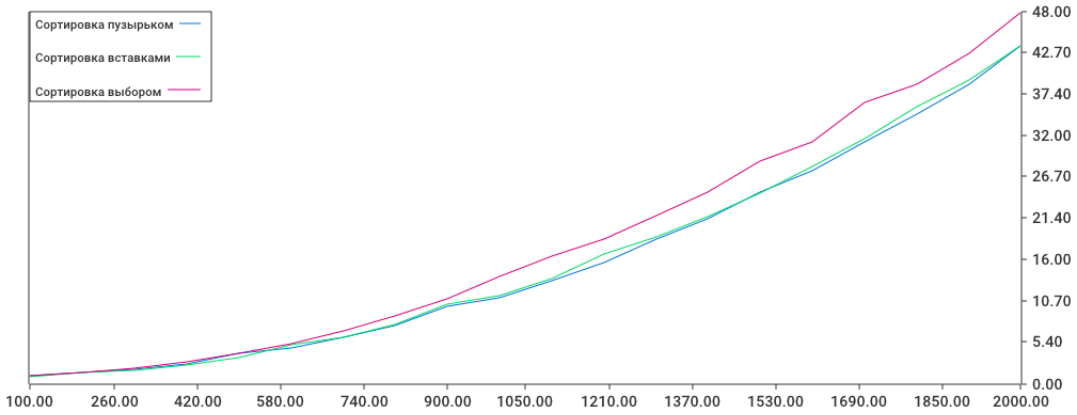

Рис. 4.6: Сравнение 3 исследуемых алгоритмов по времени (массив отсортирован в обратном порядке)

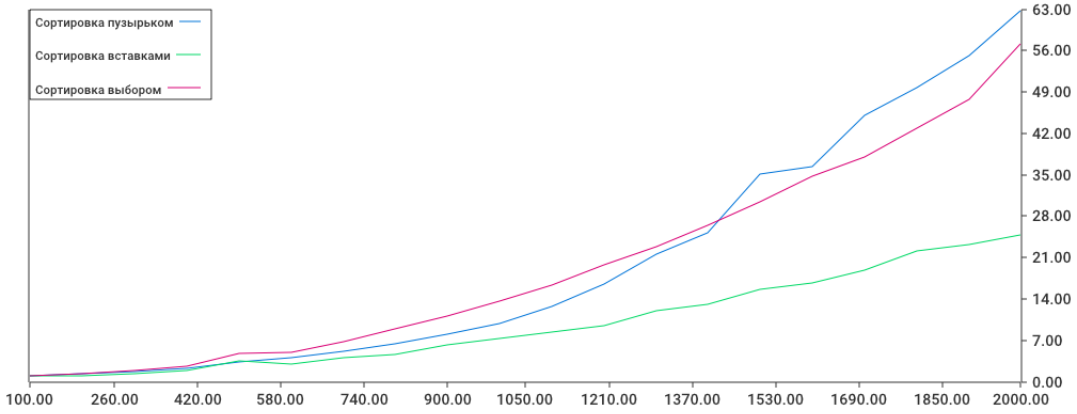

Рис. 4.7: Сравнение 3 исследуемых алгоритмов по времени (массив сгенерирован случайно

4.3 Сравнительный анализ алгоритмов

Были протестированы алгоритмы сортировки. Самый эффективный из исследуемых алгоритмов по времени на отсортированном массиве - сортировка вставками, самый медленный - сортировка пузырьком. На отсортированном в обратном порядке массиве сортировки вставками и пузырьком приблизительно одинаковы по скорости и они быстрее сортировки выбором, то есть самый медленный алгоритм - сортировка пузырьком. Самый эффективный из исследуемых алгоритмов по времени на случайном массиве - сортировка вставками, при длине массива меньше 1400 сортировка

пузырьком быстрее сортировки выбором, при большем количестве эллементов сортировка пузырьком медленее сортировки выбором.

4.4 Вывод экспериментальной части

Таким образом, подтвердилось предположение, что самый эффективный алгоритм в лучшем случае алгоритм - сортировка вставками, а самый медленный сортировка пузырьком. Предположение о худшем случае оказалось неверно. Быстрыми алгоритмами в этом случае являются сортировки вставками и пузырьком, самый медленный - сортировка выбором.

5. Заключение

В данной работе были изучены алгоритмы сортировки массива: пузырьком, выбором, вставками. Получены практические навыки реализации исследуемых алгоритмов. Была подсчитана трудоемкость исследуемых алгоритмов. Проведён сравнительный анализ алгоритмов по времени и трудоемкости. Экспериментально подтверждены различия в эффективности алгоритмов с указанием лучших и худших случаев. Цель работы достигнута, решены поставленные задачи. Получены практические навыки реализации алгоритмов сортировки, а также проведена исследовательская работа вычислении трудоемкости алгоритмов.

Список литературы

[1] Gratzer George A. More Math Into LaTeX. 4th изд. Boston: Birkhauser, 2007.

[2] Go Documentation [Электронный ресурс]. Режим доступа: https://go.dev/doc/. Дата обращения: 20.09.2020.

[3] Debian – универсальная операционная система [Электронный ресурс]. Режим доступа: https://www.debian.org/. Дата обращения: 20.09.2020.

[4] Linux – Getting Started [Электронный ресурс]. Режим доступа: https://linux.org. Дата обращения: 20.09.2020.

[5] Открытый урок. Первое сентября. [Электронный ресурс]. Режим доступа: https://urok.1sept.ru/articles/637896. Дата обращения: 03.12.2021.

[6] Algolist – Сортировка выбором [Электронный ресурс]. Режим доступа: http://algolist.ru/sort/select_sort.php. Дата обращения: 03.12.2021.

[7] Algolist – Сортировка простыми вставками [Электронный ресурс]. Режим доступа: http://algolist.ru/sort/insert_sort.php. Дата обращения: 03.12.2021.

[8] Algolist – Сортировка пузырьком [Электронный ресурс]. Режим доступа: http://algolist.ru/sort/bubble_sort.php. Дата обращения: 03.12.2021.