Отчет по лабораторной работе №3
по курсу "Анализ алгоритмов"
по теме "Изучение алгоритмов сортировки"

Студент: Барсуков Н.М. ИУ7-56
Преподаватель: Волкова Л.Л., Строганов Ю.В.

Содержание

Введение

Алгоритмы сортировки - класс невычислительных алгоритмов. Благодаря значительной экономии времени при двоичном поиске по сравнению с последовательным поиском разработчики программного обеспечения нередко предпочитают хранить информацию в отсортированном виде, чтобы поиск нужных данных можно было вести посредством двоичного или других методов, отличных от полного перебора [**McConnel**].

1 Аналитический раздел

В данном разделе описана поставленная цель и задачи необходимые для ее решения. Приведено описание трех алгоритмов сортировки.

1.1 Постановка задачи

Цель: Изучить и реализовать 3 версии алгоритма сортировки по выбору. Для одной из них требуется расписать нахождение зависимости количества операций от размера массива и его изначального наполнения. Для двух других требуется лишь написать зависимость времени их выполнения от количества сортируемых элементов. Всего рассматривается 3 случая

Для решения поставленной цели, необходимо выполнить следующие поставленные задачи:

1) изучить:
   (a) алгоритм сортировки выбором;
   (b) алгоритм сортировки пузырьком (с флагом);
   (c) алгоритм сортировки вставками.
2) реализовать выше перечисленные алгоритмы;
3) ввести модель оценки сложности алгоритма;
4) провести анализ сложности;
5) произвести замеры времени работы алгоритмов;
6) провести анализ;
7) сделать выводы

Сортировка вставками [**McConnel**] - сортировка, основная идея, который, заключается в том, что при добавлении нового элемента в уже отсортированный список, его стоит сразу вставлять в нужное место вместо того, чтобы вставлять его в произвольное место, а затем снова сортировать весь список. Сортировка вставками считает первый элемент любого списка отсортированным списком длины 1. Двухэлементный отсортированный список создается добавлением второго элемента исходного списка в нужное место одно элементного списка содержащего первый элемент. Теперь можно вставить третий элемент исходного списка в отсортированный двухэлементный список. Этот процесс повторяется до тех пор, пока все элементы исходного списка не окажутся в расширяющейся отсортированной части списка.

Сортировка пузырьком [**McConnel**] - сортировка основной идеей, которой является выталкивание маленьких значений на вершину списка в то время, как большие значения опускаются вниз. У пузырьковой сортировки есть много различных вариантов.

Алгоритм пузырьковой сортировки совершает несколько проходов по списку. При каждом проходе происходит сравнение соседних элементов. Если порядок соседних элементов неправильный, то они меняются местами. Каждый проход начинается с начала списка. Сперва сравниваются первый и второй элементы, затем второй и третий, потом третий и четвертый и так далее; элементы с неправильным порядком в паре переставляются. При обнаружении на первом проходе наибольшего элемента списка он будет переставляться со всеми последующими элементами пока не дойдет до конца списка. Поэтому при втором проходе нет необходимости производить сравнение с последним элементом. При втором проходе второй по величине элемент списка опустится во вторую позицию с конца. При продолжении процесса на каждом проходе по крайней мере одно из следующих по величине значений встает на свое место. При этом меньшие значения тоже собираются наверху. Если при каком-то проходе не произошло ни одной перестановки элементов, то все они стоят в нужном порядке, и исполнение алгоритма можно прекратить. Стоит заметить, что при каждом проходе ближе к своему месту продвигается сразу несколько элементов, хотя гарантированно занимает окончательное положение лишь один.

Сортировка выбором - алгоритм суть которого заключается в том что:

1) разделяем массив на 2 части: отсортированную и неотсортированную. На начальный момент времени отсортированная часть пуста;

2) производим поиск минимального элемента в неотсортированной части массива и вставляем его в конец отсортированной части;

3) повторяем второй шаг, пока размер не отсортированнойчасти массива не станет равным 1.

1.2 Вывод

В данном разделе была поставлена цель и описаны задачи необходимые для ее решения. Приведено описание алгоритмов

2 Конструкторский раздел

В данном разделе представлены схемы трех алгоритмов сортировки: сортировка пузырьком с флагом, сортировка вставками, сортировка выбором. Располагается описание сложности алгоритмов (взята из литературы). Сложность для сортировки вставками описана по выбранной модели в данном разделе.

2.1 Алгоритмы

В данном подразделе представлены схемы трех алгоритмов сортировки:

1) сортировка пузырьком
2) сортировка вставками
3) сортировка выборкой

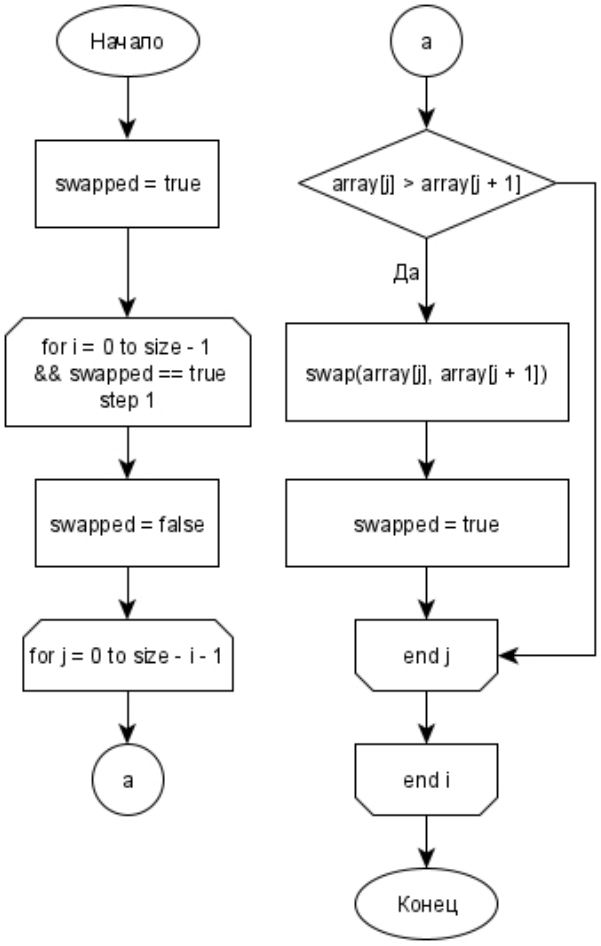

Рис. 1: Схема сортировки пузырьком

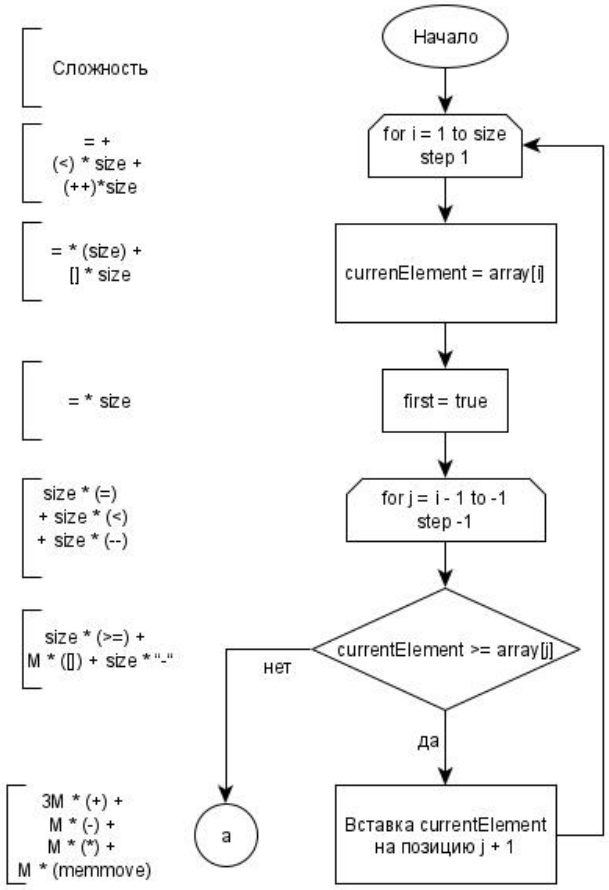

Рис. 2: Сортировка вставками схема 1

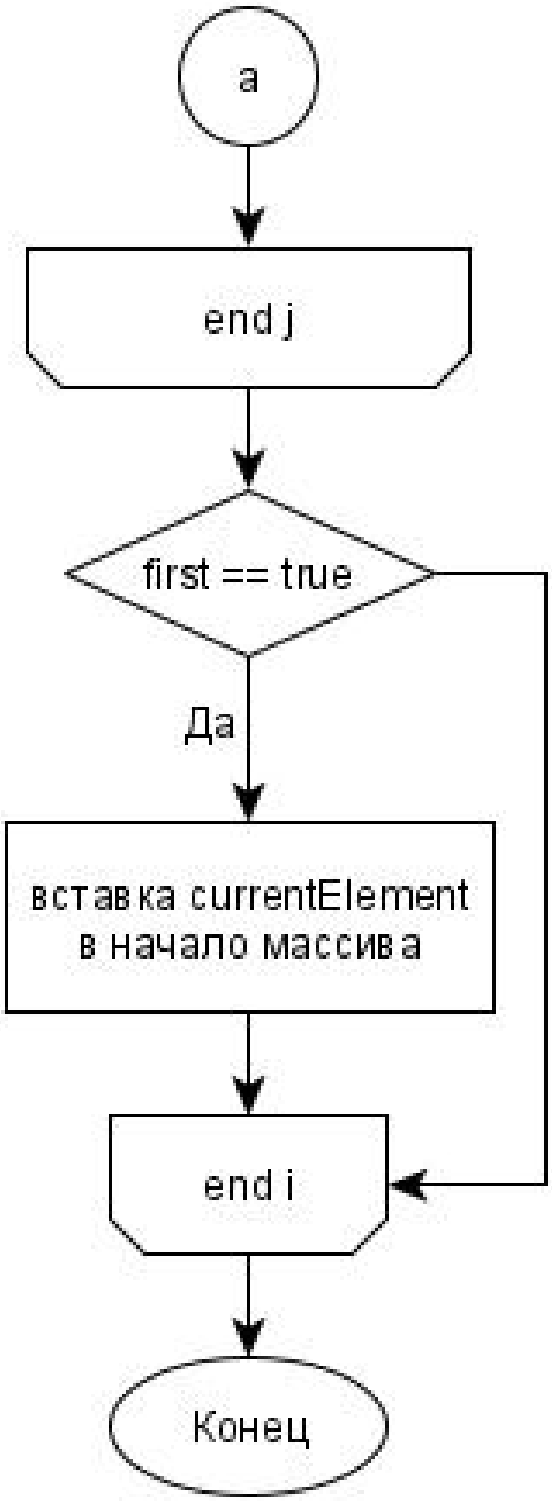

Рис. 3: Сортировка вставками схема 2

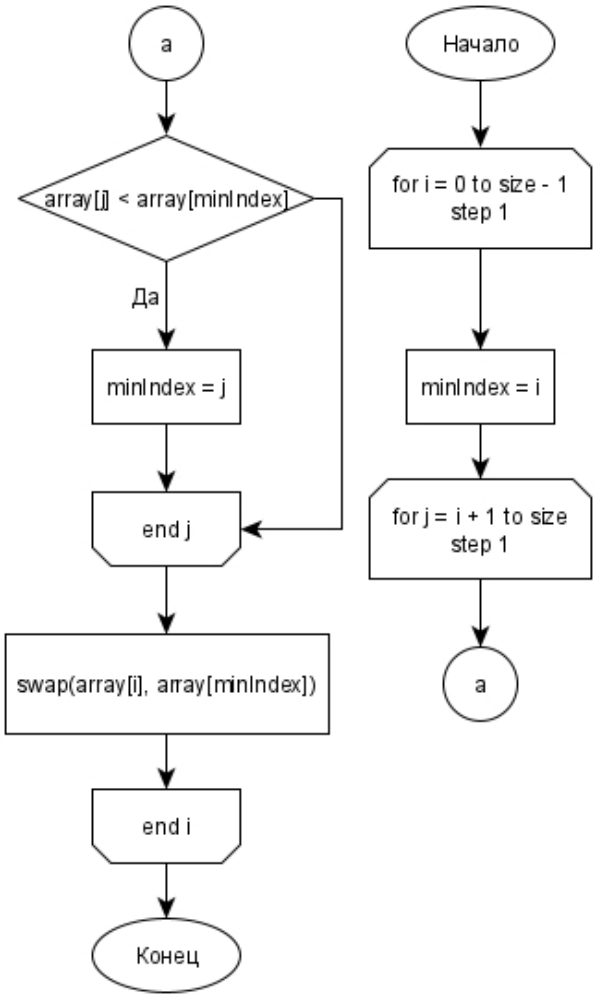

Рис. 4: Схема сортировки выбором

2.2 Модель трудоемкости

Основные правила оценки трудоемкости:

1. Операции единичной трудоемкости: +, -, *, /,
2. C - подобная модель оценки трудоемкости циклов:
   (a) **for** (**int** i = 0; i < N; i++){}
   $A = 2 + N * (2 + A())$;
   (b) **for** (**int** i = 0; i < N + 1; i++){}
   $A = 3 + N * (3 + A())$.

2.3 Оценка сложности алгоритмов

1) пузырек:
   (a) лучший случай: $O(M)$;
   (b) худший случай: $O(M^2)$.
2) сортировка выбором:
   (a) лучший случай: $O(M)$;
   (b) худший случай: $O(M^2)$.

3 Технологический раздел

В данном разделе указаны минимальные системные требования. Описан используемый язык и среда разработки. Представлено описание интерфейса и его скриншоты. Приведен листинг 3 алгоритмов умножения матрицы.

3.1 Минимальные требования

Минимальные системные требования: PC с операционной системой Windows XP/Vista/7/8/10. Требуются устройства ввода: клавиатура, мышь. Устройство вывода: монитор.

3.2 Выбор языка и среды разработки

Для решения данной поставленной задачи, мной был выбран язык c++ стандарта c11 по причине использования ООП. Так же использовалась среда QT

3.3 Интерфейс

Интерфейс представляет из себя простую консольную команду в которой пользователь видит временные результаты сортировки результаты сравнения в процентном соотношении. Ниже представлен скриншот выполнения. Данный интерфейс выбран из за простоты и удобства.

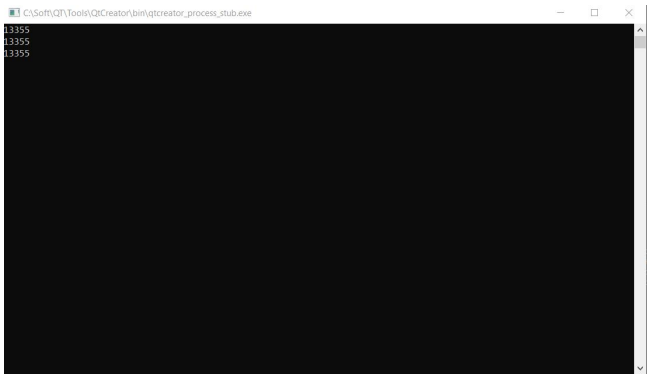

Рис. 5: Интерфейс программы

3.4 Листинг

В данном подразделе приведен листинг 3 сортировок:

1) сортировка пузырьком 1;

2) сортировка вставками 2;

3) сортировка выборочная. 3;

Листинг 1: Сортировка пузырьком

```
#include "sorts.h"


void bubbleSort(int *& array, unsigned size, clock_t&
    time) {

    clock_t tickStart = clock();
    bool swapped = true;
    unsigned iMaxPos = size - 1;
    unsigned jMaxPos = size - 1;
    for (unsigned i = 0; i < iMaxPos && swapped == true;
        ++i, --jMaxPos) {
        swapped = false;
        for (unsigned j = 0; j < jMaxPos; ++j) {
            if (array[j] > array[j + 1]) {
                std::swap(array[j], array[j + 1]);
                swapped = true;
            }
        }
    }
    time = clock() - tickStart;
}
```

Листинг 2: Сортировка вставками

```
#include"sorts.h"

void insertionSort(int*& array, unsigned arraySize,
    clock_t& time) {

    clock_t tickStart = clock();

    for (unsigned i = 1; i < arraySize; ++i) {
        int currentElement = array[i];
        bool first = true;
        for (unsigned j = i; j > 0; --j) {
            if (currentElement >= array[j - 1]) {
                memmove(array + j + 1, array + j, (i -
                    j) * sizeof(int));
                array[j] = currentElement;
                first = false;
                break;
            }
        }
        if (first) {
            memmove(array + 1, array, i * sizeof(int));
            array[0] = currentElement;
        }
    }
```

```
23
24      time = clock() - tickStart;
25  }
```

Листинг 3: Сортировка выбором

```
1
2   #include<sorts.h>
3
4   void selectionSort(int*& array, unsigned arraySize,
        clock_t& time) {
5
6       clock_t tickStart = clock();
7
8       unsigned maxI = arraySize - 1;
9       for (unsigned i = 0; i < maxI; ++i) {
10          unsigned minIndex = i;
11          for (unsigned j = i + 1; j < arraySize; ++j) {
12              if (array[j] < array[minIndex]) {
13                  minIndex = j;
14              }
15          }
16          std::swap(array[i], array[minIndex]);
17      }
18
19      time = clock() - tickStart;
20  }
```

3.5 Вывод

В данном разделе были приведены минимальные системные требования к программе. Указан язык и среда разработки. Приведен образец интерфейса. Располагается листинг.

4 Исследовательский раздел

В данном разделе указаны характеристики машины на которой проводилось тестирование. Представлены замеры и результаты сравнения алгоритмов

4.1 Характеристики оборудования

1. Компьютер:
   (a) Тип компьютера Компьютер с ACPI на базе x64;
   (b) Операционная система Microsoft Windows 10 Pro.
2. Системная плата:
   (a) тип ЦП DualCore Intel Core i5-6200U, 2700 MHz (27 x 100);
   (b) системная плата HP 8079;
   (c) чипсет системной платы Intel Sunrise Point-LP, Intel Skylake-U;
   (d) системная память 8072 МБ (DDR4 SDRAM).

4.2 Замеры и сравнение

Параметры замеров:

1) время вычисляется суммарно на 100 тестов;
2) замеры производятся для:
   (a) идеальный случай;
   (b) худший случай;
   (c) произвольный.
3) замеры сделаны для массивом размером от 1000 до 10000.

Ниже представлены замеры трех алгоритмов сортировки: сортировка пузырьком4.2, сортировка вставками4.2, сортировка выбором4.2.

Таблица 3: Таблица результатов сортировки пузырьком

Алгоритм	Пузырьком		
Размер	Лучший	Худший	Произвольный
1000	0	1343	904
2000	0	5544	4837
3000	0	8632	8539
4000	0	11079	8717
5000	0	17398	13867
6000	0	25271	19930
7000	0	34136	27401
8000	0	44612	36021
9000	0	56303	45723
10000	0	73336	60571

Таблица 1: Таблица результатов замеров сортировки выбором

Алгоритм	Выбором		
Размер	Лучший	Худший	Произвольный
1000	437	375	357
2000	1505	1494	1478
3000	3307	3622	3564
4000	4344	4423	4261
5000	3246	3246	3323
6000	4692	4776	4655
7000	6351	6509	6407
8000	8248	8576	8319
9000	10442	10919	10531
10000	12904	13321	12898

Таблица 2: Таблица результатов сортировки вставками

Алгоритм	Вставками		
Размер	Лучший	Худший	Произвольный
1000	0	391	218
2000	0	1201	484
3000	0	2006	1063
4000	0	2369	1221
5000	15	3635	1873
6000	15	5349	2560
7000	15	7087	3639
8000	15	9408	4681
9000	31	11876	5883
10000	31	14557	7440

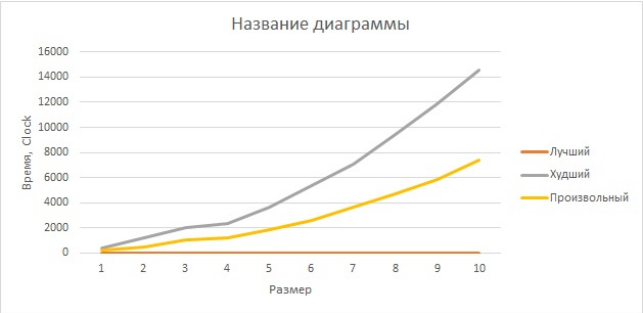

Рис. 6: График результатов сортировки выбором

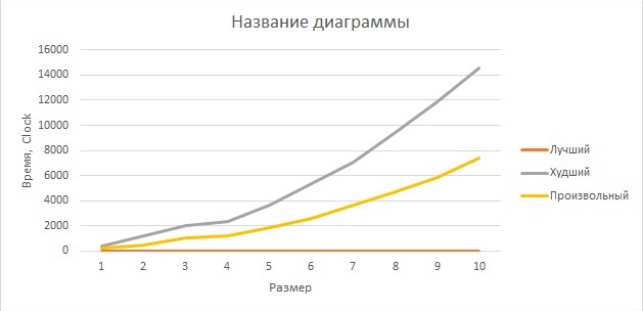

Рис. 7: График результатов сортировки вставками

Рис. 8: График результатов замеров сортировки пузырьком

1. лучший случай: Как мы можем заметить соритировка вставками и пузырьком, почти идеально работают для лучших случаев 0 или 31 один. Это в 416 эффективнее чем сортировка выбором.

2. худший случай: Самый худший результат показала сортировка пузырьком: она в 5 раз хуже сортировки выбором и сортировки вставками

3. произвольный случай: Самым худший результат показал сортировка пузырьком, она в 4.5 хуже сортировки Выбором и хуже в 8.5 раз сортировки вставками.

4.3 Вывод

В данном разделе приведены результаты замеров в таблице и приведены графики. Произведено сравнение алгоритвом для разных вариантов данных.

5 Заключение

В данной лабораторной работе про изучили, реализовали три алгоритма сортировок:

1. сортировка пузырьком
2. сортировка вставками
3. сортировка выбором

Сортировка вставками оказалась в среднем эффективнее сортировки пузырька и выбором. Но отметим тот факт что сортировка вставками и пузырьком отлично показали себя для лучшего случая.

Список литературы

[1] Дж. Макконнелл. Анализ алгоритмов. Активный обучающий подход.-М.:Техносфера, 2009.