Министерство образования и науки Российской Федерации

Севастопольский Государственный Университет

Кафедра ИС

ОТЧЕТ

по лабораторной работе №6

Анализ алгоритмов сортировки

Выполнил: ст. гр. ИТб-11

Куркчи А.Э.

Проверил:

Шишкевич В.Е.

Севастополь

2015

# 1. Цель

Научиться оценивать сложность и количество операций для алгоритмов сортировки.

2. Постановка задач

Вариани №16

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 16 | Сортировка посредством выбора | Модифицированный метод «пузырька» | Сортировка Шелла | Структура Товар, содержит название, стоимость и код, сортировка по убыванию кода. |

3. Текст программы

#include <iostream>

#include <chrono>

#include <stdlib.h>

#include <time.h>

#include <algorithm>

#include <random>

#include <string.h>

#define ITERATIONS\_COUNT 100000

#define ARRAY\_SIZE\_MIN 10

#define ARRAY\_SIZE\_MAX 500

#define ARRAY\_SIZE\_STEP 10

**using** **namespace** std**;**

struct data\_t **{**

int id**;**

char name**[**10**];**

float price**;**

void **operator=(**int v**)** **{**

id **=** v**;**

strcpy**(**name**,**"Item name"**);**

price **=** 1.1f**;**

**}**

bool **operator>(**const data\_t **&**v**)** const **{**

**return** id **>** v**.**id**;**

**}**

**};**

auto seed **=** default\_random\_engine**(**chrono**::**system\_clock**::**now**().**time\_since\_epoch**().**count**());**

void fill\_array**(**data\_t **\***a**,**int n**)** **{**

**for(**int i**=**0**;**i**<**n**;**i**++)** **{**

a**[**i**]** **=** rand**();**

**}**

**}**

void shuffle\_array**(**data\_t **\***a**,**int n**)** **{**

shuffle**(**a**,**a**+**n**,**seed**);**

**}**

void empty**(**data\_t **\***a**,**int n**)** **{}**

void select\_sort**(**data\_t **\***a**,** int n**)** **{**

int i**,**j**,**imax**;**

**for(**i**=**n**;**i**>**1**;**i**--)** **{**

imax **=** 1**;**

**for(**j**=**1**;**j**<=**i**;**j**++)**

**if(**a**[**j**]** **>** a**[**imax**])**

imax **=** j**;**

swap**(**a**[**i**],**a**[**imax**]);**

**}**

**}**

void bubble\_sort**(**data\_t **\***a**,**int n**)** **{**

int i**,**j**;**

**for(**i**=**n**;**i**>**1**;**i**--)**

**for(**j**=**1**;**j**<**i**;**j**++)**

**if(**a**[**j**]** **>** a**[**j**+**1**])**

swap**(**a**[**j**],**a**[**j**+**1**]);**

**}**

void shell\_sort**(**data\_t **\***a**,**int n**)** **{**

int i**,**j**,**temp**,**step **=** n**/**2**;**

**while(**step **>** 0**)** **{**

**for(**i**=**0**;**i**<(**n**-**step**);**i**++)** **{**

j **=** i**;**

**while(**j **>=** 0 **&&** a**[**j**]** **>** a**[**j**+**step**])** **{**

swap**(**a**[**j**],**a**[**j**+**step**]);**

j**--;**

**}**

**}**

step **=** step **/** 2**;**

**}**

**}**

long long pass**(**data\_t **\***a**,**int n**,**void **(\***f**)(**data\_t **\*,**int**))** **{**

auto start\_time **=** chrono**::**high\_resolution\_clock**::**now**();**

**for(**int i**=**0**;**i**<**ITERATIONS\_COUNT**;**i**++)** **{**

shuffle\_array**(**a**,**n**);**

f**(**a**,**n**);**

**}**

auto end\_time **=** chrono**::**high\_resolution\_clock**::**now**();**

**return** chrono**::**duration\_cast**<**chrono**::**milliseconds**>(**end\_time**-**start\_time**).**count**();**

**}**

int main**(**int argc**,**char **\*\***argv**)** **{**

srand**(**time**(**0**));**

data\_t **\***a **=** **new** data\_t**[**ARRAY\_SIZE\_MAX**];**

**for(**int n**=**ARRAY\_SIZE\_MIN**;**n**<=**ARRAY\_SIZE\_MAX**;**n**+=**ARRAY\_SIZE\_STEP**)** **{**

fill\_array**(**a**,**n**);**

cout **<<** n **<<** " "**;**

cout **<<** pass**(**a**,**n**,**empty**)** **<<** " "**;**

cout **<<** pass**(**a**,**n**,**select\_sort**)** **<<** " "**;**

cout **<<** pass**(**a**,**n**,**bubble\_sort**)** **<<** " "**;**

cout **<<** pass**(**a**,**n**,**shell\_sort**)** **<<** " "**;**

cout **<<** endl**;**

**}**

**delete** a**;**

**}**

4. Результат

Результаты, полученные в ходе выполнения программы, отображены на рисунке 1.

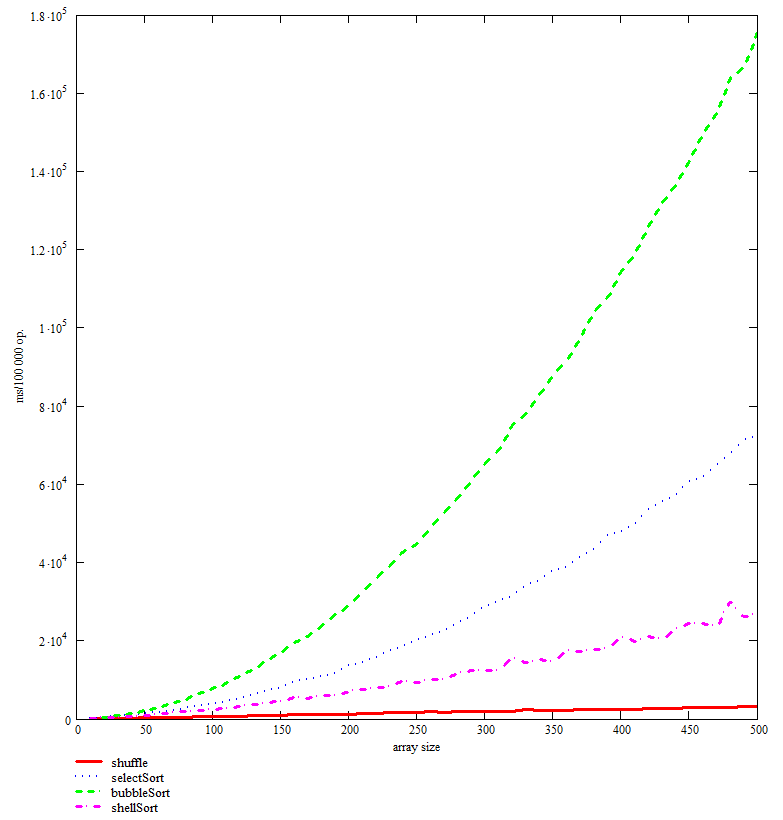


Рисунок 1 – График времени выполнения сортировок

На графике по шкале Ox отложены длины сортируемого массива, а по шкале Oy время 100 000 выполнений функции в миллисекундах.

Вывод

В ходе лабораторной работы были изучены алгоритмы сортировки, оценка их сложности и количества операций. Самой быстрой из предложенных алгоритмов сортировок оказалась сортировка Шела, после неё сортировка прямым выбором, и самой медленной оказалась пузырьковая сортировка, отработавшая при 100 000 раз на массиве из 500 элементов за 176 115 мс.