Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра безопасности информационных систем (БИС)

http://edu.fb.tusur.ru/theme/image.php/fbtheme/core/1487495777/spacer

Сортировки

Отчет о выполненной практической работе

по предмету «Технологии и методы программирования»

Выполнили

Студенты гр. 745

\_\_\_\_\_\_\_Кузнецов Д.С.

\_\_\_\_\_\_\_ЗагребаМ.А.

08.04.2017

Принял

Доцент кафедры БИС

\_\_\_\_\_\_\_ Романов А.С.

08.04.2017

Томск 2017

1 Введение

Цель работы: Научиться применять методы сортировки.

Задание

1. Получить задание на программирование трех методов сортировки.
2. Создать абстрактный класс Sort, в нем определить абстрактный метод сортировки. Сделать три класса-наследника и переопределить абстрактный метод сортировки на соответствующий варианту задания.
3. Сгенерировать массивы размерностью в: 1, 2, 3,4,5, 10, 15, 20, 25, 30, 50, 75, 100, 250, 500 элементов.
4. Для каждой из размерностей высчитать среднее значение количества сравнений и перестановок, время сортировки, проведя по 100 экспериментов.
5. Обязательно сгенерировать лучший и худший варианты для каждой из сортировок и получить значения для них (например, полностью упорядоченный массив, полностью неупорядоченный массив – у каждого из сортировок они свои)
6. Построить зависимость сравнений и перестановок от n (n - количество элементов в массиве ) отобразить графически. Можно использовать Mathcad,Excel или любую другую программу.
7. На той же плоскости отобразить зависимости: y= an; y=nlog2n , y = n2; y = n3.
8. Сделать выводы о трудоемкости методов сортировки.
9. Написать отчет и защитить у преподавателя.

2 Ход работы



Дан неотсортированный массив

Исходный код для сортировки расческой

virtualvoidsort(intarr[], intsize)

{

int k;

int step = size;

intsiz = size;

while (step > 1)

{

step /= 1.247f;

if (step < 1)

step = 1;

k = 0;

for (inti = 0; i + step <siz; ++i)

{

if (arr[i]>arr[i + step])

{

swap(arr[i], arr[i + step]);

k = i;

N++;

}

K++;

}

if (step == 1)

siz = k + 1;

}

}

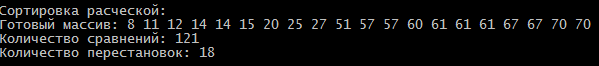


Рисунок 1 – Результат работы сортировки расческой

Исходный код для сортировки выбором

virtualvoid sort(intarr[], intsize)

{

for (inti = 0; i<size - 1; i++)

{

int min = i;

for (int j = i + 1; j <size; j++)

{

K++;

if (arr[j] <arr[min])

{

min = j;

}

}

if (arr[i] != arr[min])

{

swap(arr[i], arr[min]);

N++;

}

};

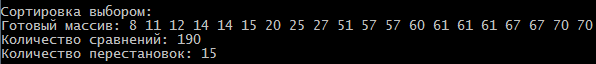


Рисунок 2 – Результат работы сортировки выбором

virtualvoid sort(intarr[], intsize)

{

size\_tBlockSizeIterator;

size\_tBlockIterator;

size\_tLeftBlockIterator;

size\_tRightBlockIterator;

size\_tMergeIterator;

size\_tLeftBorder;

size\_tMidBorder;

size\_tRightBorder;

for (BlockSizeIterator = 1; BlockSizeIterator<size; BlockSizeIterator \*= 2)

{

K++;

for (BlockIterator = 0; BlockIterator<size - BlockSizeIterator; BlockIterator += 2 \* BlockSizeIterator)

{

K++;

LeftBlockIterator = 0;

RightBlockIterator = 0;

LeftBorder = BlockIterator;

MidBorder = BlockIterator + BlockSizeIterator;

RightBorder = BlockIterator + 2 \* BlockSizeIterator;

RightBorder = (RightBorder<size) ?RightBorder :size;

int\* SortedBlock = newint[RightBorder - LeftBorder];

while (LeftBorder + LeftBlockIterator<MidBorder&&MidBorder + RightBlockIterator<RightBorder)

{

K++;

if (arr[LeftBorder + LeftBlockIterator] <arr[MidBorder + RightBlockIterator])

{

K++;

N++;

SortedBlock[LeftBlockIterator + RightBlockIterator] = arr[LeftBorder + LeftBlockIterator];

LeftBlockIterator++;

}

else

{

K++;

N++;

SortedBlock[LeftBlockIterator + RightBlockIterator] = arr[MidBorder + RightBlockIterator];

RightBlockIterator++;

}

}

while (LeftBorder + LeftBlockIterator<MidBorder)

{

N++;

K++;

SortedBlock[LeftBlockIterator + RightBlockIterator] = arr[LeftBorder + LeftBlockIterator];

LeftBlockIterator++;

}

while (MidBorder + RightBlockIterator<RightBorder)

{

N++;

K++;

SortedBlock[LeftBlockIterator + RightBlockIterator] = arr[MidBorder + RightBlockIterator];

RightBlockIterator++;

}

for (MergeIterator = 0; MergeIterator<LeftBlockIterator + RightBlockIterator; MergeIterator++)

{

K++;

N++;

arr[LeftBorder + MergeIterator] = SortedBlock[MergeIterator];

}

deleteSortedBlock;

}

}



Рисунок 3 – Результат работы сортировки слиянием

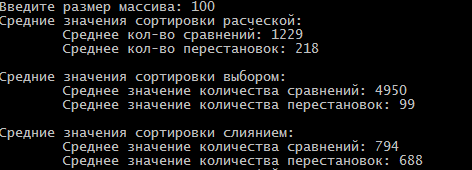


Рисунок 4 – Средние значения сортировок

Таблица 1 – Таблица средних значений для трех сортировок с разным кол-вом элементов в массиве



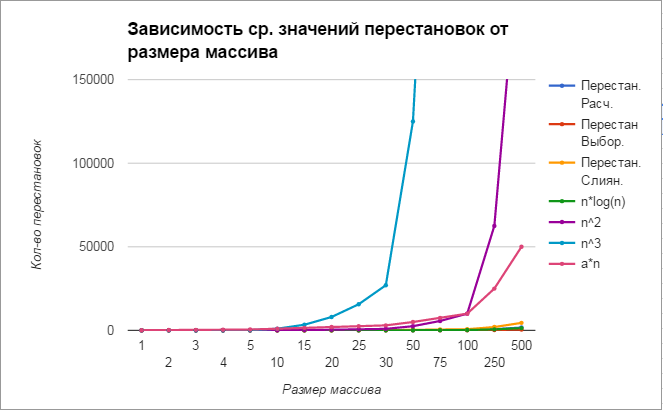


Рисунок 5 – График зависимости средних значений перестановок от размера массива

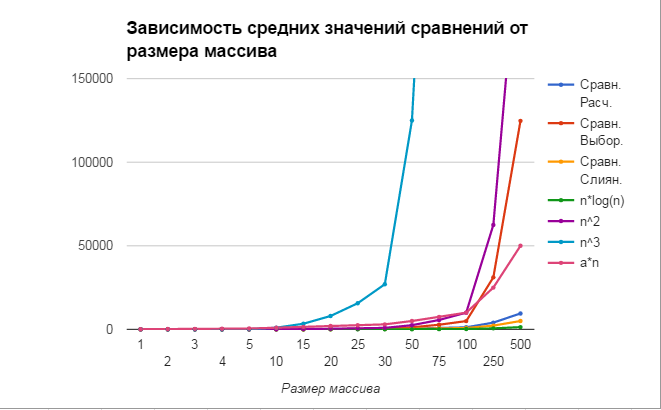


Рисунок 6 – График зависимости средних значений сравнений от размера массива

3 Итог

В ходе выполнения лабораторной работы были реализованы различные способы сортировки данных.