ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГОПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«КАЗАНСКИЙ ПРИВОЛЖСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт вычислительной математики и информационных технологий

Кафедра системного анализа и информационных технологий

**КУРСОВОЙ ПРОЕКТ**

по учебной дисциплине

«Теория и методы программирования»

НА ТЕМУ:

**ЭКСПЕРИМЕНТАЛЬНЫЙ АНАЛИЗ РАЗЛИЧНЫХ МЕТОДОВ СОРТИРОВКИ**

**Выполнил:**

студент 2 курса

дневного отделения

Рахимулин Ринат Рустамович,

обучающийся по профессиональной образовательной программе подготовки бакалавров

«09-208 – Информационная безопасность»

**Научный руководитель:**

Преподаватель, доцент

Васильев Александр Валерьевич

Казань

2014

**СОДЕРЖАНИЕ:**

1. Введение…………………………………………………………………………………………………………………………………3
2. Практическая реализация………………………………………………………………………………………………………4
3. Таблицы расчетов времени работы методов сортировок на различных объемах и типах данных……………………………………………………………………………………………………………………………….…….6
   1. Таблица№1. Для массива, заполненного случайными элементами………………………6
   2. Таблица№2. Для массива, исходно упорядоченного в обратном порядке……………7
   3. Таблица№3. Для частично упорядоченного массива………………………………………………8
   4. Таблица №4. Сравнение простых методов……………………………………………………………...9
   5. Таблица №5. Сравнение простых методов……………………………………………………………..10
   6. Таблица №1 в виде горизонтальных гистограмм\* (сортировка массива СТРОК)..11
   7. Таблица №2 в виде горизонтальных гистограмм (сортировка массива СТРОК)……12
   8. Таблица №1 в виде вертикальных гистограмм(сортировки массива ЧИСЕЛ ОТ 1 до 1млн)…………………………………………………………………………………………………………………………..13
   9. Таблица №2 в виде вертикальных гистограмм(сортировки массива ЧИСЕЛ ОТ 1 до 1млн…………………………………………………………………………………………………………………………….14
   10. Таблица №4 в виде вертикальных гистограмм……………………………………………………….15
   11. Таблица №4 в виде вертикальных гистограмм……………………………………………………….15
4. Диаграммы(функции), визуально показывающие соотношение практических реализаций с теоретическими предположениями……………………………………………………………………………………..16
5. Гибридный метод сортировки……………………………………………………………………………………………….17
   1. Сравнение времени сортировки цифр от 0 до 9……………………………………………………….18
   2. Сравнение времени сортировки целых чисел от 1 до 1млн…………………………………….19
   3. Код гибридной сортировки………………………………………………………………………………………..20
6. Заключение……………………………………………………………………………………………………………………………..21
7. Использованные ресурсы……………………………………………………………………………………………………….22

**Введение.**

Целью этой работы было рассмотреть различные методы сортировок на основе экспериментальных данных. Главная задача – проверить, как же согласуется теоретическая модель сложности с практической; определить аспекты, которые на практике могут быть решающими при выборе сортировок; оценить нюансы поведения сортировок на различных типах входных данных; выявить, если это возможно, какие сортировки с худшим временем работы в теории, будут не такими плохими в применении на практике.

Исходя из смысла задания и рекомендаций по его выполнению, становится понятно, что в главную часть этой работы не вошло описание сортировок и принципов их работы – с этим мы познакомились на лекциях. Гораздо важнее иллюстративность представленного материала и выводы, напрямую не касающиеся теоретических изложений, а лишь подтверждающие или опровергающие их.

В данной работе реализованы различные методы сортировки:

* сортировка «пузырьком»(BubleSort), вставками(InsertSort) и выбором(SelectionSort);
* сортировка Шелла (с различными длинами промежутков)(ShellSort);
* быстрая сортировка(QuickSort);
* сортировка слиянием(MergeSort);
* сортировка деревом(TreeSort, или иначе – пирамидальная сортировка);
* поразрядная сортировка(RadixSort);

В каждой программе происходит сравнение их работы на различных входных данных:

* массивах различных типов:
  + массивах цифр (от 0 до 9)
  + массивах целых чисел
  + массивах строк
  + массивах дат
* массивах различной длины:
  + порядка 50 элементов
  + порядка 500 элементов
  + порядка 5000 элементов
  + порядка 50000 элементов
* по-разному порожденных массивах:
  + случайно сгенерированных
  + частично упорядоченных (не для всех случаев, в некоторых случаях – невозможно провести метод проверки на предупорядоченность)
  + упорядоченных в обратном порядке

Для наглядного представления теоретических выкладок и их соответствия практическим реализациям использовались следующие методы (соответствие теоретическим оценкам, в каких ситуациях какую сортировку использовать и т.д.):

* описание методик сравнения.
* для достоверности результатов проводились серии испытаний, а не единичные сравнения(100 испытаний);
* указывалась результативность методов в среднем;
* для наглядности используются диаграммы, гистограммы и прочая визуализация, а программный код убран в приложения.

На основе полученных результатов сформулирована и реализована собственная гибридная сортировку, использующая преимущества разных методов в зависимости от входных данных.

**Практическая реализация.**

Была написана программа на языке С# для каждого вида(метода) сортировки. Она автоматически подсчитывает время работы для каждого метода сортировки. Программа фиксирует текущее время непосредственно до и после сортировки массива. Программа также находит разность между первым и вторым текущим временем. После программа формирует новый массив и вновь посчитывает время работы, находит среднее время после нескольких циклов.

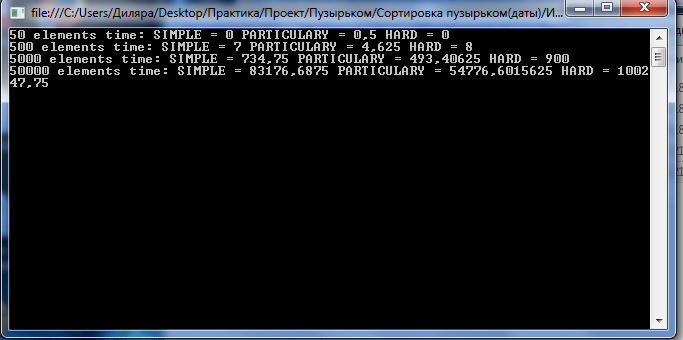
Конечно, время её работы зависит от различных входных данных(даты, строки, числа, цифры) и их объема.

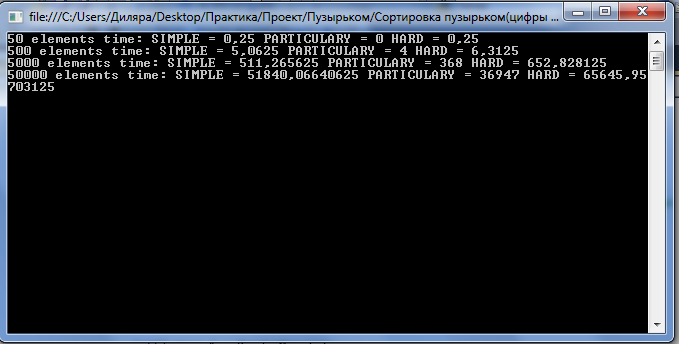
Примеры выводных данных:

Сортировка пузырьком чисел от 0 до 100 000(1 картинка) и Сортировка пузырьком массива дат(от 00.00.0000 до 23.02.2014)(2 картинка).

Где: в строке сначала указывается количество элементов, а после слов

* SIMPLE - время сортировки обычного рандомно заданного массива;
* PARTICULARY – время сортировки частично отсортированного массива;
* HARD – время сортировки обратно упорядоченного рандомного массива





**Таблицы расчетов времени работы методов сортировок на различных объемах и типах данных:** (в качестве простой сортировки – BubleSort, экономия на InsertSort и SelectionSort);

**Таблица №1.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Для массива, заполненного случайными элементами: | | | | | | |
| Объемы и типы данных | Bubble sort | Merge sort | Tree sort | Shell sort | Quick sort | Radix sort |
| 50 | 0,13 | 10,04 | 0,05 | 0,06 | 1,03 |  |
| 500 | 7,1213 | 2,2816 | 0,39 | 4,3306 | 2,22 |  |
| 5000 | 759,46 | 15,265 | 4 | 697,578 | 6,003 |  |
| 50000 | 83176,69 | 174,07 | 53 | 45214,39 | 62,031 |  |
| Даты |  |  |  |  |  |  |
| 50 | 1,28 | 0,36 | 0,12 | 0,86 | 0,11 |  |
| 500 | 150,5548 | 3,8625 | 2,22 | 117,2836 | 5,225 |  |
| 5000 | 16283,375 | 52,015 | 49,03 | 16853,906 | 63,195 |  |
| 50000 | 1689959,7 | 919,6 | 634,63 | 1662217 | 971,297 |  |
| Слова |  |  |  |  |  |  |
| 50 | 0,19 | 3,53 | 0,025 | 0,05 | 0,32 |  |
| 500 | 8,1609 | 1,3903 | 0,3125 | 5,5 | 0,44 |  |
| 5000 | 950,5625 | 15,75 | 2,5078 | 639,8125 | 3,7 |  |
| 50000 | 78724,64 | 226,69 | 38 | 45499,95 | 60 |  |
| от0до9 |  |  |  |  |  |  |
| 50 | 0,08 | 0,47 | 0,025 | 0,09 | 0,02 | 0,075 |
| 500 | 8,6532 | 1,3706 | 0,3125 | 4,7 | 5,21 | 0,4375 |
| 5000 | 763,2656 | 16,86 | 4,578 | 515,937 | 63,24 | 5,859 |
| 50000 | 94840,07 | 200,66 | 48 | 48650,98 | 590,245 | 111,214 |
| от0до1млн |  |  |  |  |  |  |

**Таблица №2.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Для массива, исходно упорядоченного в обратном порядке: | | | | | | |
| Объемы и типы данных | Bubble sort | Merge sort | Tree sort | Shell sort | Quick sort Radix sort | |
| 50 | 0,1 | 0,12 | 0,01 | 0,06 | 0,01 |  |
| 500 | 9,8014 | 1,232 | 0,19 | 5,3206 | 0,7 |  |
| 5000 | 985,138 | 16,5625 | 1,3 | 698,75 | 1,25 |  |
| 50000 | 98247,8 | 284,39 | 30 | 67245,94 | 21,15 |  |
| Даты |  |  |  |  |  |  |
| 50 | 1,17 | 0,5 | 0,12 | 0,83 | 1,22 |  |
| 500 | 138,1547 | 2,375 | 2,48 | 127,7043 | 4,18 |  |
| 5000 | 15498 | 37,594 | 32,03 | 16625,265 | 49,2 |  |
| 50000 | 1721074 | 549 | 448,538 | 1707722 | 653 |  |
| слова |  |  |  |  |  |  |
| 50 | 0,13 | 0,06 | 0,25 | 0,12 | 0,02 |  |
| 500 | 8,2308 | 2,7706 | 0,4625 | 5,16 | 0,35 |  |
| 5000 | 928,453 | 297,5 | 1,9906 | 781,125 | 2,5 |  |
| 50000 | 83880,61 | 25791,13 | 32 | 45798,53 | 32,55 |  |
| от0до9 |  |  |  |  |  |  |
| 50 | 0,06 | 0,09 | 0,02 | 0,02 | 1,01 | 0,04 |
| 500 | 8,8016 | 6,5709 | 0,045 | 4,36 | 4,1 | 0,34 |
| 5000 | 752,82 | 658,3757 | 1,3125 | 538,1875 | 45,501 | 4,5 |
| 50000 | 89645,96 | 71421,02 | 17 | 555753,8 | 501,011 | 72,125 |
| от0до1млн |  |  |  |  |  |  |

**Таблица №3.**

|  |  |
| --- | --- |
| Для частично упорядоченного массива: | |
|  | Bubble sort |
| 50 | 0,5 |
| 500 | 4,625 |
| 5000 | 493,406 |
| 50000 | 54776,6 |
| Даты |  |
| 50 | 1,05 |
| 500 | 59,5 |
| 5000 | 6781,8 |
| 50000 | ~700000 |
| Слова |  |
| 50 | 0,06 |
| 500 | 4,04 |
| 5000 | 348,937 |
| 50000 | 34870,23 |
| от0до9 |  |
| 50 | 0,12 |
| 500 | 3,924 |
| 5000 | 368 |
| 50000 | 36947 |
| от0до1млн |  |

Как видно из таблицы – в случае сортировки «пузырьком» при проверке на предупорядоченность экономия времени доходит до 20%.

**Таблица №4.** Сравнение времени работы простых методов сортировок на основе обычного рандомного массива.

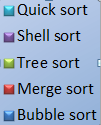
|  |  |  |  |
| --- | --- | --- | --- |
|  | Вставками | Пузырьком | Выбором |
| 50 | 0,1 | 0,13 | 0,5 |
| 500 | 2 | 7,1213 | 4,75 |
| 5000 | 213 | 759,46 | 409 |
| 50000 | 23396 | 83176,69 | 43194,5 |
| Даты |  |  |  |
| 50 | 0,5 | 0,48 | 1 |
| 500 | 41,7 | 51,5548 | 89 |
| 5000 | 4007 | 6283,375 | 11909,4 |
| 50000 | 434112,88 | 689959,7 | 794089,1 |
| Слова |  |  |  |
| 50 | 0,5 | 0,08 | 2 |
| 500 | 2,25 | 6,6532 | 2,5 |
| 5000 | 162 | 511,2656 | 205,3 |
| 50000 | 20466 | 51840,07 | 30372,8 |
| числа от 0до1млн |  |  |  |
| 50 | 0,5 | 0,09 | 0,25 |
| 500 | 3,75 | 5,1609 | 3,06 |
| 5000 | 159,98 | 487,5625 | 259,9 |
| 50000 | 18068,98 | 48724,64 | 38426,6 |
| цифры от0до10 |  |  |  |

**Таблица №5.** Сравнение времени работы простых методов сортировок на основе обычного рандомного массива, отсортированного в обратном порядке(«тяжелый» случай).

|  |  |  |  |
| --- | --- | --- | --- |
| Простые методы(обратно отсортированный массив | Вставками | Пузырьком | Выбором |
| 50 | 0,01 | 0,14 | 0,1 |
| 500 | 0,1 | 9,8014 | 5 |
| 5000 | 1 | 985,138 | 437,8 |
| 50000 | 3 | 100247,8 | 44675,03 |
| Даты |  |  |  |
| 50 | 0,99 | 0,47 | 0,5 |
| 500 | 93 | 53,1547 | 103,8 |
| 5000 | 7393,9 | 7498 | 17215,35 |
| 50000 | 953265,25 | 800088,5 | 911765,8 |
| Слова |  |  |  |
| 50 | 0,01 | 0,04 | 0,1 |
| 500 | 4 | 6,8016 | 2,5 |
| 5000 | 345 | 652,82 | 217 |
| 50000 | 57576 | 65645,96 | 32847,4 |
| числа от 0до1млн |  |  |  |
| 50 | 0,1 | 0,14 | 0,1 |
| 500 | 5,5 | 9,8014 | 2 |
| 5000 | 291,44 | 985,138 | 205 |
| 50000 | 40212,93 | 100247,8 | 29097 |
| цифры от0до10 |  |  |  |

**Таблица №1 в виде горизонтальных гистограмм\* (сортировка массива СТРОК):**

\*по горизонтали – время выполнения/работы метода сортировки



**50 элементов**

**500 элементов**

**5000**

**элементов**

**50 000**

**элементов**

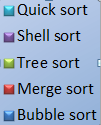
**50 elements**

**500 elements**

**50000 elements**

**5000 elements**

**Таблица №2 в виде горизонтальных гистограмм (сортировка массива СТРОК):**



**50 элементов**

**500 элементов**

**5000**

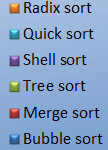
**элементов**

**50 000**

**элементов**

**Таблица №1 в виде вертикальных гистограмм(сортировки массива ЧИСЕЛ ОТ 1 до 1млн):**

\*по вертикали – время выполнения/работы метода сортировки



**50 элементов**

**500 элементов**

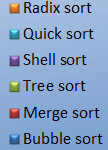
**5000**

**элементов**

**50 000**

**элементов**

**Таблица №2 в виде вертикальных гистограмм(сортировки массива ЧИСЕЛ ОТ 1 до 1млн):**



**50 элементов**

**500 элементов**

**5000**

**элементов**

**50 000**

**элементов**

**Таблица №5** в виде вертикальных гистограмм (для обычного рандомного массива СТРОК).

**Таблица №5** в виде вертикальных гистограмм (для обратно упорядоченного массива СТРОК).

**Диаграммы(функции)\*, визуально показывающие соотношение практических реализаций с теоретическими предположениями:**

\*по вертикали – время выполнения

\*соотношение практических данных(цветная кривая) и теоретических (черная кривая);

**Гибридный метод сортировки.**

Описание:

В качестве гибридного метода сортировок – были использованы быстрая сортировки, сортировка деревом и сортировка пузырьком, на базе которых(лучшей статистике работы) и был построен исследуемый алгоритм.

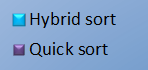
Суть работы – рекурсия. Сначала происходит проверка: если количество оперируемых элементов(сортируемых) меньше 16 –\* начинает работать сортировка пузырьком. В ином случае – пока глубина рекурсии не достигла величины 2\*log(n) происходить сортировка деревом(где n-изначальное количество элементов). Каждый вызов рекурсии производит соответствующую проверку на 16 элементов. Если же глубина рекурсии достигла заданного значения – происходит сортировка деревом оставшейся части массива(опять же до 16 оставшихся элементов, на которых происходит сортировка пузырьком).

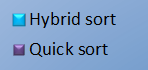
\*16 – было взято из лекций на основе работы стандартной сортировки в библиотеках С++;

Как видно из диаграмм\*\*, гибридный метод работает быстрее, чем рассмотренные стандартные самые теоретически быстрые варианты(Quick sort, Tree Sort and Buble Sort). До 1000 элементов – даже быстрее, чем стандартная сортировка из библиотек С++!

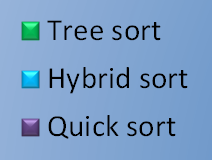
\*\*по горизонтали – время выполнения, по вертикали – длина массивов;

1. Сравнение времени сортировки цифр от 0 до 9:





1. Сравнение времени сортировки целых чисел от 1 до 1млн:



Код гибридной сортировки:  
//Гибридная сортировка

static void GybridSort(ref int[] mass, int start, int end, ref int twoLogn)

{

//Если длина меньше/равно 16 - то сортировка пузырьком.

if (end - start <= 16)

{

for (int i = start; i < end; i++)

{

for (int j = start; j < end-i; j++)

{

if (mass[j].CompareTo(mass[j + 1]) > 0)

{

Swap(mass, j, j + 1);

}

}

}

}

else

{

//Проводим Быструю сортировку, до глубины 2\*log(n)

//А затем - сортировку деревом

if (twoLogn < 2 \* Math.Log(mass.Length, 2))

{

int i = start, j = end;

//Опорный элемент:

int x = mass[(start + end) / 2];

do

{

//Двигаем два курсора

while (mass[i].CompareTo(x) < 0) i++;

while (mass[j].CompareTo(x) > 0) j--;

if (i <= j)

{

if (i < j) Swap(mass, i, j);

i++;

j--;

}

} while (i <= j);

twoLogn++;

if (i < end)

GybridSort(ref mass, i, end, ref twoLogn); ;

if (start < j)

GybridSort(ref mass, start, j, ref twoLogn);

}

else

{

//Дерево

for (int i = end - start / 2; i >= start + 16; i--)

{

Push(mass, i, end - 1);

}

for (int i = end-start - 1; i > start+16; i--)

{

Swap(mass, start, i);

Push(mass, start, i - 1);

}

//"Добиваем" через сортировку пузырьком

for (int i = start; i < start+16; i++)

{

for (int j = start; j < start+16-i; j++)

{

if (mass[j].CompareTo(mass[j + 1]) > 0)

{

Swap(mass, j, j + 1);

}

}

}

}

}

}

//Меняет местами

static void Push(int[] a, int i, int last)

{

int k = i;

if ((2 \* i + 1) <= last)

{

if ((a[2 \* i + 1] > a[i]))

k = 2 \* i + 1;

if ((2 \* i + 2) <= last && a[2 \* i + 2] > a[k])

k = 2 \* i + 2;

if (k > i)

{

Swap(a, i, k);

Push(a, k, last);

}

}

}

**Заключение.**

Алгоритм **пузырьковой сортировки(BubleSort)** имеет плохие показатели даже среди простых методов с квадратичной сложностью. Несколько лучше неё работают **сортировки вставками(InsertSort) и выбором(SelectionSort)**.

**Сортировка Шелла(ShellSort)** отлично справляется только с упорядоченными данными.

Сортировку Шелла нецелесообразно использовать на практике. Она сложна в реализации, но не дает такую ​​скорость, что и методы, которые сравнимы с ней по сложности программной реализации.

**Сортировка слиянием(MergeSort)** отлично работает со сложными, нестандартными данными (например, с датами), но в общем показала лишь средний результат. Следует помнить, что **сортировка слиянием** требует дополнительную память размером порядка N.

Если вам нужно отсортировать массив цифр или же небольших чисел, размерность которого менее 1000, то будет лучше использовать **древесную(пирамидальную)(TreeSort or HeapSort)**или**поразрядную сортировки(RadixSort)**.

При сортировке больших массивов исходных данных лучше использовать **быструю сортировку(QuickSort)**.

**Быстрая сортировка**, действительно, оказалось самой быстрой. Но если нужно гарантировать приемлемое время работы метода, то надо применять либо древесную сортировку, либо сортировку слиянием.

**Использованные ресурсы.**

1. Сайт поддержки программистов cyberforum.com
2. Сайт компании MicroSoft по реализациям функций msdn.com
3. Сайт электронной энциклопедии Wikipedia.com