**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕНЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.Шухова»**

**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 3

По дисциплине: Алгоритмы и СД

Тема: «Сравнительный анализ методов сортировки (Pascal/C)»

Выполнил: ст.группы ПВ - 21

Браткова И.О.

Белгород 2017

**Цель работы:** изучение методов сортировки массивов и приобретение навыков в проведении сравнительного анализа различных методов сортировки.

**З а д а н и е**

1. Изучить временные характеристики алгоритмов.

2. Изучить методы сортировки:

1) включением;

2) выбором;

3) обменом:

3.1) улучшенная обменом 1;

3.2) улучшенная обменом 2;

4) Шелла;

5) Хоара;

6) пирамидальная.

3. Программно реализовать методы сортировки массивов.

4. Разработать и программно реализовать средство для проведения экспериментов по определению временных характеристик алгоритмов сортировки.

5. Провести эксперименты по определению временных характеристик алгоритмов сортировки. Результаты экспериментов представить в виде таблицы 9, клетки которой содержат время выполнения алгоритма сортировки массива с заданным количеством элементов. Провести эксперимент для упорядоченных, неупорядоченных и упорядоченных в обратном порядке массивов (для каждого типа массива заполнить отдельную таблицу).

6. Построить графики функций временной сложности алгоритмов сортировки.

7. Определить аналитическое выражение функции временной сложности алгоритмов сортировки.

8. Определить порядок функций временной сложности алгоритмов сортировки при сортировке упорядоченных, неупорядоченных и упорядоченных в обратном порядке массивов.

***Задание 1***

*Временная сложность (ВС) алгоритма* — это зависимость времени выполнения алгоритма от количества обрабатываемых входных данных. Здесь представляет интерес среднее и худшее время выполнения алгоритма.

Наиболее точной оценкой является аналитическое выражение для функции: *t = t(N)*, где *t* — время, *N* — количество входных данных (размерность). Данная функция называется *функцией временной сложности* (ФВС).

Математическая запись, позволяющая не учитывать эти условия, называется О-нотацией. Она определяется следующим образом. Если *f*(*N*) и *g*(*N*) есть функции, определенные в пространстве положительных целых чисел, тогда запись *f*(*N*) = О(*g*(*N*)) обозначает, что существует такая константа *с*, что для всех остаточно больших положительных целых *N* | *f*(*N*)| ≤ *с*|*g*(*N*)|. При этих условиях можно также сказать, что «*f*(*N*) имеет порядок не больший, чем *g*(*N*)» или «*f*(*N*) растет не быстрее, чем *g*(*N*)».

***Задание 2***

*Сортировка* — это процесс перестановки элементов данного множества в определенном порядке. Сортировка является идеальным примером огромного разнообразия алгоритмов, выполняющих одну и ту же задачу, многие из которых в некотором смысле являются оптимальными, а большинство имеет какие-либо преимущества по сравнению с остальными.

Сортировки бывают:

1. ***Включением:***

Этот метод обычно используют игроки в карты. Элементы (карты) условно разделяются на готовую последовательность а1,…, аi-1 и входную последовательность аi,…, аn. На каждом шаге, начиная с i=2 и увеличивая i на единицу, берут 1-й элемент входной последовательности и передают его в готовую последовательность, вставляя на подходящее место.

*Временная сложность:*

Если первоначальный массив отсортирован, то на каждом просмотре делается только одно сравнение, так что эта сортировка имеет порядок O(N).

Среднее число сравнений в сортировке простыми вставками также составляет O(N2).

1. ***Выбором:***

При сортировке выбором массив разбивается на две части: отсортированная и неотсортированная. На каждом шаге из неотсортированной части выбирают наименьший элемент и присоединяют его к отсортированной части.

*Временная сложность:*

На первом просмотре выполняется N – 1 сравнение, на втором — N – 2 и т.д. Следовательно, общее число сравнений равно (N – 1) + (N – 2) + (N – 3) + ... + 1 = N\*(N – 1) / 2, что составляет O(N2).

1. ***Обменом:***

Идея обменной сортировки заключается в том, что два элемента, нару-шающие требуемый порядок, меняются местами.

*Временная сложность:*

На первом просмотре выполняется N – 1 сравнение, на втором — N – 2 и т.д. Следовательно, общее число сравнений равно (N – 1) + (N – 2) + (N – 3) + ... + 1 = N\*(N – 1) / 2, что составляет O(N2).

* 1. ***улучшенная обменом 1:***

После каждого прохода в сортировке обменом может быть сделана проверка, были ли совершены перестановки в течение данного прохода. Если перестановок не было, то это означает, что массив упорядочен и дальнейших проходов не требуется.

*Временная сложность:*

В худшем случае, когда массив упорядочен в обратном порядке, выполняется N – 1 проходов. На первом проходе выполняется N – 1 сравнение, на втором — N – 2 и т.д. Следовательно, общее число сравнений равно (N – 1) + (N – 2) + (N – 3) + ... + 1 = N·(N – 1) / 2, что составляет O(N2).

В лучшем случае, когда массив уже упорядочен, потребуется всего один проход и N – 1 сравнение, что составляет O(N).

* 1. ***улучшенная обменом 2:***

В отличие от улучшенной сортировки обменом 1 здесь в течение про-хода фиксируется последний элемент, участвующий в обмене. В очередном проходе этот элемент и все предшествующие в сравнении не участвуют, т.к. все элементы до этой позиции уже отсортированы.

*Временная сложность:*

Аналогична улучшенной сортировке обменом 1.

1. ***Шелла:***

Сортировка Шелла — это улучшенный метод сортировки вставками. Был предложен Д.Л. Шеллом в 1959 г. Рассмотрим этот метод на примере (см. ниже). При первом проходе группируются и сортируются элементы, отстоящие друг от друга на 5 позиций: (Х1,Х6), (Х2,Х7), (Х3), (Х4), (Х5), т.е. выполняется сортировка массива с шагом 5; при втором проходе — эле-менты, отстоящие друг от друга на три позиции: (Х1,Х4,Х7), (Х2,Х5), (Х3,Х6); при третьем — на 1 позицию.

Если некоторый массив частично отсортирован с использованием шага h, а затем сортируется с использованием меньшего шага, то массив остает-ся частично отсортированным по шагу h, т.е. последующие частичные сор-тировки не нарушают результата предыдущих сортировок. Следующий шаг сортировки меньше предыдущего, а последний — равен 1.

*Временная сложность*:

Анализ сложен. О(N(*log 2*N)), при правильном выборе шагов: O(N1.2).

1. ***Хоара:***

Сортировка Хоара — лучший из известных до сего времени метод сор-тировки массивов. Он обладает столь блестящими характеристиками, что его автор Ч. Хоар назвал эту сортировку быстрой. Быстрая сортировка основана на том факте, что для достижения наибольшей эффективности желательно разбить массив на подмассивы и сортировать подмассивы мень-шего размера.

*Временная сложность:*

Если каждый раз одна из частей массива содержит не более одного элемента, то порядок будет O(N2).

Если массив разделён на две равные части: O(N·log2N).

1. ***Пирамидальная:***

Пирамидальная сортировка является улучшенной сортировкой выбором. Из массива, состоящего из N элементов, выбирается максимальный и меняется местами с последним. Затем рассматривается массив из N = N – 1 элементов. Процесс повторяется до тех пор, пока количество рассматриваемых элементов больше одного. Пира-мидальная сортировка отличается от сортировки выбором поиском максимального элемента. Чтобы понять пирамидальную сортировку, массив нужно интерпретировать как бинарное дерево, в корне которого находится первый элемент массива, на втором уровне — второй и третий, на третьем — с четвертого по седьмой и т.д. Для i-го элемента массива можно определить номер P элемента, являющегося родителем, как P = i div 2; номер L элемента, являющегося левым сыном, как L = 2·i; номер R элемента, являющегося правым сыном, как R=2·i+1. Если в массиве (в дереве) N элементов, то последний элемент, имеющий хотя бы одного левого сына, имеет номер (N div 2).

***Временная сложность:***

O(N·log2N).

***Задание 3***

|  |  |
| --- | --- |
| *void vkl\_sort(int A[],int nn)*  **//Сортировка включением**  { int i, j, k;  for (j=1; j<nn; j++)  {  k = A[j];  i = j-1;  while ((k<A[i]) && (i>=0))  {  A[i+1] = A[i]; i -= 1;  }  A[i+1] = k;  }  } | *void vibor\_sort(int A[],int nn)*  **//Сортировка выбором**  { int i, j, x, k;  for (i=0; i<nn-1; i++)  {  x = A[i]; k = i;  for (j=i+1; j<nn; j++)  if (A[j] < x)  {  k = j; x = A[k];  }  A[k] = A[i]; A[i] = x;  }  } |
| *void o\_sort(int A[], int nn)*  **//Сортировка обменом**  { int i, j, tmp;  for (i=0; i<nn-1; i++)  {  for (j=(nn-1); j>i; j--)  {  if (A[j-1]>A[j])  {  tmp = A[j-1]; A[j-1] = A[j];  A[j] = tmp;  }  }  }  } | *void shell\_sort(int A[], int nn)*  **//Сортировка Шелла**  { int i, j, step, tmp;  for (step=nn/2; step>0; step/=2)  for (i=step; i<nn; i++)  {  tmp = A[i];  for (j=i; j>=step; j-=step)  {  if (tmp < A[j-step]) A[j] = A[j-step];  else break;  }  A[j] = tmp;  }  } |
| *void q\_sort(int A[], int l, int r)*  **// Сортировка Хоара**  {  int x = A[l], i = l, j = r, t;  while (i<=j)  {  while (A[i]<x) i++;  while (A[j]>x) j--;  if (i<=j)  {  t = A[i];  A[i] = A[j];  A[j] = t;  i++;  j--;  }  }  if (l<j) q\_sort(A,l,j);  if (i<r) q\_sort(A,i,r);  }  void hoar\_sort(int a[], int nn)  {  q\_sort(a, 0, nn-1);  } | *void sift(int A[], int l,int r)*  **//Пирамиальная сортировка**  {  int i=l, j = 2\*r+1, x=A[l], k;  if ((j<r) && (A[j]<A[j+1])) j++;  while ((j<=r) && (x<A[j]))  {  k = A[i]; A[i] = A[j];  A[j]=k; i = j; j = 2\*j+1;  if ((j<r) && (A[j]<A[j+1])) j++;  }  }  void heap\_sort(int A[], int nn)  {  int l = nn/2, x, i, r = nn-1;  while (l>0)  {  l = l-1; sift(A, l, r);  }  while (r>0)  {  x = A[0]; A[0] = A[r]; A[r] = x;  r--; sift(A, l, r);  }  } |

***Задание 4***

#include <time.h> #define N ***<необходимое количество элементов>***

…

int main()

{ setlocale(LC\_ALL, "Rus"); int a[N], b[N], c[N], d[N], e[N], f[N], g[N], k[N], i=0, t;

for(i=0; i<N; i++) a[i] = rand() % 100;

clock\_t time\_start= clock(); vkl\_sort(a, N); clock\_t time\_end = clock() - time\_start;

printf("Включением %f\n", (double)time\_end / CLOCKS\_PER\_SEC);

return 0;

}

***Задание 5***

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сортировка | Количество элементов в массиве **(НЕУПОРЯДОЧЕННЫЕ)** | | | | | | | | |  |
|  | 5000 | 10000 | 15000 | 20000 | 25000 | 30000 | 35000 | 40000 | 45000 |  |
| Включением | 0,018 | 0,07 | 0,162 | 0,284 | 0,453 | 0,622 | 0,92 | 1,172 | 1,48 |  |
| Выбором | 0,034 | 0,143 | 0,31 | 0,59 | 0,897 | 1,26 | 1,686 | 2,205 | 2,768 |  |
| Обменом | 0,058 | 0,27 | 0,705 | 1,172 | 1,919 | 2,77 | 3,81 | 5,058 | 6,439 |  |
| Обменом 1 | 0,59 | 0,278 | 0,653 | 1,028 | 1,403 | 1,778 | 2,153 | 4,931 | 6,34 |  |
| Обменом 2 | 0,52 | 0,229 | 0,599 | 0,969 | 1,339 | 1,709 | 2,079 | 3,394 | 5,24 |  |
| Шелла | 0 | 0,002 | 0,002 | 0,003 | 0,004 | 0,005 | 0,006 | 0,008 | 0,009 |  |
| Хоара | 0 | 0,001 | 0,002 | 0,001 | 0,002 | 0,002 | 0,003 | 0,003 | 0,004 |  |
| Пирамидальная | 0 | 0 | 0 | 0 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |  |
|  |  |  |  |  |  |  |  |  |  |  |
| Сортировка | Количество элементов в массиве **(УПОРЯДОЧЕННЫЕ)** | | | | | | | | |  |
| 5000 | 10000 | 15000 | 20000 | 25000 | 30000 | 35000 | 40000 | 45000 |  |
| Включением | 0 | 0 | 0 | 0 | 0,001 | 3000 | 3500 | 4000 | 4500 |  |
| Выбором | 0,034 | 0,139 | 0,305 | 0,592 | 0,912 | 1,0591 | 1,28 | 1,5009 | 1,7218 |  |
| Обменом | 0,033 | 0,136 | 0,383 | 0,548 | 0,856 | 1,0086 | 1,2144 | 1,4202 | 1,626 |  |
| Обменом 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| Обменом 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| Шелла | 0 | 0 | 0,001 | 0,001 | 0,001 | 0,0015 | 0,0018 | 0,0021 | 0,0024 |  |
| Хоара | 0 | 0,008 | 0,016 | 0,018 | 0,024 | 0,0306 | 0,0364 | 0,0422 | 0,048 |  |
| Пирамидальная | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
|  |  |  |  |  |  |  |  |  |  |  |
| Сортировка | Количество элементов в массиве **(ОБРАТНОУПОРЯДОЧЕННЫЕ)** | | | | | | | | |  |
| 5000 | 10000 | 15000 | 20000 | 25000 | 30000 | 35000 | 40000 | 45000 |  |
| Включением | 0,037 | 0,136 | 0,324 | 0,565 | 0,847 | 1,229 | 1,666 | 2,217 | 2,768 |  |
| Выбором | 0,0035 | 0,141 | 0,315 | 0,56 | 0,863 | 1,221 | 1,676 | 2,162 | 2,796 |  |
| Обменом | 0,048 | 259 | 0,438 | 0,858 | 1,202 | 1,771 | 2,393 | 3,134 | 3,96 |  |
| Обменом 1 | 0,038 | 0,183 | 0,373 | 0,632 | 1,092 | 1,286 | 1,816 | 2,4 | 2,939 |  |
| Обменом 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0,001 | 0,001 |  |
| Шелла | 0,001 | 0,001 | 0,001 | 0,002 | 0,002 | 0,002 | 0,002 | 0,003 | 0,003 |  |
| Хоара | 0,002 | 0,004 | 0,007 | 0,012 | 0,015 | 0,017 | 0,019 | 0,022 | 0,026 |  |
| Пирамидальная | 0 | 0 | 0 | 0 | 0 | 0 | 0,001 | 0,001 | 0,001 |  |

***Задание 6***

Включением: Выбором:

Обменом: Обменом 1:

Обменом 2: Шелла:

Хоара: Пирамидальная:

***Задание 7-8***

1. ***Включением:***

Худшее: O(N2)

Лучшее: O(N)

Среднее: O(N2)

1. ***Выбором:***

Худшее: O(N2)

Лучшее: O(N2)

Среднее: O(N2)

1. ***Обменом:***

Худшее: O(N2)

Лучшее: O(N2)

Среднее: O(N2)

* 1. ***улучшенная обменом 1:***

Худшее: O(N2)

Лучшее: O(N)

Среднее: O(N2)

* 1. ***улучшенная обменом 2:***

Худшее: O(N2)

Лучшее: O(N)

Среднее: O(N2)

1. ***Шелла:***

Худшее: O(N2)

Лучшее: O(N\*log2N), но при правильном выборе шагов: O(N1.2).

Среднее: O(N2), зависит от выбранных шагов.

1. ***Хоара:***

Худшее: O(N2)

Лучшее: O(N\*logN) – при делении на 2 части, O(N) – при делении на 3 части.

Среднее: O(N\*logN).

1. ***Пирамидальная:***

Худшее: O(N·log2N).

Лучшее: O(N·log2N).

Среднее: O(N·log2N).