ФГБОУ ВО “Чувашский государственный университет им. И. Н. Ульянова”
Факультет: ИВТ
Кафедра: Вычислительной техники
Предмет: Структуры и алгоритмы обработки данных

Лабораторная работа № 5.
ИССЛЕДОВАНИЕ МЕТОДОВ СОРТИРОВКИ

Вариант 7

Выполнил: студент группы ИВТ-41-20
Галкин Дмитрий Сергеевич
Проверил:
доцент
Павлов Леонид Александрович

Чебоксары
2022

***Цель работы*:** Ознакомление с методами сортировки, получение практических навыков программирования задач сортировки, получение навыков экспериментальных исследований алгоритмов.

№ варианта	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	+	+			+	+		+			+	+		+		+	
2	+		+		+		+		+	+			+		+	+	
3		+		+		+	+	+		+			+	+			+
4			+	+			+	+	+		+	+			+		+
5	+	+			+	+	+			+		+			+	+	
6	+		+		+			+	+		+		+		+		+
7		+		+	+	+		+		+			+	+			+
8			+	+	+		+		+		+	+		+		+	
9	+	+				+	+	+		+		+		+			+
10	+		+				+	+	+		+		+		+	+	
11		+		+	+	+	+				+		+	+		+	
12			+	+	+			+	+	+		+			+		+

Номера столбцов соответствуют следующим алгоритмам поиска:

1. Простая сортировка вставками.
2. Сортировка бинарными вставками.
3. Сортировка вставками в связанный список.
4. Сортировка Шелла.
5. Пузырьковая сортировка.
6. Шейкер-сортировка.
7. Быстрая сортировка (рекурсивный вариант).
8. Быстрая сортировка (итерационный вариант).
9. Цифровая обменная сортировка.
10. Простая сортировка выбором.
11. Пирамидальная сортировка.
12. Сортировка подсчетом (перечислением).
13. Сортировка распределяющим подсчетом.
14. Сортировка естественным двухпутевым слиянием.
15. Сортировка простым двухпутевым слиянием.
16. Сортировка слиянием списков для естественного двухпутевого слияния.
17. Сортировка слиянием списков для простого двухпутевого слияния.

2. Сортировка бинарными вставками

```
size := arr.length
for i := 1 to i + 1 {
   if arr[i] < arr[i - 1] {
      temp := arr[i];
      j := i - 1;
      // Начало с индекса 0
      low := 0;
      // Конец предыдущего значения
      high := i - 1;
      // Поиск пополам, эффективно сок
      while (low <= high) {
         mid := (low + high) / 2;
         if temp < arr[mid] {
            high := mid - 1;
         } else {
            low := mid + 1;
         }
          countCompare++;
      }
   }

   for to j - 1 {
      arr[j + 1] := arr[j];
      transport++;
   }
   arr[high + 1] :=  temp;
   transport++;
}
```

10. Простая сортировка выбором

```
n = arr.length
for i := 0 to I + 1 {
   minIndex := min(arr, i, n - 1);
   swap(values, i, minIndex);
   transport++;
}

PROCEDURE min(int[] arr, int begin, int end) {
      minVal := arr[begin];
      minIndex := begin;
      for i := begin + 1 to i + 1 {
         countCompare++;
         if arr[i] < minVal {
            minVal := arr[i];
            minIndex := i;
         }
      }
```

```
    return minIndex;
  }
```

13. Сортировка распределяющим подсчетом

```
max := arr[0]
min := arr[0]
for i := 1 to i + 1 {
  if arr[i] < min {
    countCompare++;
    min := arr[i];
  }
  if arr[i] > max {
    countCompare++;
    max := arr[i];
  }
}

contingSort(arr, min, max);

PROCEDURE contingSort(int[] arr, int min, int max) {
  int[] count := new int[max - min + 1] // Массив счетчиков

  // Кол-во раз встречается каждое число
  for i := 0 to i + 1) {
    countCompare++
    count[arr[i] - min]++;
  }

  index := 0

  // Пробегаем по всем счетчикам
  for i := 0 to i + 1) {
    transport++
    for j := 0 to j + 1) {
      arr[index++] := i + min;
      transport++;
    }
  }

}
```

14. Сортировка естественным двухпутевым слиянием

```
s := false
do
```

```
{
        if(s = 0) then
        {
                i := 1
                j := N
                k := N + 1
                l := 2N
        }
        else
        {
                i := N + 1
                j := 2N
                k := 1
                l := N
        }
d := 1
f := 1
doAgain{
        if(K[i] > K[j]) then
        {
                R[k] := R[j]
                k := k + d
                Dec(j)
                if(K[j+1] <= K[j])
                        go to doAgain
                do
                {
                        R[k] := R[i]
                        k := k + d
                        Inc(i)
                } while(K[i-1] <= K[i])
        }
        else
        {
                if(i=j) then
                {
                        R[k] := R[i]
                        if(f=0) then
                        {
                                s := 1 – s
                                go to doAgain
                        }
                        else break
```

```
                }
                R[k] := R[i]
                k := k + d
                if(K[i-1] <= K[i]) then go to doAgain
                do
                {
                        R[k] := R[j]
                        K := k + d
                        Dec(j)
                } while(K[j+1] <= K[i])
        }
        f := 0
        d := !d
        k <-> l
        go to doAgain
} while(f)
if(s = 0) then (R[1],…,R[2N]) <- (R[N+1],…,R[2N])
```

16. Сортировка слиянием списков для естественного двухпутевого слияния

```
Function Merger(a, low, p ,l)
{
        for i := low to low+1
                x1[i-low] := a[i]
        for i := 0 to p-1
                x2[i] := a[i+l+low]
        s := (x1[1]=(x2[p-1]=(a[low+l-1]>a[p-1])?
        (a[low+l-1]+1):(s[p-1]+1)))
        i=(j=0)
        k := low
        while(x1[i] < s || x2[j] < s) do
        {
                if(x1[i] < x2[j] then a[k] := x1[i++]
                else
{
a[k] := x2[j++]
k++
                }
                return a
}

Smerger(a, m, n)
```

```
{
        l := 1
        while ( l <= (n-m)) do
        {
                low := m
                p := m – 1
                while(l+p < n) do
                {
                        p := (low +2*l-1 < n)?(low2*l-1):n
                        Merger(s, low, up ,l)
                        low := p – 1
                }
l*=2
}
return a
}
```

Поиск?	1000		2000		5000	
Кол-во	Срав	Пересылка	Срав	Пересылка	Срав	Пересылка
Бин встав.	8547	250678	19120	1000874	54442	6250511
Шелла.	13667	8645	32336	20903	99786	66861
Пузырьк.	498586	249686	1997041	998887	12492672	6245543
Шейкер.	499501	249687	1999005	998888	12497528	6245546
Быстр рекурс.	11002	7117	24801	15587	71131	43539
Прост выбор.	499500	1000	1999001	2001	12497506	5006
Расп подсчет.	1011	2996	2013	5996	5015	14996
Ест д.п слиян.	6937	8117	15842	18236	48688	56725
Слиян списков прост д.п.	29928	9976	65856	21952	185424	61808

Вывод: в ходе данной лабораторной работы я ознакомился с методами сортировки, получил практические навыки программирования задач сортировки и навыки экспериментальных исследований алгоритмов.