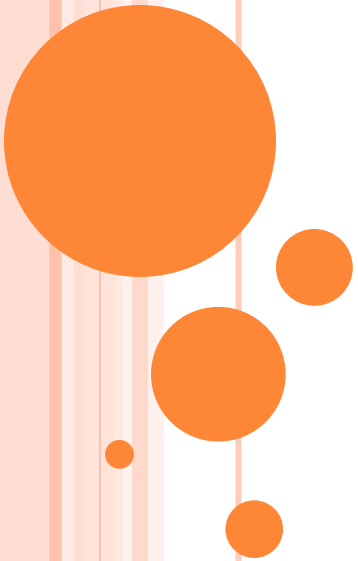
Сортировки



Определение

Сортировка — это алгоритм для упорядочивания элементов в массиве.

Цель сортировки – упорядочить информацию и облегчить поиск требуемых данных

Определение

Практически каждый алгоритм сортировки можно разбить на три части:

- сравнение двух элементов, определяющее упорядоченность этой пары;
- перестановку, меняющую местами неупорядоченную пару элементов;
- сортирующий алгоритм, определяющий выбор элементов для сравнения и отслеживающий общую упорядоченность массива данных.

Определение

Сортировка данных в оперативной памяти называется внутренней, а сортировка данных в файлах называется внешней

Пузырьковая сортировка

- Просматривая массив с первого элемента, найти минимальный и поменять его местами с первым элементом.
- Просматривая массив со второго элемента, найти минимальный и поменять его местами со вторым элементом.
- И, так далее, до последнего элемента.

Пузырьковая сортировка

```
for(i=0;i<k-1;++i) // выбор верхней границы массива
   for(j=k-1;j>i;--j) // просмотр массива ”снизу” ” вверх”
   {
               if(ms[j-1]>ms[j])  //  условие  замены выполнено
      {
      m=ms[j-1]; // замена j-1 и j элементов
      ms[j-1]=ms[j];
      ms[j]=m;
      }
   }
```

Сортировка методом Шелла

- Сначала сортируются все элементы, отстоящие друг от друга на три позиции
- Затем сортируются элементы, расположенные на расстоянии двух позиций
- Наконец, сортируются все соседние элементы

Сортировка методом Шелла

```
for(gap = k/2; gap > 0; gap /= 2)
do {
     flg = 0;
     for(i = 0, j = gap; j < k; i++, j++)
      if(ms[i] > ms[j]) // сравниваем отстоящие на
gap   элементы
     {
    n = ms[j];
    ms[j] = ms[i];
    ms[i]= n;
    flg = 1; // есть еще не рассортированные данные
     }
} while (flg); // окончание этапа сортировки
```

Сортировка вставками

- Упорядочиваются два элемента массива
- Вставка третьего элемента в соответствующее место по отношению к первым двум элементам.
- Этот процесс повторяется до тех пор, пока все элементы не будут упорядочены.

Сортировка вставками

```
for(i = 1; i < n; i++)
{
   value = a[i]; значение предыдущего элемента
   for (j = i - 1; j >= 0 && a[j] > value; j--)
   {
      a[j + 1] = a[j];
   } сдвиг всех элементов направо
   a[j + 1] = value; запись в освободившийся или в
тот же элемент
}
```

Сортировка выбором

- Выбирается элемент с наименьшим значением и делается его обмен с первым элементом массива.

- Затем находится элемент с наименьшим значением из оставшихся n-1 элементов и делается его обмен со вторым элементом и т.д. до обмена двух последних элементов.

Сортировка выбором

```
for(i=0; i<k-1; i++) // выбор исходного элемента к сравнению
{ i1=i;
for(j=i+1; j<k; j++) // просмотр массива ”снизу” ”вверх”
if(ms[i1]>ms[j])  i1=j;  //  фиксируем  координату элемента в массиве
m=ms[i]; // замена i1 и i элементов
ms[i]=ms[i1];
ms[i1]=m;
}
```

Быстрая сортировка

Сортировка Хоара (1960) до сих пор одна из самых быстрых.
Быстрая сортировка относится к алгоритмам «разделяй и властвуй».
QuickSort является существенно улучшенным вариантом алгоритма сортировки пузырьком

Алгоритм

1. Выбрать опорный элемент из массива.
2. Перераспределить элементы в массиве так, чтобы меньшие опорного были перед ним, а большие или равные после.
3. Применить первые два шага к двум подмассивам слева и справа от опорного элемента.

Алгоритм выбора опорного

1.За опорный выбирается средний
2. Два индекса один в начале массива, другой в конце
3. Индексы приближаются друг к другу, пока не найдётся пара элементов, где один больше опорного и расположен перед ним, а второй меньше и расположен после.
4. Эти элементы меняются местами.
Обмен происходит до тех пор, пока индексы не пересекутся. Алгоритм возвращает последний индекс

```
void hoar(int *ms, int l, int r)
{
int i, j, t;
int sr = ms[(l + r) / 2];
i = l; j = r;
do{
	while (ms[i] < sr) i++; // ищем слева элемент больше среднего
	while (ms[j] > sr) j--; // ищем справа элемент меньше среднего
	if (i <= j) // если левая граница не прошла за правую
	{
		t = ms[i];
		ms[i] = ms[j];
		ms[j] = t;
				i++; j--;
	}
} while (i <= j); // пока границы не совпали

if (i < r)
	hoar(ms, i, r);
if (j>l)
	hoar(ms, l, j);
}
```

Сортировка в строке

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]

```
for (i = 0; i < M; i++)
    for (j=0;j<M; j++)
        if (matr[1][j] < matr[1][i])
        {
            int temp = matr[1][j];
            matr[1][j] = matr[1][i];
            matr[1][i] = temp;
```

Сортировка в столбце

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]

```
for (i = 0; i < n; i++)
      for (j=0;j<n; j++)
            if (matr[i][2] < matr[j][2])
            {
                  int temp = matr[i][2];
                   matr[i][2] = matr[j][2];
                  matr[j][2] = temp;
            }
```

Сортировка строк

1. Найти признак для i-ой и j-ой строк
2. Обменять все элементы в этих строках

```
for (i = 0; i<n;i++)
    for (j = 0; j<n;j++)
      //вычисление признаков сравнения
      if(признак i - ой > признак j - ой)
            for(k = 0; k<m; k++)
            {
                  int t = matr[i][k];
                  matr[i][k] = matr[j][k];
                  matr[j][k] = t;
            }
```