

排序算法

1. 冒泡排序 (Bubble Sort)

□

```
void bubble(int a[],int length){
    for(int i=length-1;i>=1;i--){        //n-1轮冒泡, n-1,n-2,...,1
        for(int j=0;j<i;j++){            //冒泡的次数0,1,...,i-1
            if(a[j]>a[j+1]){                //如果前面的大于后面的交换(从小到大)
                int tmp = a[j];
                a[j] = a[j+1];
                a[j+1] = tmp;
            }
        }
    }
}
```

2. 选择排序 (Selection Sort)

□

```
void selection(int a[],int length){
    for(int i=0;i<length;i++){            //找位置
        int k=i;
        for(int j=i+1;j<length;j++){      //找包括i在内的最小的
            if(a[j]<a[k])
                k = j;
        }
        if(k!=i){                          //把找到的最小的交换到i位置
            int temp = a[i];
            a[i] = a[k];
            a[k] = temp;
        }
    }
}
```

3. 插入排序 (Insertion Sort)

□

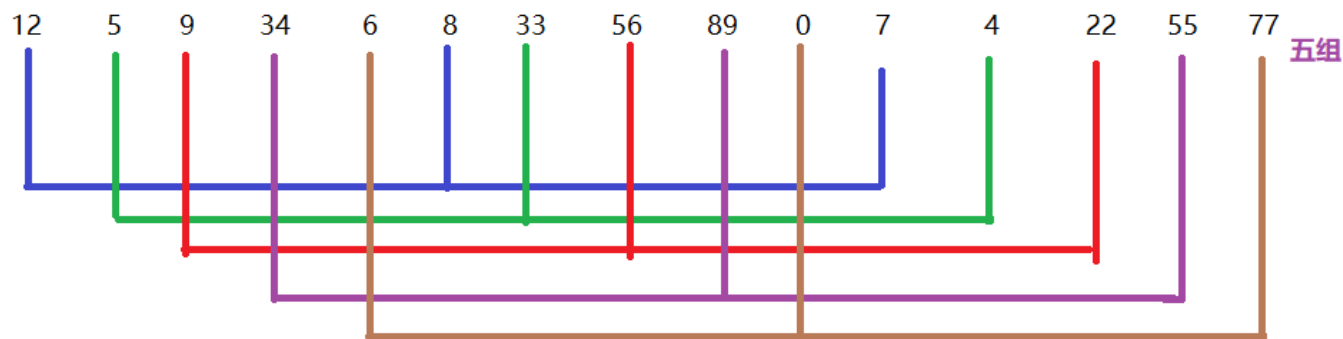
```
void insertion(int a[],int length){
    int j,k;
    for(int i=0;i<length;i++){
        for(j=i-1;j>=0;j--){
            if(a[j]<a[i])
                break;
        }
        if(j!=i-1){
            int temp=a[i];
            for(k=i-1;k>j;k--)
                a[k+1] = a[k];
            a[k+1] = temp;
        }
    }
}
```

4. 希尔排序 (Shell Sort)

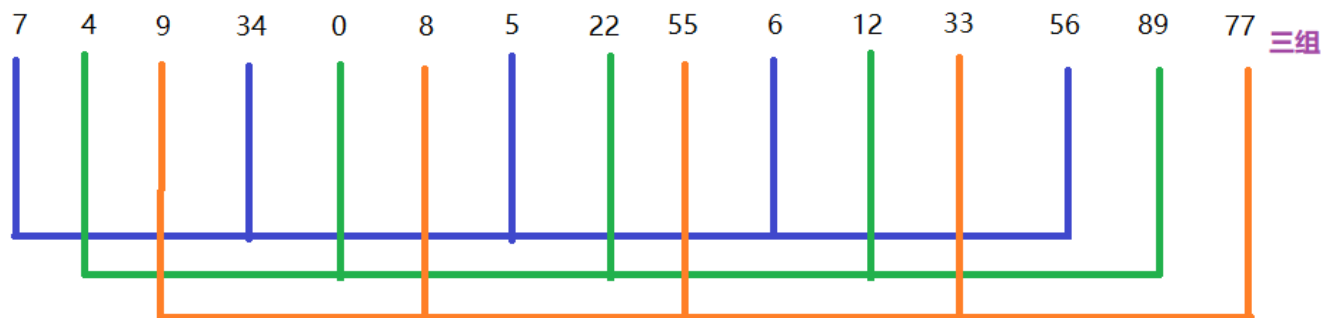
□

shell排序实际上是一种直接插入排序推广，其基本原理为其先将一组数分成若干组；此处应该注意，分组的方式不能几个几个紧挨着分组，而是采用每次所分组数均为素数且最后一次分组为1的方法。采用分组的好处是，在每次排序完后都是将小的数尽量往前面赶，大的数尽量往后面赶，最后一次排序直接采用直接插入排序。运用到了直接插入排序越有序有快的特性。

例如12、5、9、34、6、8、33、56、89、0、7、4、22、55、77的排序步骤如下：



第一次排序结果:



第二次排序结果:

5 0 8 6 4 9 7 12 33 34 22 55 56 89 77

分为一组,直接插入排序所得结果:

0 4 5 6 7 8 9 12 22 33 34 55 56 77 89

```
void shell(int a[],int length){
    int drr[] = {5,3,1};
    int lend = sizeof(drr)/sizeof(drr[0]);
    for(int i=0;i<lend;i++){
        int gap = drr[i];
        for(int j = gap;j<length;j++){
            int temp = a[j];
            int k = 0;
            for(k = j-gap;k >= 0;k-=gap){
                if(a[k] > temp)
                    a[k+gap] = a[k];
                else
                    break;
            }
            a[k+gap] = temp;
        }
    }
}
```

5. 快速排序 (Quick Sort)

□

```
void quick(int a[],int left,int right){
    int mid,l=left,r=right,temp;
```

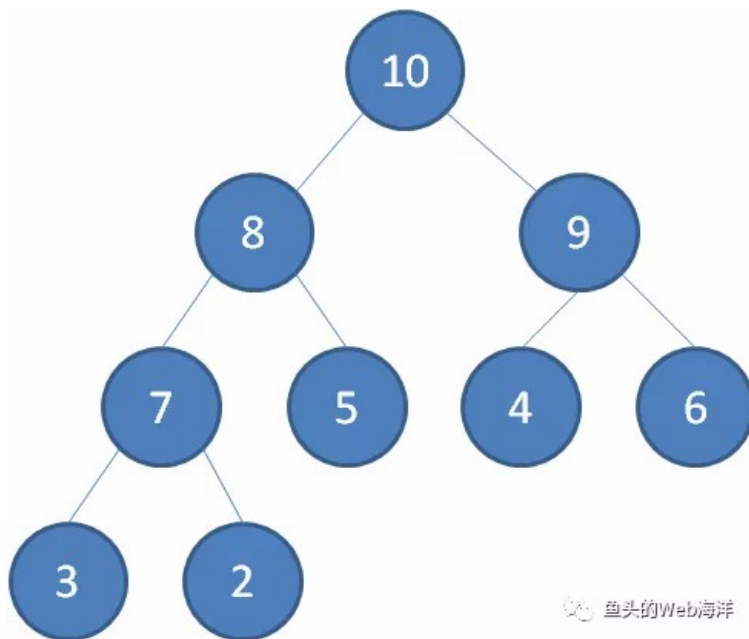
```
mid = a[(l+r)/2];
do{
    while(a[l]<mid) l++;    //在左半部找比mid大的数
    while(a[r]>mid) r--;    //在右半部找比mid小的数
    if(l<=r){
        temp = a[l];
        a[l] = a[r];
        a[r] = temp;
        l++;
        r--;
    }
}while(l<=r);

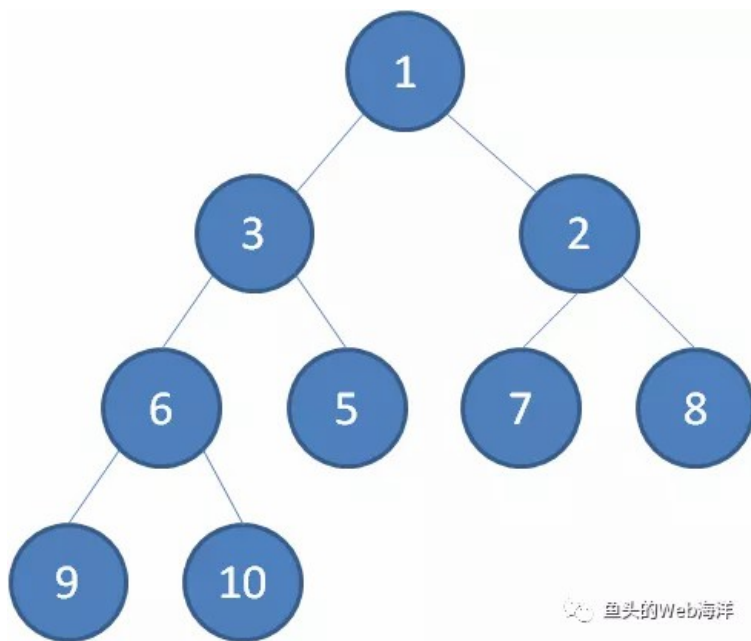
if(left<r) quick(a,left,r);
if(l<right) quick(a,l,right);
}
```

6. 并归排序 (Merge Sort)

□

7. 堆排序 (Heap Sort)





□

8. 计数排序 (Counting Sort)

□

9. 桶排序 (Bucket Sort)

□

```
void bucket(int a[],int length,int maxn){
    int b[maxn+1];
    memset(b,0,sizeof(b));
    for(int i=0;i<length;i++)
        b[a[i]]++;
    int p=0;
    for(int i=0;i<=maxn;i++){
        while(b[i]>0){
            a[p++] = i;
            b[i] --;
        }
    }
}
```

10. 基数排序 (Radix Sort)

□

11. 特别的冒泡排序——鸡尾酒排序 (Cocktail Sort)

□

12. 排序算法比较

1. $O(n^2)$ 的排序算法
 - 冒泡排序
 - 选择排序
 - 插入排序
 - 希尔排序

2. $O(n \log n)$ 的排序算法

- 并归排序
- 快速排序
- 堆排序

3. 线性的排序算法

- 计数排序
- 桶排序
- 基数排序

名称	数据对象	稳定性	时间复杂度		额外空间复杂度
			平均	最坏	
冒泡排序	数组	✓	$O(n^2)$		$O(1)$
选择排序	数组	✗	$O(n^2)$		$O(1)$
	链表	✓			
插入排序	数组、链表	✓	$O(n^2)$		$O(1)$
堆排序	数组	✗	$O(n \log n)$		$O(1)$
归并排序	数组	✓	$O(n \log^2 n)$		$O(1)$
			$O(n \log n)$		$O(n) + O(\log n)$ 如果不是从下到上
	链表				$O(1)$
快速排序	数组	✗	$O(n \log n)$	$O(n^2)$	$O(\log n)$
希尔排序	数组	✗	$O(n \log^2 n)$	$O(n^2)$	$O(1)$
计数排序	数组、链表	✓	$O(n + m)$		$O(n + m)$
桶排序	数组、链表	✓	$O(n)$		$O(m)$
基数排序	数组、链表	✓	$O(k \times n)$	$O(n^2)$	🐟 鱼头的Web海洋