阅读：3,605　　作者：解学武

归并排序算法及其C语言具体实现

本节介绍一种不同于插入排序和选择排序的排序方法——归并排序，其排序的实现思想是先将所有的记录完全分开，然后两两合并，在合并的过程中将其排好序，最终能够得到一个完整的有序表。

例如对于含有 n 个记录的无序表，首先默认表中每个记录各为一个有序表（只不过表的长度都为 1），然后进行两两合并，使 n 个有序表变为 [n/2] 个长度为 2 或者 1 的有序表（例如 4 个小有序表合并为 2 个大的有序表），通过不断地进行两两合并，直到得到一个长度为 n 的有序表为止。这种归并排序方法称为：2-路归并排序。

例如对无序表 {49，38，65，97，76，13，27} 进行 2-路归并排序的过程如图 1 所示：

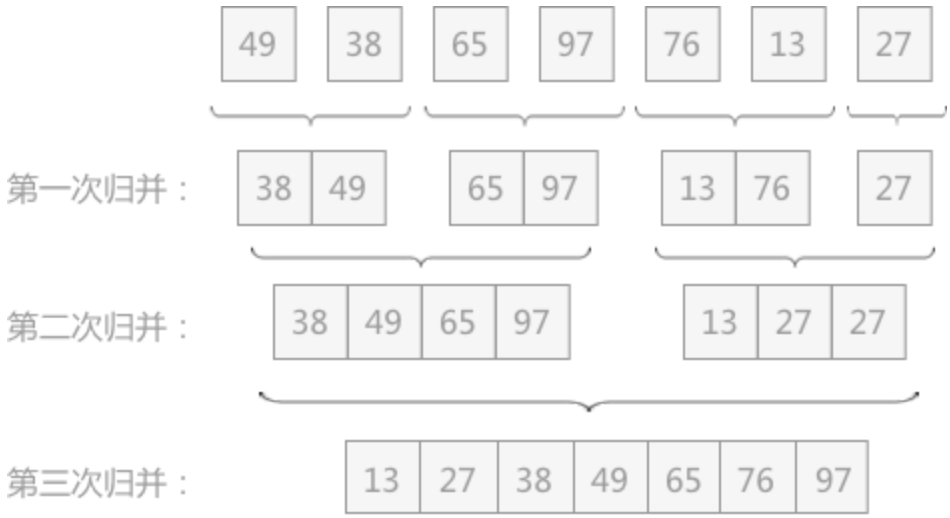

图 1 归并排序过程

> 归并过程中，每次得到的新的子表本身有序，所以最终得到的为有序表。

2-路归并排序的具体实现代码为（采用了递归的思想）：

```
01.  #include <stdio.h>
02.  #include <stdlib.h>
03.  #define  MAX 8
04.  typedef struct{
05.      int key;
06.  }SqNode;
07.  typedef struct{
08.      SqNode r[MAX];
09.      int length;
10.  }SqList;
```

```
//SR中的记录分成两部分：下标从 i 至 m 有序，从 m+1 至 n 也有序，此函数的功能是合二为一至TR数组中，使整
void Merge(SqNode SR[],SqNode TR[],int i,int m,int n){
    int j,k;
    //将SR数组中的两部分记录按照从小到大的顺序添加至TR数组中
    for (j=m+1,k=i; i<=m && j<=n; k++) {
        if (SR[i].key<SR[j].key) {
            TR[k]=SR[i++];
        }else{
            TR[k]=SR[j++];
        }
    }
    //将剩余的比目前TR数组中都大的记录复制到TR数组的最后位置
    while(i<=m) {
        TR[k++]=SR[i++];
    }
    while (j<=n) {
        TR[k++]=SR[j++];
    }
}

void MSort(SqNode SR[],SqNode TR1[],int s,int t){
    SqNode TR2[MAX];
    //递归的出口
    if (s==t) {
        TR1[s]=SR[s];
    }else{
        int m=(s+t)/2;//每次递归将记录表中记录平分，直至每个记录各成一张表
        MSort(SR, TR2, s, m);//将分开的前半部分表中的记录进行排序
        MSort(SR,TR2, m+1, t);//将后半部分表中的记录进行归并排序
        Merge(TR2,TR1,s,m, t);//最后将前半部分和后半部分中的记录统一进行排序
    }
}
//归并排序
void MergeSort(SqList *L){
    MSort(L->r,L->r,1,L->length);
}

int main() {
    SqList * L=(SqList*)malloc(sizeof(SqList));
    L->length=7;
    L->r[1].key=49;
    L->r[2].key=38;
    L->r[3].key=65;
    L->r[4].key=97;
    L->r[5].key=76;
    L->r[6].key=13;
    L->r[7].key=27;
```

```
58.      MergeSort(L);
59.      for (int i=1; i<=L->length; i++) {
60.          printf("%d ",L->r[i].key);
61.      }
62.      return 0;
63.  }
```

运行结果为：

```
13 27 38 49 65 76 97
```

> 提示：归并排序算法在具体实现时，首先需要将整个记录表进行折半分解，直到分解为一个记录作为单独的一张表为止，然后在进行两两合并。整个过程为分而后立的过程。

总结

归并排序算法的时间复杂度为 `O(nlogn)` 。该算法相比于堆排序和快速排序，其主要的优点是：当记录表中含有值相同的记录时，排序前和排序后在表中的相对位置不会改变。

例如，在记录表中记录 a 在记录 b 的前面（记录 a 和 b 的关键字的值相等），使用归并排序之后记录 a 还在记录 b 的前面。这就体现出了该排序算法的稳定性。而堆排序和快速排序都是不稳定的。