


归并排序

基本思想：通过“归并”两个或两个以上的记录的有序子序列，逐步增加记录有序序列的长度

假设初始的序列含有 n 个记录，可以看成 n 个有序的子序列，每个子序列的长度为1，然后两两归并，得到 $\lceil n/2 \rceil$ 个长度为2的有序子序列；再两两归并，如此重复直到得到一个长度为 n 的有序序列为止。这种排序方法称为二路归并排序

举例：


初始关键字：
[49] [38] [65] [97] [76] [13] [27]



一趟归并后：
[38 49] [65 97] [13 76] [27]



二趟归并后：
[38 49 65 97] [13 27 76]



三趟归并后：
[13 27 38 49 65 76 97]

思考：如何归并两个有序表呢？

两组归并算法

二趟归并后: [38 49 65 97] [13 27 76]

```
215 void merge(RecordNode *r, RecordNode *r1, int low, int m, int high)
216 {
217     int i, j, k;
218     i=low; j=m+1; k=low;
219     while((i<=m)&&(j<=high))
220     {
221         if(r[i].key<=r[j].key) r1[k++]=r[i++];
222         //从两个有序文件中选择小的记录放到新的文件中
223         else r1[k++]=r[j++];
224     }
225     while(i<=m) r1[k++]=r[i++]; //复制第一个文件的剩余部分记录
226     while(j<=high) r1[k++]=r[j++]; //复制第二个文件的剩余部分记录
227 }
```

一趟归并算法

有一个子文件长度小于length: [38 49] [65 97] [13 76] [27]

只有一个子文件: [38 49] [65 97] [13 76] [27 90] [30]

```
229  /*一趟归并排序，结果放到r1中*/
230  void mergePass(RecordNode *r,RecordNode *r1,int n,int length)
231  {
232      int j,i=0;
233      while(i+2*length-1<n)//本趟归并的有序子文件长度
234      {
235          merge(r,r1,i,i+length-1,i+2*length-1);
236          i+=2*length;
237      }
238      if(i+length-1<n-1)/*剩下两个子文件，但一个长度小于length*/
239          merge(r,r1,i,i+length-1,n-1);
240      else
241          for(j=i;j<n;j++)/*剩下一个子文件了*/
242              r1[j]=r[j];
243  }
```

二路归并算法

```
245 void mergeSort(SortObject *pvector)
246 {
247     RecordNode record[num];
248     int length=1;
249     while(length<pvector->n)
250     {
251         mergePass(pvector->record,record,pvector->n,length);
252         length*=2;
253         mergePass(record,pvector->record,pvector->n,length);
254         length*=2;
255     }
256 }
```

算法分析:

时间复杂度: 要进行 $\lceil \log_2 n \rceil$ 趟的排序, 每一趟的时间消耗是 $O(n)$, 故时间复杂度 $T(n) = O(n \log_2 n)$

空间复杂度: 增加了record数组的空间 $S(n) = O(n)$

稳定性: 二路归并算法是稳定的