**归并排序**

**归并排序是建立在归并操作上的一种有效的排序算法。算法是采用了分治法的一个非常典型的应用。归并排序是讲已经排序的表合并成一个表。**

**基本思想：**

**设两个有序的子序列(相当于输入序列)放在同一序列中相邻的位置上：array[low..m]，array[m + 1..high]，先将它们合并到一个局部的暂存序列 temp (相当于输出序列)中，待合并完成后将 temp 复制回 array[low..high]中，从而完成排序。**

**在具体的合并过程中，设置 i，j 和 p 三个指针，其初值分别指向这三个记录区的起始位置。合并时依次比较 array[i] 和 array[j] 的关键字，取关键字较小（或较大）的记录复制到 temp[p] 中，然后将被复制记录的指针 i 或 j 加 1，以及指向复制位置的指针 p加 1。重复这一过程直至两个输入的子序列有一个已全部复制完毕(不妨称其为空)，此时将另一非空的子序列中剩余记录依次复制到 array 中即可。**

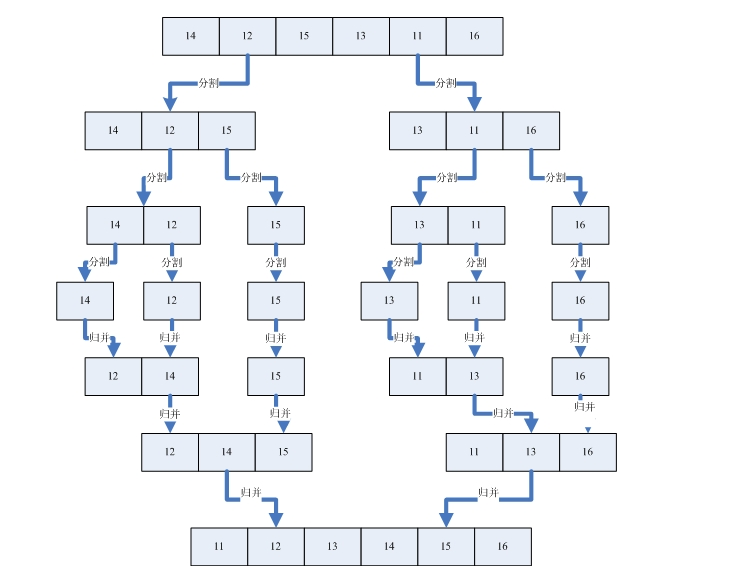
**若将两个有序表合并成一个有序表，称为2-路归并。**

**举例：**

**假设有数组int a[] = {-1,14,12,13,11,16}；-1下标为0的位置，我们不参与排序。不适用**

**有效元素的个数为5个。假设我们有一个没有排好序的序列，那么首先我们使用分割的办法将这个序列分割成一个个已经排好序的子序列。然后再利用归并的方法将一个个的子序列合并成排序好的序列。分割和归并的过程可以看下面的图例。**

**先分割，在合并**



**源代码实现：**

**#define MAX 6**

**int sr[ ] = {-1,14,12,15,13,11,16};**

**int tr1[7] = {0}；**

**msort(sr,tr1,1,6);**

**void msort(int sr[],int tr1[],int s,int t)**

**{**

**int m;**

**int TR2[MAX + 1]; //下标为0的位置不填充数据**

**if(s == t)**

**{**

**TR1[s] = SR[s];**

**}else{**

**m = (s + t) / 2;**

**Msort(SR,TR2,s,m); //把数组分为1-m左边的部分**

**Msort(SR,TR2,m + 1,t); //把数组分为m + 1 –t右边的部分**

**Merge(TR2,TR1,s,m,t); //对划分好的组进行排序。**

**}**

**}**

**void merge(int sr[],int tr[],int i,int m,int n)**

**{**

**int j,k,l;**

**//找到较小的填充到tr数组中**

**for(j = m + 1,k = i;i <= m && j <= n;k++)**

**{**

**if(SR[i] < SR[j])**

**{**

**TR[k] = SR[i++];**

**}else{**

**TR[k] = SR[j++];**

**}**

**}**

**把剩下的数据填充到tr数组中即可。**

**if(i <= m)**

**{**

**for(l = 0;l <= m - i;l++)**

**{**

**TR[k + 1] = SR[i + l];**

**}**

**}**

**if(j <= n)**

**{**

**for(l = 0;l <= n - j;l++)**

**{**

**TR[k + l] = SR[i + l];**

**}**

**}**

**}**