1. 归并排序

以基本的2-路归并排序作为例子来介绍归并排序。

（一）2-路归并排序原理

序列中共n个数，将序列两两分组，分为 [n / 2]个组，组内进行单独排序，然后将这些组两两归并，每两个组合并成一个组；于是生成了[n / 4]个组，组内依然单独排序；重复同样的操作，直到合并成一个组为止。时间复杂度为O(nlogn)。

（二）2-路归并排序实例讲解

序列： 12 ，33， 32， 23， 55， 43， 23， 21， 55

1. 第一次归并

得到5组：【12， 33】， 【32， 23】， 【55， 43】， 【23， 21】， 【55】

组内排序：【12， 33】， 【23， 32】， 【43， 55】， 【21， 23】， 【55】

1. 第二次归并

两两归并，得到3组：【12， 33， 23， 32】， 【43， 55， 21， 23】， 【55】

组内排序：【12，23, 32， 33】，【21， 23， 43， 55】， 【55】

1. 第三次归并

得到2组：【12， 23， 32， 33， 21， 23， 43， 55】， 【55】

组内排序：【12， 21， 23， 23， 32， 33， 43， 55】， 【55】

4．第四次归并

得到1组：【12， 21， 23， 23， 32， 33， 43， 55， 55】

组内排序：【12， 21， 23， 23， 32， 33， 43， 55， 55】

1. 算法实现
2. 思路：

归并排序，就是将两个有序序列合并成一个有序序列，并且需要不停地递归。将序列分为左右两半，对两个子区间分别递归归并排序，然后合并两个有序的子区间为有序序列即可。

1. 代码实现

const int maxn = 100;

//归并两个区间[l1, r1], [l2, r2]

void merge(int a[], int l1, int r1, int l2, int r2) {

int i = l1, j = l2;

int temp[maxn], index = 0;

while (i <= r1 && j <= r2) {

if (a[i] <= a[j]) {

temp[index++] = a[i++];

}

else {

temp[index++] = a[j++];

}

}

while (i <= r1) {

temp[index++] = a[i++];

}

while (j <= r2) {

temp[index++] = a[j++];

}

for (int i = 0; i < index; ++i) {

a[l1 + i] = temp[i];

}

}

//用归并排序一个序列

void mergeSort(int a[], int left, int right) {

if (left < right) {

int mid = (left + right) / 2;

//递归

mergeSort(a, left, mid);

mergeSort(a, mid + 1, right);

//归并

merge(a, left, mid, mid + 1, right);

}

}