**使用顺序表实现有序表合并**

61520518 谢鹏飞[[1]](#footnote-1)

1. **作业要求**
   * 使用顺序表实现有序表合并
     + 生成两个集合La，Lb（无重复元素）
     + 对La、Lb按着升序排序（使用学过的方法）
     + La、Lb合并，存储在新表Lc（升序）
     + 分析时间与空间复杂度
2. **主体思路**

首先构建一个顺序（1～n）数组，长度为La与Lb长度之和，利用库函数swap对数组中的元素进行充分多次数的互换位置，使得整个数组充分随机，并且并不重复。

接着将整个数组拆分为La和Lb，利用Quick-Sort算法进行升序排序。

然后将La，Lb合并，利用Merge-Sort的思想，利用两个指标进行递增，比较元素值，得到新表Lc。

1. **核心代码**
   * **建立随机数组**

//generate a non-repeatable array

int\* arr = new int[a + b];

for (int i = 0; i < a + b; i++) //arr:{1, 2, 3, 4, ..., a + b}

arr[i] = i;

for (int i = 0; i < a + b; i++) { //randomly swap these numbers

int indicator;

if (i == 0) indicator = 1;

else indicator = rand() % i;

swap(arr[i], arr[indicator]);

}

* + **快速排序**

void Sort(int\* A, int l, int r, int\* T) {

if (r - l > 1) {

int m = (l + r) / 2;

int p = l, q = m, i = l;

Sort(A, l, m, T);

Sort(A, m, r, T);

while (p < m || q < r) {

if (q >= r || (p < m && A[p] <= A[q]))

T[i++] = A[p++];

else T[i++] = A[q++];

}

for (int i = l; i < r; i++) A[i] = T[i];

}

}

* + **合并&排序**

//merge this two

int p = 0, q = 0;

int\* temp = new int[a + b];

int\* Lc = new int[a + b];

while (p <= a && q <= b) {

if (p == a || (q < b && La[p] >= Lb[q]))

temp[p + q] = Lb[q++];

else

temp[p + q] = La[p++];

}

1. **结果展示与分析**

最终得到的是一个顺序排序的数组，实验结果符合预期。

快速排序的时间复杂度是飘忽不定的，平均结果是O(nlgn)，最坏结果是O(n^2).

空间复杂度则是O(lgn),这是由于每次调用的空间都是一个定长，而总调用次数取决于递归的深度，类似一个二叉树，递归深度近似为log\_2^n.

1. **总结**

快速排序的做法在下面的合并两个数组的时候也有借鉴，这种算法相对来说是比较高效率的了。随机数组的建立过程不知道别人是否有更好的做法，我自己写的这种相对来说还是有一点局限性的，只能生成随机的整数数组（当然随机pop掉就可以制造出随机的、非连续的整数数组，再随机用两个这样的数组相除就可以得到随机的float数组，但是这样确定所生成的随机数的范围还是有些麻烦的）。

Ps: 老师，以后我可以还是用LaTeX仿照这个模板写吗？... 感觉无论是公式还是代码呈现效果，还是LaTeX美观一些。

1. 来自吴健雄学院615205班. Email: me@panoan.org [↑](#footnote-ref-1)