第四题

问题描述：

给定一个升序排列好的实数数组，找到目标数值的在实数数组中出现的起始位置和末尾位置。如果目标数值没有出现在实数数组中就返回[-1,-1]。算法复杂度应限制在O(logn)。

算法描述：

引入新的结构——下标。使用二分法对实数数组下标进行划分，将原始数组划分为left，mid和right三部分，接着将目标值与mid进行比较，目标值大于mid则利用right部分递归调用查找算法；目标值小于mid则利用left部分递归调用查找算法；目标值等于mid则返回当前的下标。之后通过对目标数值下标左右分别在次小的实数数组上查找确定该数值对应的starting和ending下标值。

伪代码：

|  |
| --- |
| SEARCH（List，Target）  1：left = 0;right = List‘s length-1;  2：if List’s length is 0 then  3： if List’s value is equal to Target then  4： return (left + right) / 2, (left + right) / 2;  5： else  6： return [-1,-1];  7： end if  8：end if  9：location = (left + right) / 2;  10：middle = List\_s(Where List\_s is a deep copy of List);  11：if middle > Target then  12： right = location – 1;  13： return SEARCH(left part of List, Target);  14：end if  15：if middle < Target then  16： left = location + 1;  17： return SEARCH(right part of List, Target);  18：end if  19：if middle == Target then  20： starting = find the left bound of Target from the existing List;  21： ending = find the right bound of Target from the existing List;  20： return starting, ending;  21：end if |

子问题缩减示意图：

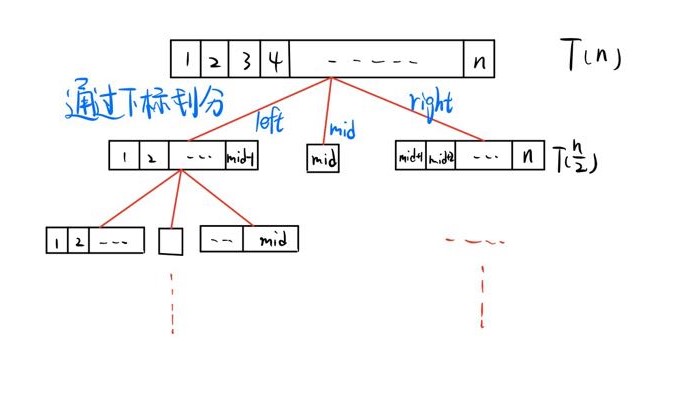


图 1子问题缩减示意图

证明算法的正确性：

由于给定实数数组是按照升序排列的，所以将该数组按照下标进行划分后，得到left，middle和right三部分，这三部分内部也是具有升序排列性质的。将目标数值与middle进行比较，根据升序的性质，如果目标数值小于middle则目标数值一定出现在left子数组中；如果目标数值大于middle则目标数值一定出现在right子数组中。

根据有序数组的性质，通过递归调用left或者right子数组一定能够唯一（或存在多个该数值时能找到其中一个的位置）确定目标数值在原始数组中的位置。因此该算法是可证明正确的。

分析算法的复杂度：

假设给定实数数组的长度为n。由于该算法每次将当前数组划分成三部分，且主要部分为left和right（通过下标数值进行划分），middle部分只包含一个数值并且仅用来比较，每次递归调用选择一侧的分支继续进行。因此可将该算法的两步之间操作近似地描述为T(n)=T(n/2)+O(1)。每步将目标数值与middle数值进行比较的复杂度为O(1)。

该算法在进行根据下标向下划分的时候，如果middle值恰好是目标数值则会提前停止，因此考虑最坏情况，该算法会将原始实数数组完全二分成n个实数，因此最坏情况共进行logn次划分。

则有

T(n) = T(n/2) + 1

= T(n/2^2) + 2

= T(n/2^3) + 3

…

= T(n/2^logn) + logn

=O(logn)

第五题

问题描述：

给定一个n个定点的凸多边形，我们可以将它划分成不同的小块，这里我们统一划分成三角形。当n等于4时，共有两种不同的划分方法；当n等于5时，共有五种不同的划分方法。寻找一个算法能够计算n个定点的凸多边形可以有多少种不同的三角形划分方法。

算法描述：

利用分而治之的思想，我们考虑一步一步将n个定点的凸多边形进行分割，即每次将n个定点的凸多边形分成两个（或三个）凸多边形。考虑最小的分割要求是三角形，而且三角形能够唯一确定三个顶点的一种划分方式，所以我们使用三角形对凸多边形进行划分。考虑首先将凸多边形的顶点进行标号以便后续表示{V1,V2,V3,…,Vn}。首先选定开始顶点为{V1,V2}，那么这两个顶点可以与剩下所有顶点集合中的一个进行连接并且唯一确定一个三角形划分，假设选择顶点Vi，，则原始凸多边形，被分割后为三部分----唯一确定的三角形{V1,V2,Vi}，{V2,V3,…,Vi}和{Vi,…Vn,V1}。那么除了三角形部分是唯一划分之外，得到了另外两部分为凸i-1边形和凸N-i+2边形。那么就可以递归调用，之后搜索全部可行的i以计算可行方案数量，其中已知凸三边形划分是1。

伪代码：

|  |
| --- |
| FindTriangle（n）  1：if n == 2 or n == 3 then  2： num = 1;  3： return num;  4：end if  5：for i = 3 to n do  6： num += FindTriangle(i - 1) \* FindTriangle(n – i + 2);  7：end for  8：return num; |

子问题缩减示意图：

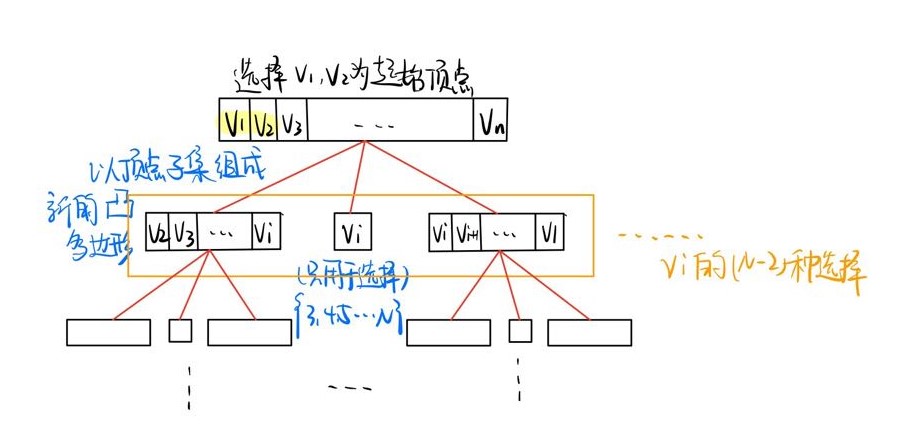


图 2子问题缩减示意图

证明算法的正确性：

通过上面的算法描述我们可以得知，该算法的正确性是建立在几何性质以及递归调用的基础上的，因此该算法是可证明正确的。

分析算法的复杂度：

由上面设计的算法，我们可以将其循环部分完全展开，得到算法公式如下：

T(n) =

= T(2)\*T(n-1) + T(3)\*T(n-2) + … + T(n-1)\*T(2)

由已知

T(2) = 1 = O(1)

T(3) = T(2) \* T(2) = T(2)^2

T(4) = T(2) \* T(3) + T(3) \* T(2) = T(2) \* T(2)^2 \* 2 = 2T(2)^3

T(5) = T(2) \* T(4) + T(3) \* T(3) + T(4) \* T(2) = T(2) \* 2T(2)^3 \* 2 + T(2)^2^2 = 5T(2)^4

由此归纳可知

T(n) = O(2^n)

第六题

问题描述：

给定一个由k个链表组成的列表，每一个链表已经为升序排列。找到一个O(knlogk)复杂度的算法使其能够将k个链表归并到一个升序链表。其中每个链表的长度是n。

算法描述：

采用分而治之的思想，参考归并排序算法（将数组按下标二次划分，并通过归并将子问题合并）。因此对于k个链表，我们可以使用二分的思想将链表按照下标的结构进行向下划分，当划分到最小单位时，每个分支是一个链表，这样一共划分了logk次。对于最小单位的两个链表进行归并整合时，从每个链表的起始位置开始比较合成一个新的链表，每次归并的复杂度为logn，共有k个链表需要进行归并，由下向上完成归并后即得到了一个升序的目标链表。

伪代码：

|  |
| --- |
| MERGECOMBINE（A , l , r）  1：if l < r then  2： m = (l + r) / 2;  3： MERGECOMBINE(A , l , m);  4： MERGECOMBINE(A , m + 1 , r);  5： MERGE(A , l , m , r);  6：end if  MERGE(A , l , m , r)  1：i = 0 ; j = 0;  2：for k = l to r do  3： if L[i] < R[j] then  4： A[k] 🡨 L[i];  5： i++;  6： if all element in L have been copied then  7： Copy the remainder elements from R into A;  8： break;  9： end if  10: else  11： A[k] 🡨 R[j];  12： j++;  13： if all element in R have been copied then  14： Copy the remainder elements from L into A;  15： break;  16： end if  17： end if  18：end for |

子问题缩减示意图：

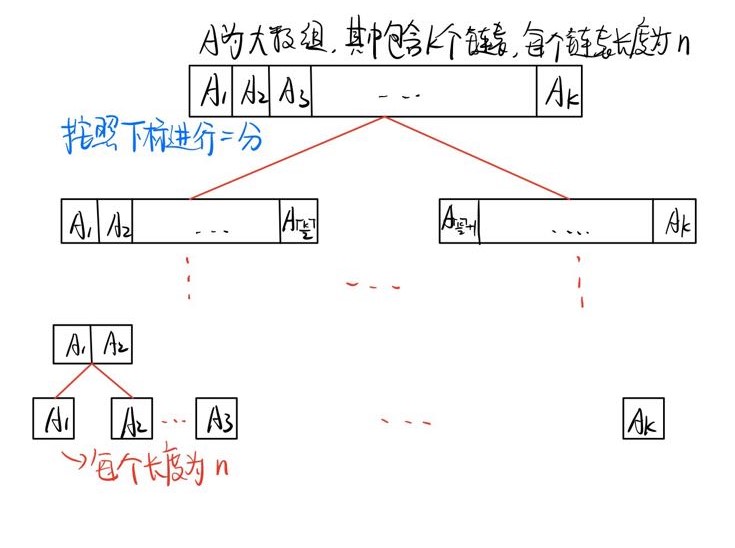


图 3子问题缩减示意图

证明算法的正确性：

基于循环不变量方法。

初始化：k = l。这时由于A[l,…,k-1]是空的，所以循环不变量保持当前状态。

维护阶段：假设L[i] < R[j]，并且A[l,…,k-1]保持在k-l小的元素。在将L[i]复制到A[k]中之后，A[l..k]将包含第k-l+1小的元素。

因此循环不变量仍然保持不变，并且在for循环中不断加入新的元素，这样的状态将持续到整个算法结束。

结束阶段：结束标志应该为k = r + 1。根据循环不变量原理形如A[l..k-1]的数组将一定会包含r-l+1小的元素，而且它们是有序（升序）排列的。

分析算法的复杂度：

由于是对列表中的链表个数进行二分，因此划分步骤的复杂度为T(k) = 2T(k/2)。归并的步骤我们首先考虑最简单的一步，即把两个n位的链表进行合并，这一步操作的时间复杂度是O(n)。不妨假设最简单的一步即两个链表中只包含了一个元素，那么当前算法将退化成归并排序算法，我们知道其时间复杂度为O(nlogn)，因此在简化的情形下我们此处的算法时间复杂度为O(klogk)。考虑实际过程中最简单一步的两个链表不是一个元素而是n位的链表，而这步操作是将归并排序算法进行线性扩展，因此对于实际的归并操作其时间复杂度应该是O(nk)，而一共完成了从下到上共logk次划分。因此，

T(k) = O(knlogk)