第5章 分治法

*Steven*

1.

①为一个分治算法编写伪代码，该算法同时求出一个元素数组的最大元素和最小元素的值。

②假设，为该算法的键值比较次数建立递推关系式并求解。

③请将该算法与解同样问题的蛮力算法进行比较。

①：

|  |
| --- |
| def max\_min(list, start, end):      if start == end:          return (list[start], list[start])  # 只有一个元素时，max=min，这么做也是为了应对数组长为奇数的情况      elif end-start ==1:          return (max(list[start], list[end]), min(list[start], list[end])) # 有两个元素时，max和min只需用蛮力法解决即可      else: #多个元素时分治，max是各部分max的最大，min是各部分min的最小          mid = (start+end)//2          (max\_l, min\_l) = max\_min(list, start, mid) # 左子数组的max和min          (max\_r, min\_r) = max\_min(list, mid+1, end) # 右子数组的max和min          return (max(max\_l, max\_r), min(min\_l, min\_r)) |

②：比较过程其实就是上述的max和min函数。此外由于，故不会出现上述的start==end的情况。

设长度为的数组的比较次数为，由于该算法采用分治，将查找数组分为左右两个子数组，且从子数组的结果到整个数组的结果还需分别比较一次max和min。所以综上可得递推公式为

可得通项公式为

③：

时间复杂度：

分治法：

蛮力法：

空间复杂度：

分治法：，因为不需要额外空间

蛮力法：，同样不需要额外空间

2. 我们也可以不用递归方法来实现合并排序，方法是先合并数组中的相邻元素，然后合并这些有序对，依此类推。写伪代码实现该自底向上的合并排序算法。

|  |
| --- |
| list=[]  def MergeSort(list)  for(length=2;length<=int(0.5len(list));length\*=2){  #每段的长度，从2增长到list长度的一半(向下取整)          for(start=0;start+length<=len(list);start+=length){  #得到每段的起始点              mid=int((start+length)/2) #中点位置             MergeList(list,start,mid-1,mid,start+length-1)  #调用已有的MergeList函数将两列表合并              }          }      return list |

3.

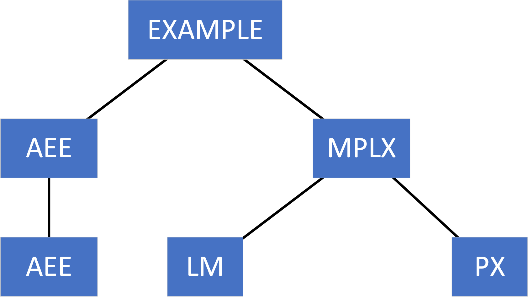
①应用快速排序对序列*E, X, A, M, P, L, E* 按照字母顺序排序，并画出相应的递归调用树。

②对于本章给出的快速排序的版本：

a. 一个所有元素都相等的数组，是该算法的最差输入还是最优输入？还是两者都不是？

* 1. b. 一个严格递减的数组，是该算法的最差输入还是最优输入？还是两者都不是？

①：



②：

a. 既不是最差也不是最优。该输入在本题所给的算法中既不满足a[i]<x也不满足a[j]>x，只会不断地执行swap(a[i], a[j])，无法退出循环。

b. 是最差输入。由于该情况下始终满足a[i]<x，在一层递归中最终会将长为n的数组划分为一个为空，另一个长为n-1的情况，规模缩小量最小。

4. 设计一个算法对个实数组成的数组进行重新排列，使得其中所有的负元素都位于正元素之前（算法需兼顾空间效率和时间效率）。分析算法的时间复杂度和空间复杂度。

伪代码：

|  |
| --- |
| def rerank(nums):      i = 0      j = len(nums) - 1      while i < j:          while i < j and nums[i] < 0:              i = i + 1          while i < j and nums[j] > 0:              j = j - 1          if i < j:              swap(nums, i, j)      return nums |

时间复杂度：无论输入情况如何，都需要遍历整个数组，所以时间复杂度为

空间复杂度：不需要额外空间，所以空间复杂度为