高级算法设计与分析作业3
分治

傅彦璋 23S003008

2024.5.3

1

采取类似二分查找的思路。先选择数组中间的一个元素a_{mid}，并检查其两边的元素以确定其出现次数。如果出现次数为1则算法终止并输出a_{mid}。如果出现次数不为1，那么这个元素将有序数组分为了左右两部分：小于a_{mid}的子数组和大于a_{mid}的子数组。我们递归地在长度为奇数的子数组查找。

Algorithm 1: 查找出现一次的元素

Data: 题目3.1中的有序整数数组$a_1, ..., a_n$
Result: 仅出现一次的元素

初始化$l \leftarrow 1$，$r \leftarrow n$;
while $l \le r$ **do**
  $mid \leftarrow \lfloor (l+r)/2 \rfloor$;
  在$a_{mid-1}, a_{mid}, a_{mid+1}$中统计$a_mid$出现的次数$t$;
  if $t = 1$ **then**
    return a_{mid}
  end
  记值为a_{mid}的元素为a_{mid1}和a_{mid2};
  if $mid1$是偶数 **then**
    $r \leftarrow mid1 - 1$;
  end
  else
    $l \leftarrow mid2 + 1$;
  end
end

显然（严格证明可参考作业8第1题）算法的时间复杂度是$O(\log n) \in o(n)$。

2

任意连续子序列B为包含A中位置i到位置j（$i \le j$）的元素的连续子序列，考虑对这个子序列求解。B中最大连续子序列的位置表示为$(f(i,j),g(i,j))$。B中包含位置j的最大连续子序列的位置表示为$(r(i,j),j)$。B中包含位置i的最大连续子序列的位置表示为$(i,l(i,j))$。

那么，对任意连续子序列B，如果我们已经得知B的左右两个连续子序列的f,g,l,r的值，我们就可以用动态规划的方式得到B的f,g,l,r的值。我们用动态规划的方式求得序列A对应的$f(1,n)$和$g(1,n)$，问题就解决了。于是有Algorithm2。

Algorithm2中，s可以通过线性时间进行求前缀和的预处理得到。

Longest_Continuous_Subsequence(1,n)输出的二元组即为题目中要求的使得$A[i]+...+A[j]$最大的i,j。

每次递归地调用Longest_Continuous_Subsequence(L,R)消耗的时间是$O(1)$的，调用次数是$O(n)$的。s的预处理时间是$O(n)$的。所以算法总体的时间复杂度是$O(n)$的。

Algorithm 2: Longest_Continuous_Subsequence(L,R)

Data: L和R，满足$L \le R$，用$s(i,j)$表示$A_i + ... + A_j$

Result: $f(L,R), g(L,R), l(L,R), r(L,R)$

if L=R **then**
  $f(L,R) \leftarrow L, g(L,R) \leftarrow L, l(L,R) \leftarrow L, r(L,R) \leftarrow L$;
  return $f(L,R), g(L,R)$
end
$mid \leftarrow \lfloor (L+R)/2 \rfloor$; /* 分治地计算左右两个子序列的f, g, l, r */
执行Longest_Continuous_Subsequence(L,mid);
执行Longest_Continuous_Subsequence($mid+1$,R);
$a \leftarrow s(f(L,mid), g(L,mid))$;
$b \leftarrow s(f(mid+1,R), g(mid+1,R))$;
$c \leftarrow s(r(L,mid), l(mid+1,R))$;
if $a \ge b, a \ge c$ **then** /* 计算f和g */
  $f(L,R) \leftarrow f(L,mid)$;$g(L,R) \leftarrow g(L,mid)$;
else if $b \ge a, b \ge c$ **then**
  $f(L,R) \leftarrow f(mid+1,R)$;$g(L,R) \leftarrow g(mid+1,R)$;
else
  $f(L,R) \leftarrow r(L,mid)$;$g(L,R) \leftarrow l(mid+1,R)$
end
if $l(L,mid) = mid$ *and* $s(mid+1, l(mid+1,R)) \ge 0$ **then** /* 计算l */
  $l(L,R) = l(mid+1,R)$
else
  $l(L,R) = l(L,mid)$
end
if $r(mid+1,R) = mid+1$ *and* $s(r(L,mid), mid) \ge 0$ **then** /* 计算r */
  $r(L,R) = r(L,mid)$
else
  $l(L,R) = l(mid+1,R)$
end
return $f(L,R), g(L,R)$

3

令集合$S' = S$，然后将集合S'中的每个元素s都改为$x-s$，得到新的集合S''。如果S和S''中有相同元素，那么S中存在两个元素的和为x，否则不存在。

Algorithm3的输出即为问题的答案。算法的时间瓶颈在于Algorithm4中的排序操作，故算法的时间复杂度为$O(n\log n)$。

Algorithm 3: 判断是否存在和为x的两个元素

Data: S和x

Result: “是”或“否”

$S_1 \leftarrow S$;

$S_2 \leftarrow \emptyset$;

for $a \in S_1$ **do**

  $S_2 = S_2 \cup \{x-a\}$;

end

return判断两集合是否有相同元素(S,S_2)

Algorithm 4: 判断两集合是否有相同元素(U,V)

Data: 实数集合U和实数集合V

Result: “是”或“否”

将集合U中元素升序排序得到序列$A = a_1, a_2, ..., a_n$;

将集合V中元素升序排列得到序列$B = b_1, b_2, ..., b_m$;

$i \leftarrow 1$;$j \leftarrow 1$;

while $i \leq n, j \leq m$ **do**

  if $a_i = b_j$ **then**

    return是

  else if $a_i < b_j$ **then**

    $i \leftarrow i+1$

  else

    $j \leftarrow j+1$

  end

end

return否

4

设计两个算法。

算法一：

将集合用$O(nm)$的时间代价生成出来。然后，对这个集合执行作业8题目4的线性时间算法，找到其中第k小元素。算法整体的时间复杂度为$O(nm)$。

算法二：

注意：该算法复杂度与元素值大小有关。

先将两个数组分别排序得到序列$a_1, a_2, ..., a_n$和$b_1, b_2, ..., b_m$。第k小元素不会小于$a_1 \cdot b_1$，不会大于$a_n \cdot b_m$。所以我们在$[a_1 \cdot b_1, a_n \cdot b_m]$范围内二分答案。对于这个范围内选取的一个值$val$，我们只需要判断集合中是否恰好有$k$个元素小于$val$（或者有不超过$k$个元素小于$val$，超过$k$个元素小于等于$val$），如果是，则在判断过程中我们顺便选取这$k$个数中最大的即为问题所求。算法描述如Algorithm5。

Algorithm 5: 计算第k小元素

Data: 实数数组A和实数数组B

Result: 第k小元素

1 将A排序得到升序序列$a_1, ..., a_n$;
2 将B排序得到升序序列$b_1, ..., b_m$;
3 $L \leftarrow a_1 \cdot b_1$;$R \leftarrow a_n \cdot b_m$;
4 **while** $L < R$ **do**
5     $val \leftarrow (L + R)/2$;
6     $cnt_1 \leftarrow 0$; $cnt_2 \leftarrow 0$; /* cnt_1, cnt_2将分别存储集合中小于val的元素数量和小于等于val的元素数量 */
7     $pos_2 \leftarrow m$; $ans \leftarrow -\infty$;
8   **for** $i \leftarrow 1$ *to* n **do**
9   在有序序列$a_i \cdot b_1, ..., a_i \cdot b_{pos_2}$里二分查找$val$;
10         $pos_1 \leftarrow$有序序列中小于val的元素数量;
11         $pos_2 \leftarrow$有序序列中小于等于val的元素数量;
12         $cnt_1 \leftarrow cnt_1 + pos_1$;
13         $cnt_2 \leftarrow cnt_2 + pos_2$;
14         $ans = \max(ans, a_i \cdot b_{pos_1})$;
15   **end**
16   **if** $cnt_1 = k$ **then** /* 刚好k个元素小于val，返回其中最大的那个 */
17   **return** ans
18   **else if** $cnt_1 < k, cnt_2 \geq k$ **then** /* 第k小的元素刚好等于val */
19   **return** val
20   **else if** $cnt_1 > k$ **then** /* 小于val的元素多于k个，说明val应取更小值 */
21         $R \leftarrow val$
22   **else** /* 小于等于val的元素少于k个，说明val应取更大值 */
23         $L \leftarrow val$
24   **end**
25 **end**

我们考虑算法的时间复杂度。不失一般性，不妨设$n < m$。 Algorithm5中第9行至

第14行需要在长度为$O(m)$的有序序列中进行二分，时间代价是$O(\log m)$的。第4行至第25行的二分过程需要执行的次数为$O(\log \frac{ab}{\epsilon})$，其中$a$是数组$A$中最大值，$b$是数组$B$中最大值，$\epsilon$是集合中值最接近的两个元素的差值。第1行和第2行排序的代价是$O(m \log m)$的。

算法的总时间复杂度为$O(n \log m \log \frac{ab}{\epsilon} + m \log m)$.