ProblemSet 3

9月23日

Problem 1

(a)

使用替换法

用数学归纳法证明$T(n) \geq d \cdot nlgn + d'n$

Basis: 当$n = 1$时，$T(1) = 1'$，故只要$d' \leq 1$即可成立

Inductive:

假设当$n = 2/k$时该式子也成立，即$T(k/2) \geq d \cdot \frac{k}{2} lg \frac{k}{2} + d' \cdot \frac{k}{2}$

则当$n = k$时，有$T(k) = 2T(k/2) + k \geq 2d \cdot \frac{k}{2} lg \frac{k}{2} + 2d' \cdot \frac{k}{2} + k$

$$= d \cdot klgk + (2d' + 1 - d)k$$

当$d' - d \geq -1$时上面的不等式成立

即此时$T(k) \in \Theta(klgk)$

故得证$T(n) \in \Theta(nlgn)$

(b)

当$n = i$时，结点的子问题的$size$分别是$2/n$和$n - 2$。

$\therefore$ 当$n \geq 4$时，$2/n \leq n - 2$

$\therefore T(n) = T(n-2) + T(n/2) + n \leq 2T(n-2) + n$

假设$T'(n) = 2T'(n-2) + n$

由递归树可知当$i = 0, 1, 2, \ldots, n$时，有2^i个节点，每个节点的花费是$n - 2i$

$\therefore T'(n) = \sum_{i=0}^{n} 2^i \cdot (n - 2i) = n\sum_{i=0}^{n} 2^i - 2\sum_{i=0}^{n} i \cdot 2^i$

$$= n\sum_{i=0}^{n} 2^i - (n \cdot 2^{n+2} - \sum_{i=0}^{n} 2^{i+1} + 2)$$

$$= (n+2) \cdot \sum_{i=0}^{n} 2^i - n \cdot 2^{n+2} - 2$$

$$= (n+2) \cdot (2^{n+1} - 1) - n \cdot 2^{n+2} - 2$$

$$= n \cdot 2^{n+1} - n + 2^{n+2} - 2 - n \cdot 2^{n+2} - 2$$

$$= 2^{n+2} - n \cdot 2^{n+1} - 4 - n$$

$$\leq 2^{n+2}$$

$$\leq d \cdot 2^n$$

其中$d \geq 4$

$T(n) \leq T'(n) \leq d \cdot 2^n$

$T(n) = O(2^n)$

用替换法证明:

Basis:

当$n = 1$时，$T(1) = 1 \leq 2$，成立

Inductive Hypothesis:

假设当$n = k - 2$时也成立，即$T(k) = O(2^k) \leq d \cdot 2^k - d' \cdot k$

假设当$n = k/2$时也成立，即$T(k/2) \leq d \cdot 2^{k/2} - d' \cdot \frac{k}{2}$

Induction:

则当$n = k$，$k >= 3$时，$T(k) = T(k-2) + T(k/2) + n \leq d \cdot (2^{3k/2}) - d' \cdot \frac{3k}{2}$

$T(k) = O(2^n)$当$d/d' = 3/2^4(2^{3/2} - 1)$

所以也成立

故2^n是$T(n)$的一个渐近意义上的上界

Problem2

(a)

$T(n) = T(\alpha) + T(n - \alpha) + cn$

$$= \sum_{i=0}^{n/a} c(n - ia) + (n/a)ca$$

$$= \Theta(n^2)$$

(b)

$T(n) = T(\alpha n) + T((1-\alpha)n) + cn$

$\quad = \sum_{i=0}^{log_{1/a}n} cn + \Theta(n)$

$\quad = cnlog_{1/a}n + \Theta(n)$

$\quad = \Theta(nlgn)$

Problem3

Brief Overview

将n位的数字分为前$\frac{n}{2}$位和后$\frac{n}{2}$位，将求平方的表达式拆解成完全平方式，再做三个乘法，采用分治法递归完成

Pseudocode

$FastSquare(x)$

if $x\ is\ 1-digit$

  return $x*x$

$x_l = most\ significant\ |x|/2\ digit\ of\ x$

$x_r = least\ significant\ |x|/2\ digit\ of\ x$

$z_1 = FastSquare(x_l)$

$z_2 = FastSquare(x_r)$

$z_3 = FastSquare(x_l + x_r)$

return $z_1 \cdot 2^n + z_2 + 2^{\frac{3n}{2}} \cdot ((z_3 - z_1 - z_2)/2)$

Time analysis

$T(n) = 3T(n/2) + O(n) = 3T(n/2) + an$

画出递归树，有lgn层，当$i = 0, 1, 2, \ldots,$时，每一层的值是$(\frac{3}{2})^i an$

$T(n) = \sum i = 0lgn(\frac{3}{2})^i an = C_0(\frac{3}{2}^) lgnan = aC_0 n^{lg3}$

$= O(n^{lg3})$

Problem 4

Brief Overview

因为数组的长度n是未知的，所以不能直接采用分治策略二分查找，可以采用一个类似的逆过程确定一个长度已知的子数组，然后进行二分查找。总的时间复

Pseudocode

$MultiSearch(A, x)$

1 $index = 1$

2 **do** $index = index * 2$ **until** $A[index] >= x$

3 $BinarySeach(A[index/2, \ldots, index], x)$

Time Analysis

$line3$的花费时间是$O(lg(index/2))$，而$O(lg(index/2)) \in O(lgn)$，所以$T(n) = O(lgn)$

Problem5

Brief Overview

先找出一个多数党的人，再对所有人进行扫描，找出所有多数党人

Pseudocode

$Identification(Delegates)$

Suppose$candidate\ is\ anyone, count = 0$

for $i = 1\ to\ Delegates.length$

  if $count == 0$

    $candidate = Delegates[i]$

    $count = count + 1$

  else

    if $Delegates[i]\ gives\ candidate\ a\ smile$

      $count = count + 1$

    else $count = count - 1$

$new\ array\ Majority$

$Majority.add(candidate)$

for $delegate\ in\ Delegates$

  if $candidate\ gives\ delegate\ a\ smile$

    $Majority.add(delegate)$

return $Majority$

Correctness Argument

当循环遇到一个多数党和一个少数党时，$count$值就会抵消变成0，此时$candidate$的值就会更新一次。因为多数党的人数大于$n/2$，所以对于最终的$candida$

Time Analysis

$T(n) = O(n)$

Problem 6

Brief Overview

根据提示，要通过改变返回信息简化求$cross\ maximum\ subarray$的过程，书上的方法是分别找出左部分以mid为终点和右边部分以$mid+1$为起点的最大

Pseudocode

$FindMaximumSubarray(A, low, high)$

if $high == low$

  return $(A[low], A[low], A[low], A[low])$

else $mid = (low + high)/2$

  $(llmax, lrmax, lmax, lsum) = FindMaximumSubarray(A, low, mid)$

  $(rrmax, rlmax, rmax, rsum) = FindMaximumSubarray(A, mid + 1, high)$

  $crossmax = lrmax + rlmax$

  $lmax = max(llmax, lsum + rlmax)$

  $rmax = max(rrmax, rsum + lrmax)$

  return $(lmax, rmax, max(lmax, rmax, crossmax), lsum + rsum)$

Time Complexity

$Combine$的过程变成了只要合并两个子数组，不需要再通过教材上迭代的过程，需要$\Theta(1)$的复杂度，所以时间$T(n) = 2T(2/n) + \Theta(1)$

Problem7

(a)

只要比较第二个和第三个元素的值，较大的那一个就是第二大的元素

$SecMax(A)$

if $A[2] >= A[3]$

  return $Heap[2]$

else

  return $A[3]$

(b)

Brief Overview

设有另外一个大顶堆B，把堆A的根节点放进B中，然后每次取出B的根节点，并把此元素原先在A中的两个子节点插入B中，重复$k-1$次，最后取出B的顶

Pseudocode

$Find(A, k)$

1 $new\ array\ B$

2 $B.HeapInsert(A[1])$

3 **for** $i = 1\ to\ k-1$

4   $top = B.HeapExtractMax()$

5   $B.HeapInsert(A[top.idx * 2])$

6   $B.HeapInsert(A[top.idx * 2 + 1])$

7 **return** $B.HeapGetMax$

Time Analysis

$line4, 5, 6$的操作至多需要lgk的时间复杂度，因为B的规模最大为$k+1$，而这样的过程需要重复$k-1$次，所以$T = O(klgk)$