Kunal Patel

CSE 310 Recitation 3

Objectives:

1. Exercise on applying master theorem to give tight asymptotic bounds for recurrences.
2. Apply different techniques to find the upper bound of a given recurrence.

Instruction

1. For all recitation: **the solution should be clearly typed or written and must be saved in .pdf or .jpg format**. Note: **unreadable answer receives no credits!**
2. All recitation must be submitted through the link posted on Blackboard, we do NOT accept any hand-in submissions or submissions sent through emails!

Question

1. [6 pts] Find the Big-*O* for the following recurrence relations. State clearly which technique you used to find the solution; i.e. if you get the upper bound by using a recursion tree, draw it, otherwise state which case you applied the master method with.

1) $T(n) = T(\sqrt{n}) + 1$

Induction method

$T(n) = T(n^{1/2}) + 1 = T(n^{1/2^2}) + 2 = T(n^{1/2^3}) + 3 = \ldots T(n^{1/2^k}) + K$

$n^{1/2^k} = 2 \Rightarrow 2^{2^k} = n = \lg\lg(n)$

$T(1) = 1$

Base Case: $n = 2^{2^k}$

$T(2^{2^k}) = T([2^{2^k}]^{1/2}) + 1 \Rightarrow T(2^{2^{k-1}}) + 1 = T(2^{k-2}) + 2 = \ldots T(2^0) + K = T(2) + K$

$T(2) = T(2^{1/2}) + 1 \Rightarrow T(1) + 1 = 1 + 1 = 2$

$T(2^{2^k}) = 2 + K$

k+1 case:

$T(k+1) = T([k+1]^{1/2}) + 1 \Rightarrow T(n^{1/2^k}) + K$

$\Rightarrow \boxed{O(\lg\lg n)}$

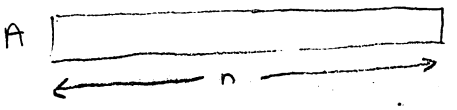
2) $T(n) = 3\,T\left(\frac{n}{2}\right) + n^2$

case 3:

$f(n) = \Theta(n^2) \rightarrow c > \log_b a \Rightarrow 2 > 1.58$

$\boxed{T(n) = \Theta(n^2)}$

2. [4 pts] Assume you have an array $A[1..n]$ of n elements. A majority element of A is any element occurring in more than $n/2$ positions (so if $n = 6$ or $n = 7$, any majority element will occur in at least 4 positions). Assume that elements cannot be ordered or sorted, but can be compared for equality. (You might think of the elements as chips, and there is a tester that can be used to determine whether or not two chips are identical.) Design an efficient divide and conquer algorithm to find a majority element in A (or determine that no majority element exists).



majority element occurs more than $n/2$

```
MAJORITY-ELEMENT(A, a, n)
1. if n=1
2.    return A[1]
3. ElementL = MAJORITY-ELEMENT(A,a,n/2)
4. ElementR = MAJORITY-ELEMENT(A, n/2+1, n)
5. if ElementL = ElementR
6.       return ElementL
7. countL = COUNT-ELEMENT(A, ElementL)
8. countR = COUNT-ELEMENT(A, ElementR)
9. if countL > n/2 + 1
10.      return elementL
11. elseif count > n/2 +1
12       return elementR
13 else
14    return NULL // no MAJORITY-ELEMENT
```

```
COUNT-ELEMENT( A, element ) // how many times element occurs
1   count = 0;
2.  for i = 0 to A.length
3      if A[i] = element
4         count++
5.  return count;
```