

After Lecture 05 & 06

– Answer any questions on HW1

Practice Problems (all taken from previous exams)

1. What are the max number of levels in the recursion tree for this recurrence relation?
 - a) $\log_4(n)$
 - b) $\log_2(n)$
 - c) $\log_{\frac{4}{3}}(n)$
 - d) $\log_{\frac{1}{3}}(n)$
2. Under what case of Master's Theorem will the recurrence relation of binary search fail?
 - a) 1
 - b) 2
 - c) 3
 - d) It cannot be solved using Master's Theorem.
3. What is the purpose of using randomized quick sort over standard quick sort?
 - a) Improve the worst-case runtime
 - b) To eliminate the possibility that a particular input order will always yield worst-case runtime
 - c) To improve accuracy of output
 - d) To improve average case time complexity
4. The non-recursive work in quicksort is done in which step of the divide-conquer-combine algorithm?
 - a)
5. Give big-O bounds of $T(n)$ in each of the following recurrences. Use induction, iteration or Master Theorem.
 - 5a)

$$T(n) = T(n-1) + n \quad T(1) = O(1)$$

$$T(n) = n + T(n-1)$$

$$T(n) = n + n-1 + T(n-2)$$

$$T(n) = n + n-1 + n-2 + T(n-3)$$

$$T(n) = n + n-1 + n-2 + \dots + T(1)$$

$$T(n) = n + n-1 + n-2 + \dots + O(1)$$

$$T(n) = \sum_{i=1}^n n$$

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

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

5b)

$$t(n) = 2T\left(\frac{n}{4} + n^{\frac{1}{2}}T(1) = O(1)\right)$$

5c)

$$T(n)$$

6. Throughout this course, we assume that parameter passing during procedure calls takes constant time, even if an N -element array is being passed. This assumption is valid in most systems because a pointer to the array is passed, not the array itself. This problem examines the implications of three parameter-passing strategies:

1. An array is passed by pointer. Time = $\theta(1)$.
2. An array is passed by copying. Time = $\theta(N)$, where N is the size of the array.
3. An array is passed by copying only the subrange that might be accessed by the called procedure. Time = $\theta(q - p + 1)$ if the subarray $A[p \dots q]$ is passed. Use $n = q - p + 1$, where n is the size of the subarray passed.

Consider the recursive binary search algorithm for finding a number in a sorted array. Give recurrences for the worst-case running times of binary search when arrays are passed using each of the three methods above, and give good upper bounds on the solutions of the recurrences. Let N be the size of the original problem and n be the size of a subproblem. Binary search works by comparing the element for which you are searching to the element at index $\frac{p+r}{2}$ of a subarray of size n , where p is the first index of the subarray and r is the last index (integer division is used). Therefore, the array passed in binary search is continually divided in half.

1)

$$T(n) = T\left(\frac{n}{2}\right) + O(1) \text{ with } T(1) = O(1)$$

The array is passed by pointer, which is constant time. Therefore, the time involved in the

2)

7. Use the definition of Θ and induction to prove that the recurrence $T(n) = T(N - 1) + \theta(n)$ (worst case Quicksort) has the solution $T(n) = \Theta(n^2)$. Since we are not given any boundary conditions, we can assume the basis step for the inductive proof. Assume the claim is true for $n = k$.

$T(k) = \theta(k^2)$, in other words assume $c_1 k^2 \leq T(k) \leq c_2 k^2$ for some $c_1 > 0$, $c_2 > 0$ and k large enough.

Use that to prove the claim

8. What if you are sorting a collection of data that can have multiple entries of the same value. When calling Quicksort's Partition(A , p , r), where do elements equal to the pivot end up and why? How could we modify Quicksort and Partition (write pseudocode) so that if we happen to partition on a pivot that had many duplicate values, we can improve the runtime of Quicksort by having smaller recursive calls?

Algorithm 1 Quicksort1

```
1: function QUICKSORT1( $A, p, r$ ) ▷  
2:   if  $p < r$  then  
3:      $(q_1, q_2) = \text{PARTITION1}(A, p, r)$  ▷ \text{two return values}  
4:      $((A, p, q_1 - 1)$   
5:   end if  
6: end function
```

Algorithm 2 Partition1

```
1: function PARTITION1( $A, p, r$ ) ▷ \text{Two return values}  
2:    $\text{endLow} \leftarrow p - 1, \text{endEqual} \leftarrow p - 1$   
3:    $\text{pivot} \leftarrow A[r]$   
4:   for  $j=p$  to  $r-1$  do  
5:      $\text{get function}$   
6:   end for  
7: end function
```
