CSE 5311 Section 003 Fall 2004

Quiz 1 Solution

September 8, 2004

This is a closed-book, closed-notes, closed-computer, closed-neighbor quiz.

1. Below is pseudocode for the iterative algorithm SortedInsert, which inserts the key k into an already sorted array A and maintains the sorted order of the array.

SortedInsert(A,k)	Cost	Iterations
i = length(A)	1	1
while $i > 0$ and $k < A[i]$	1	$t + 1$
$\quad A[i+1] = A[i]$	1	t
$\quad i = i - 1$	1	t
$A[i+1] = k$	1	1

   (a) (4 points) Perform a line-by-line analysis of SortedInsert and derive a precise (non-asymptotic) expression of the running time $T(n)$, where n = length(A). You may assume a cost $c_i = 1$ for each line of pseudocode.

   See line-by-line analysis above, where t is the number of elements of the array A greater than key k.

$$T(n) = 3t + 3$$

   (b) (3 points) Describe the best-case scenario for SortedInsert and give both a precise and asymptotically-tight bound on the best-case running time.

   The best-case scenario is when the key k is larger than any element in A, and the body of the while loop does not execute (i.e., $t = 0$). Thus, $T(n) = 3 = \Theta(1)$.

   (c) (3 points) Describe the worst-case scenario for SortedInsert and give both a precise and asymptotically-tight bound on the worst-case running time.

   The worst-case scenario is when the key k is smaller than any element in A, and the body of the while loop executes n times (i.e., $t = n$). Thus, $T(n) = 3n + 3 = \Theta(n)$.

2. Consider the following alternative approach to SortedInsert that first uses binary search to find the position for key k and then shifts over the elements to the right and inserts k.

```
SortedInsert(A,k)
n = length(A)
i = Search(A,k,1,n)
for j = n to i + 1
    A[j + 1] = A[j]
A[i + 1] = k
```

```
SEARCH(A,k,p,r)
    if p < r
        q = ⌊(p+r)/2⌋
        if k ≤ A[q]
            return SEARCH(A,k,p,q)
        else return SEARCH(A,k,q+1,r)
    else return p
```

(a) (4 points) Give a recurrence describing the worst-case running time $T(n)$ of SEARCH, where $n = r - p + 1$.

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ T(n/2) + \Theta(1) & n > 1 \end{cases}$$

(b) (3 points) Give an asymptotically-tight bound for the worst-case running time of this version of SORTEDINSERT. You do not have to solve the recurrence.

SEARCH will always take $\Theta(\lg n)$ time. In the worst case, SEARCH will return the lowest value for i, causing the **for** loop to shift the entire array to the right, which takes $\Theta(n)$. Thus, the total running time of SORTEDINSERT will be $T(n) = \Theta(\lg n) + \Theta(n) + \Theta(1) = \Theta(n)$.

3. (4 points) Solve the following recurrence using the master method. Show your work.

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ T(n/2) + \Theta(1) & n > 1 \end{cases}$$

Master method: $a = 1$, $b = 2$, $f(n) = \Theta(1)$. Thus, $n^{\log_b a} = n^{\log_2 1} = n^0 = \Theta(1)$, which equals $f(n)$. Therefore, we are in case 2 of the master theorem, and $T(n) = \Theta(\lg n * \Theta(1)) = \Theta(\lg n)$.

4. (4 points) Use the substitution method to show that $T(n) = O(n)$ for the recurrence in Problem 3.

Show that $T(n) = O(n) \leq cn$.

Assume $T(n/2) \leq cn/2$.

$$\begin{aligned} T(n) &\leq cn/2 + \Theta(1) \\ &\leq cn/2 + \Theta(1) + (cn/2 - \Theta(1)) \\ &\leq cn \end{aligned}$$

given that $cn/2 - \Theta(1) \geq 0$, or $c \geq 2\Theta(1)/n$. Since $\Theta(1)$ represents a constant, for large enough n, c can be a valid constant and still satisfy the inequality. Thus, $T(n) = O(n)$.