

Analysis of Algorithms
CS 3343 Lecture Five

Prof. William Winsborough
September 18, 2008

Business

- Turn in Homeworks 1 and 2
- Read section 4.3
- Homework 3 Due Thursday September 25:
 - Exercise: 4.2-1
- If after today's lecture you want to redo your proof of correctness part of 2.1-3 and/or 2.2-2, please turn them in on Tuesday 9/23

18 September 2008

Winsborough CS 3343 Lecture 5

2

Loop Invariants Revisited: Problem 2.1-3

linearSearch(val, A[1..n])

Loc := 0

For i := 1 to n do

 if (A[i] = val) Loc := i;

return Loc

- Invariant: at the start of iteration i,
 $((\forall j. 1 \leq j < i \rightarrow A[j] \neq \text{val}) \wedge \text{Loc}=0) \vee A[\text{Loc}] = \text{val}$

18 September 2008

Winsborough CS 3343 Lecture 5

3

Loop Invariants Revisited: Problem 2.1-3, Alternate Solution

altLinearSearch(val, A[1..n])

For i := 1 to n do

 if (A[i] = val) return i

return 0

- Invariant: at the start of iteration i,
 $(\forall j. 1 \leq j < i \rightarrow A[j] \neq \text{val})$
- Termination:
 - Here we show that the correctness of the algorithm follows from the loop invariant
 - There are two cases, one for each way the loop can exit:
 - If the loop exits from the middle (return i), then clearly $A[i] = \text{val}$
 - If the loop exits due to the value of i exceeding n, then $i = n+1$, and the loop invariant tells us that val is not in $A[1..n]$.

18 September 2008

Winsborough CS 3343 Lecture 5

4

Loop Invariants Revisited: Problem 2.2-2

selectionSort(A[1..n])

for i := 1 to n-1 do

 smallest := i

 for j := i+1 to n do

 if $A[j] < A[\text{smallest}]$ smallest := j

 swap(A[i], A[smallest])

- Outer invariant:
 $\forall k. 1 \leq k < i \rightarrow A[k]$ is the k'th smallest element in $A[1..n]$
- Inner invariant:
 $\forall p. i \leq p < j \rightarrow A[\text{smallest}] \leq A[p]$

18 September 2008

Winsborough CS 3343 Lecture 5

5

The Recursion-tree Method

- Nodes represent the cost of a single subproblem among the recursive function invocations
- Sum up the costs within each level of the tree, then sum over the levels
- Basic method:
 - Allow a little sloppiness
 - Use the approach to come up with a good guess at a solution
 - Verify the solution is correct by using the substitution method

18 September 2008

Winsborough CS 3343 Lecture 5

6

Example

- $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$
 - Ignore floor and assume $n = 4^k$ for some k
- Create tree for $T(n) = 3T(n/4) + cn^2$
- The size of the subproblem at level i is $n/4^i$
 - Reach the base case when $n/4^i = 1$,
or $i = \log_4 n$
 - So height of tree is $\log_4 n + 1$
- Number of nodes at depth i is 3^i and each node has cost $c(n/4^i)^2$
 - Total cost of level i is $3^i c(n/4^i)^2 = (3/16)^i cn^2$
- The bottom level has $3^{\log_4 n} = n^{\log_4 3}$ nodes, each contributing cost $\Theta(1)$, for a total cost of $\Theta(n^{\log_4 3})$

18 September 2008

Winsborough CS 3343 Lecture 5

7

Summing the Levels

- Copy from blackboard or refer to p68-70 in text

18 September 2008

Winsborough CS 3343 Lecture 5

8