COMP9020 18s1

Week 9 Problem Set
Running Time of Programs

Foundations of Computer Science

[Show with no answers] [Show with all answers]

1. (Asymptotic running times)

   Suppose you have the choice between three algorithms:

   a. Algorithm A solves your problem by dividing it into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.

   b. Algorithm B solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.

   c. Algorithm C solves problems of size n by dividing them into nine subproblems of size $\frac{n}{3}$, recursively solving each subproblem, and then combining the solutions in $\mathcal{O}(n^2)$ time.

   Estimate the running times of each of these algorithms. Which one would you choose?

   [show answer]

2. (Recurrences for algorithm analysis)

   Recall the recurrence for Mergesort: $T(1) = 0$; $T(n) = 2T(\frac{n}{2}) + (n-1)$ for $n > 1$.

   Prove by induction that $T(n) = n \cdot (\log_2 n - 1) + 1$ for all $n = 2^k$ (with $k \geq 1$).

   [show answer]

3. (Recursive algorithms)

   a. Analyse the complexity of the following recursive algorithm to test whether a number x occurs in an *unordered* list $L = [x_1, x_2, \ldots, x_n]$ of size n. Take the cost to be the number of list element comparison operations.

      Search$(x, L = [x_1, x_2, \ldots, x_n])$:

      if $x_1 = x$ **then return** yes

      else if $n > 1$ **then return** *Search*$(x, [x_2, \ldots, x_n])$

      else return no

   b. Analyse the complexity of the following recursive algorithm to test whether a number x occurs in an *ordered* list $L = [x_1, x_2, \ldots, x_n]$ of size n. Take the cost to be the number of list element comparison operations.

      BinarySearch$(x, L = [x_1, x_2, \ldots, x_n])$:

      if $n = 0$ **then return** no

      else if $x_{\lceil \frac{n}{2} \rceil} > x$ **then return** *BinarySearch*$(x, [x_1, \ldots, x_{\lceil \frac{n}{2} \rceil - 1}])$

      else if $x_{\lceil \frac{n}{2} \rceil} < x$ **then return** *BinarySearch*$(x, [x_{\lceil \frac{n}{2} \rceil + 1}, \ldots, x_n])$

      else return yes

   [show answer]

4. **Challenge Exercise**

Without using the Master Theorem, give tight big-Oh upper bounds for the divide-and-conquer recurrence $T(1) = 1$; $T(n) = T(\frac{n}{2}) + g(n)$, for $n > 1$, where

a. $g(n) = 1$

b. $g(n) = 2n$

c. $g(n) = n^2$

[show answer]