CS 3343 Analysis of Algorithms – Fall 10

9/14/10

3. Homework

Due **9/21/10** before class

1. **Induction (3 points)**
   Let $T(0) = 1$ and $T(n) = 2\,T(n-1) + 1$ be a runtime recurrence. Prove using induction on n that $T(n) = 2^{n+1} - 1$.

2. **Recursive mystery (6 points)**

```
int mystery(int[] A, int i, int j){
  if(i>=j)
    return A[i];

  int a = mystery(A, i+1,j);
  int b = mystery(A, i, j-1);
  return a*b;
}
```

   (a) (2 points) Set up a runtime recurrence for the `mystery` method above. Do not forget the base case. Assume the initial call is `mystery(A, 1, n)` where `n` is the length of array `A`. Shortly argue why your recurrence correctly specifies the runtime.

   (b) (1 point) Is this `mystery` method a divide-and-conquer algorithm? Justify your answer shortly.

   (c) (3 points) Using the recursion tree method, come up with a guess what this runtime recurrence will solve to. Your guess should be expressed as a simple runtime function in Θ-notation (such as $\Theta(n \log n), \Theta(n^2), ...$)?

3. **Divide and Conquer (8 points)**
   Let $A[1..n]$ be a sorted array of n distinct numbers that represent n points on a line. The task is to find the distance between the two closest points. For example, if $A = [1, 6, 10, 13, 19, 20, 23]$ then $6 - 1 = 5$, $10 - 6 = 4$, $13 - 10 = 3$, $19 - 13 = 6$, $20 - 19 = 1$, $23 - 20 = 3$, and hence 1 is the smallest distance which is attained between 19 and 20.

   While it is pretty straight-forward to develop an $O(n)$-time algorithm for this task, in this exercise you should develop on $O(n)$-time **divide-and-conquer** algorithm.

   (a) (3 points) Give an $O(n)$-time **divide-and-conquer** algorithm to solve this task.

   (b) (2 points) Derive the runtime recurrence for your algorithm (do not forget the base case.) Shortly argue why your recurrence correctly specifies the runtime.

   (c) (3 points) Use the recursion tree method to argue why your runtime recurrence solves to $O(n)$; you do not need to do a big-Oh induction.

FLIP OVER TO BACK PAGE $\Longrightarrow$

4. **Guessing and Induction (6 points)**

   For the following recurrence use the recursion tree method to find a good guess of what it could solve to. Make your guess as tight as possible. Then prove that $T(n)$ is in big-Oh of your guess by big-Oh-induction (= substitution method; including base case and inductive case).

   $T(1) = 1$

   $T(n) = 4T(\frac{n}{3}) + n^2$ for $n \geq 2$.