CS 3343 Analysis of Algorithms – Fall 11

9/13/11

3. Homework

Due **9/20/11** before class

1. **Recursive mystery (4 points)**

```
 int mystery(int n, int a){
   if(n==0)
     return 0;

   int tmp1 = mystery(n/2,a); // n/2 rounds down
   int tmp2 = 0;
   for(int i=1; i<=n/2; i++)
     tmp2 += a;

   if(n%2 = 1) return tmp1+tmp2+a;
   else return tmp1+tmp2;
}
```

   (a) (1 point) What does the `mystery` method above compute?

   (b) (2 points) Set up a runtime recurrence for the `mystery` method above. Do not forget the base case.

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

2. **Recursion tree (8 points)**
   For the following recurrences use the recursion tree method to find a good guess of what they could solve to asymptotically (i.e., in big-Oh terms). Assume $T(1) = 1$. You may need to use that $a^{(b^c)} = a^{b \cdot c} = a^{(c^b)}$.

   (a) $T(n) = 3T(\frac{n}{3}) + n$ for $n \geq 2$

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

3. **Big-Oh Induction (4 points)**
   Use big-Oh-induction (= substitution method; including base case and inductive case) to prove that $T(n) \in O(n^2)$ where $T(1) = 1$ and $T(n) = 3T(n/2) + n^2$ for $n \geq 2$.

Flip over to back page ⟹

Practice Problems

(Not required for homework credit.)

1. **Induction**
   Prove by weak induction on n that the following equality holds for constant $a \neq 1$ and all $n \geq 0$:
   $$\sum_{i=0}^{n} a^i = \frac{a^{n+1} - 1}{a - 1}$$

2. **3-way mergesort**

```
int 3wayMergesort(int i, int j, int[] A){
  // Sort A[i..j]
  if(j-i<=1)
    return;

  l = (j-i)/3;
  3wayMergesort(i,i+l, A);
  3wayMergesort(i+l+1,i+2*l,A);
  3wayMergesort(i+2*l+1,j,A);
  merge(i,i+l+1,i+2*l+1); // Merges all three sub-arrays in linear time
}
```

   The first call is `3wayMergesort(1,n,A)` to sort the array $A[1..n]$.

   Set up a runtime recurrence ($T(n) = ...$) for `3-way mergesort` above. Do not forget the base case.

3. **Recursion tree**
   For the recurrence
   $$T(n) = 3T(\frac{n}{2}) + n^2 \quad \text{for} \quad n \geq 2$$
   use the recursion tree method to find a good guess of what it could solve to asymptotically (i.e., in big-Oh terms). Assume $T(1) = 1$. You may need to use that $a^{(b^c)} = a^{b \cdot c} = a^{(c^b)}$.

4. **Big-Oh Induction**
   Use big-Oh-induction (= substitution method; including base case and inductive case) to prove that $T(n) \in O(n \log n)$ where $T(1) = 1$ and $T(n) = 3T(n/3) + n$ for $n \geq 2$.