CMPS 2200 Introduction to Algorithms – Fall 12

9/17/12

# 2. Homework

Due **9/25/12** in the lab

1. **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 $n \geq 2$

2. **Induction (4 points)**
   Let $T(1) = 2$ and $T(n) = 4T(n/2) + n^3$ for $n \geq 2$. Use induction to prove that $T(n) = 2n^3$ for all $n \geq 1$.

3. **Strassen's Algorithm (4 points)**
   Apply Strassen's algorithm to compute

$$\begin{pmatrix} 1 & 0 & 2 & 1 \\ 4 & 1 & 1 & 0 \\ 0 & 1 & 3 & 0 \\ 5 & 0 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 4 \\ 2 & 0 & 1 & 1 \\ 1 & 3 & 5 & 0 \end{pmatrix}$$

   The recursion should exit with the base case $n = 1$, i.e., $2 \times 2$ matrices should recursively be computed using Strassen's algorithm. In order to save you some work, you may assume that the following is a partial solution and you only have to fill in the missing values by using Strassen's algorithm:

$$\begin{pmatrix} 5 & 4 & & \\ 4 & 5 & & \\ 8 & 1 & 3 & 7 \\ 5 & 8 & 7 & 7 \end{pmatrix}$$

4. **Divide and Conquer (6 points)**
   Let $A[1..n]$ be an array of $n$ numbers. A number in $A$ is a *majority element* if $A$ contains this number at least $\lfloor n/2 \rfloor + 1$ times.

   Write a divide-and-conquer algorithm that determines whether a given array $A[1..n]$ contains a majority element, and if so, returns it. Your algorithm should run in $O(n \log n)$ time. You are **not** allowed to sort the array.

   Set up and solve a recurrence relation for the runtime of your algorithm.

   *Hint: Start by applying the generic divide-and-conquer approach. Try to divide by two. Then try to combine the results. From the given runtime you should be able to guess how much time you are allowed to spend for dividing and combining.*

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. **Binary Integer Multiplication**
   Let $x = 10110110_2$ and $y = 11001010_2$. Compute $x \cdot y$ using the fast recursive multiplication algorithm.

5. **Compute Minimum**
   Let $A$ be an unsorted array of $n$ numbers. Develop a divide-and-conquer algorithm to compute the minimum in $A$.