CMPS 2200 Introduction to Algorithms – Fall 14

9/23/14

4. Homework

Due **10/2/14** at the beginning of class

Remember, you are allowed to turn in homeworks in groups of two. One writeup, with two names.

1. **3-way mergesort (5 points)**

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

  int 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(0,n-1,A)` to sort the array $A[0..n-1]$.

Set up a runtime recurrence ($T(n) = ...$) for `3-way mergesort` above. Then solve the recurrence using the method of your choice. What is the runtime of `3wayMergesort`?

2. **Master theorem (10 points)**
Use the master theorem to solve the following recurrences. Justify your results. Assume that $T(1) = 1$.

(a) $T(n) = 3T(\frac{n}{3}) + 3$
(b) $T(n) = 64T(\frac{n}{2}) + n^6 \log^2 n$
(c) $T(n) = 27T(\frac{n}{3}) + n \log n$
(d) $T(n) = 8T(\frac{n}{2}) + n^4$
(e) $T(n) = 64T(\frac{n}{4}) + n^3$

Flip over to back page $\Longrightarrow$

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

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

The recursion should exit with the base case $n = 1$, i.e., 2×2 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:

$$\begin{pmatrix} 2 & 2 & & \\ 5 & 6 & & \\ 3 & 2 & 5 & 4 \\ 5 & 0 & 3 & 3 \end{pmatrix}$$

4. **More Strassen's (4 points)**
Suppose you want to develop an algorithm to multiply two $n \times n$ matrices in time faster than Strassen's algorithm. Suppose your algorithm proceeds in dividing the problem up into parts of size $\frac{n}{4} \times \frac{n}{4}$, and that your divide and combine steps together take $\Theta(n^2)$ time. You would like to find out how many subproblems you need in order to be faster than Strassen's algorithm. If you have a subproblems the recurrence is $T(n) = aT(\frac{n}{4}) + \Theta(n^2)$. Find the largest (integer) value of a for which your algorithm would be asymptotically faster than Strassen's.