

CS251 Data Structures – Fall 2009

Homework 3 – 60 points

Due: Oct. 29

1. (10 points) Assume we have four algorithms with time complexities $\Theta(n)$, $\Theta(n^2)$, $\Theta(n^3)$ and $\Theta(2^n)$. Assume that each solves a problem of size 10 in 2 sec.
 - (a) For each algorithm, determine the time to solve a problem of size 20.
 - (b) For each algorithm, determine the time to solve a problem of size 100.
 - (c) For each algorithm, determine the maximum size of a problem which can be solved in one minute.
 - (d) For each algorithm, determine the maximum size of a problem which can be solved in one hour.
2. (10 points)
 - (a) Show that $O(a^{n+b}) = O(a^n)$ for any constant $a > 1$ and b .
 - (b) Show that $O(a^{2^n}) \neq O(a^n)$ for any constant $a > 1$.
3. (5 points) Problem 5.14, pg. 194 (Weiss)
4. (20 points) Determine both the exact and the Big-Oh time complexity of the following code fragments. Be sure to show how you set up your summations.

```
for(int i=1; i<=n; i++)      for(int i=1; i<=n; i++)
  for(int j=1; j<=i; j++)    for(int j=i; j<=2*i; j++)
    sum++;                  sum++;
```

```
for(int i=1; i<=n; i++)      for(int i=1; i<=n; i++)
  for(int j=1; j<=2*i; j++)  for(int j=i; j<=n; j++)
    for(int k=1; k<=j*j; k++) for(int k=i; k<=j; k++)
      sum++;                  sum++;
```

5. (5 points) Determine the Big-Oh class (in terms of n) of the following code:

```
sum = 0;
for (i=1; i<=n; i++) {
  if (i is a power of 2) {
    for(j=1; j<=i; j++) {
      sum += j;
    }
  }
}
```

You may assume that n is a power of 2, i.e. $n = 2^m$. NOTE: the answer to this problem is NOT $O(n^2)$.

(over)

6. (5 points) In class we developed a formula for the number of rectangles that can be found in an n by m grid of squares. Now assume that we take out one of the corner squares of the board. Determine a closed formula in terms of n and m for the number of rectangles in this modified board.
7. (5 points)
- (a) Show that the time complexity of Mergesort remains the same regardless of how many equal size sections we divide the array into at each step, as long as the merge time remains $O(n)$.
 - (b) What would the time complexity of Mergesort be if merging took $O(n^2)$ time?