Homework 1: Analysis

Due Friday, April 8, at 11:59 PM

Asymptotic Notation

Specify whether $f = O(g)$, or $f = \Omega(g)$ or both which would be $f = \Theta(g)$.

	f(n)	g(n)
(a)	n - 100	n - 200
(b)	$n^{\frac{1}{2}}$	$n^{\frac{2}{3}}$
(c)	100n +log(n)	n + $(log(n))^2$
(d)	nlog(n)	10n log(n)
(e)	log(2n)	log(3n)
(f)	10 log(n)	log(3n)
(g)	$n^{1.01}$	n log^2(n)
(h)	$\frac{n^2}{log(n)}$	$n(log(n))^2$
(i)	$n^{0.1}$	$log(n)^{10}$
(j)	$log(n)^{log(n)}$	$\frac{n}{log(n)}$
(k)	$\sqrt{n}$	$log(n)^3$
(l)	$n^{\frac{1}{2}}$	$5^{log_2(n)}$
(m)	$n2^n$	3^m

Iterative Substitution

Use a recursion tree to determine a good asymptotic upper bound on the recurrence. Use the substitution method to verify your answer.

1. T(n) = T(n/2) + n^2
2. T(n) = 4T(n/2) + n

Master's Theorem

When appropriate, use Master's theorem to solve the recurrence relations. If Master's theorem does not apply specify why.

1. $T(n) = 2T(n/2) + n^4$

2. $T(n) = T(7n/10) + n$

3. $T(n) = 16T(n/2) + n^2$

4. $T(n) = 7T(n/3) + n^2$

5. $T(n) = 7T(n/2) + n2$

6. $T(n) = 2T(n/4) + \sqrt{n}$

7. $T(n) = 7T(n-2) + n2$

Building Recurrence relation

For the following question you may assume that the function merge takes O(N) time.

1. Build the recurrence relation for the code shown below for BadSort.
2. Solve the recurrence relation
3. Use induction to prove the recurrence relation is correct.

```
def BadSort(A):
    if len(A) < 2:
        return A
    else:
        third = len(A)/3
        l = BadSort(A[:third])
        c = BadSort(A[third:2*third])
        r = BadSort(A[2*third:])
        return merge(l,c,r)
```