The test itself will not be as long.

- 1.) Induction
- 2.) Asymptotic Analysis
- 3.) Run-time Code analysis.
- 4.) Divide and Conquer
 - a.) Substitution
 - b.) Recurrence tree
 - c.) Masters

Proof by induction

$$\sum_{i=1}^{n+1} i \cdot 2^i = n2^{n+2} + 2 \qquad \text{For all integer n >= 0}$$

Solution:

Base cases: n=0, L.S.= 2 =R.S.
Assume f(k) is true, i.e
$$\sum_{i=1}^{k-1} i2^i = k2^{k+2} + 2$$

When n = k+1
$$\sum_{i=1}^{k+2} i2^i$$

$$= (k+2)2^{k+2} + \sum_{i=1}^{k+1} i2^i$$

$$= (k+2)2^{k+2} + k2^{k+2} + 2$$

$$= k2^{k+2} + 2^{k+3} + k2^{k+2} + 2$$

$$= k2^{k+3} + 2^{k+3} + 2$$

$$= (k+1)2^{k+3} + 2$$

Asymptotic Analysis

2.) 5pts

By definition or limit lemma

Find c and M to prove that $2n^5 + 3n^3 + 6 = O(n^5)$.

Solution:

Note
$$2m^5+3m^3+6m \le 2m^5+3m^5+6m^5$$
 for $m > 1$

$$= 11m^5$$
So piding $c=11$, $M=1$ slows
$$2m^5+3m^3+6 = O(n^5)$$

- 3.) Divide & Conquer solving recurrences and code analysis.
 - Code analysis is also an option for this question like on the quizes..

T(n)=T(n/3)+n

a.) By recursion tree method (4 pts)

Provide the tightest bound for T(n) = T(n/3) + n

T(n(3)
$$N_3$$

T(n(3)) N_3

T(n(3)) $N/9$

I see the pattern that for level interes is

 $\frac{n}{3^7}$ amount of sperations

the depth is:

 $\frac{h}{3^7} = 1$
 $\frac{h}{3^7} = 1$

b.) By substitution (3 pts)

Prove the above with substitution from the solution you found in a

c.) By Master theorem (2pts)

$$T(n) = T(n/3) + n$$
 $T(n) = Cn$
 $T(n) \le Cn$
 $T(n) \le Cn$
 $C(n) + n \le cn$
 $C(n)$

$$T(n) = T(n/3) + n$$

$$f(n) = \Theta(n')$$

$$D = 1 \quad b = 3$$

$$log_b a = 4000 log_3 1? 1$$

$$CASE 3$$

$$\Theta(n)$$

4.) Divide and Conquer analysis

- . Suppose you are choosing between the following three algorithms:
 - Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
 - Algorithm B solves problems of size n by recursively solving two subproblems of size n-1 and then combining the solutions in constant time.
 - Algorithm C solves problems of size n by dividing them into nine subproblems of size n/3, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

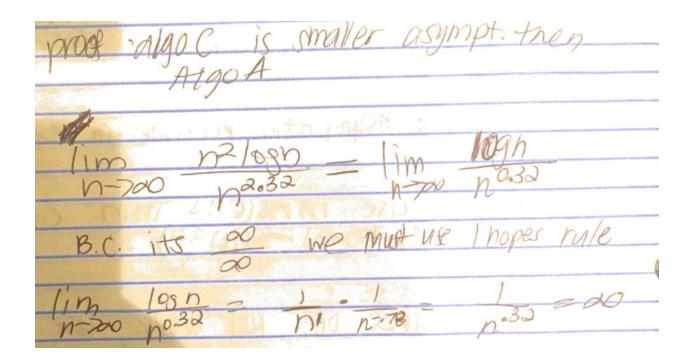
What are the running times of each of these algorithms (in big- $\!\mathit{O}$ notation), and which would you choose?

ANSWER:	Best alsopithm	(faster) asymp. C) Algo A, log, Algo B
Mexade	orst case. Algo.) Algo +, any Algo S
NORK:		$=$ $\Omega(n^232)$
Algo A	5T(n/2) + O(n)	= 0(N230)
HAO B	$9T(n/3) + O(n^2)$	O(n2/09n)

How did I come up with the answer:

Algo A: USE MASTERS THM CASE I
AGO B: MASTERS DOES NOT WORK, USE THE Methodolphee work AT vevel Work at level?
T(n-1) $T(n-1)$ a depth $1 - L = 1$
$\frac{depth}{2} = \frac{1}{2^{n-1}} = \frac{1}{2^{n-1}} = \frac{1}{2^{n-1}}$
Algo C: MASTURI THEOREM CASE 2: O(n2logn)

How do I know that $n^{\log_2(5)} > n^2 \log n$



b.)
Examining algorithms from HW2 problem 8 and 9