

Faculty of Science

**Course**: CSCI 3070U: Design and Analysis of Algorithms

**Tutorial:** #3

**Topic:** Solving recurrences

1. Use the master method to determine a tight bound for the following recurrences:
   1. T(n) = T(n/2) + n

logba = log21 = 0

Case 1: Is f(n) = n = O(n0-e)? No

Case 2: Is f(n) = n = Θ(n0)? No

Case 3: Is f(n) = n = Ω(n0+e)? Yes

Is af(n/b) = f(n/2) = n/2 ≤ cf(n) = cn? Yes (for c = ¾)

Since Case 3 applies, T(n) = Θ(f(n)) = Θ(n)

* 1. T(n) = 27T(n/3) + lg n

logba = log327 = 3

Case 1: Is f(n) = lg n = O(n3-e)? Yes (for e = 1, 2)

Since Case 1 applies, T(n) = Θ(nlogba) = Θ(n3)

* 1. T(n) = 9T(n/3) + n2

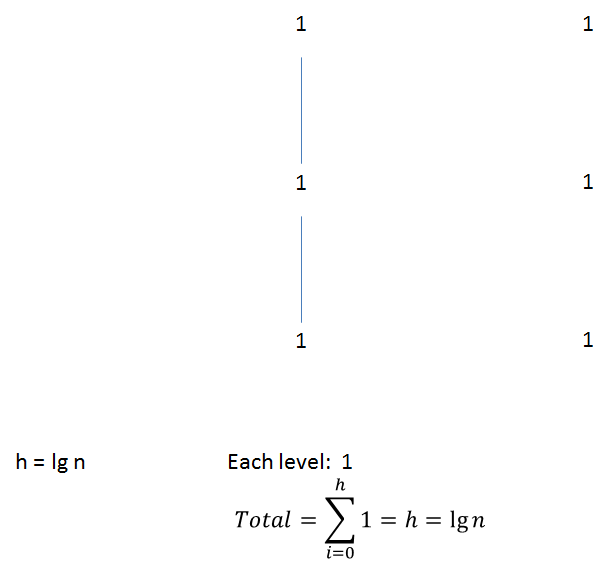
logba = log39 = 2

Case 1: Is f(n) = n2 = O(n2-e)? No

Case 2: Is f(n) = n2 = Θ(n2)? Yes

Since Case 2 applies, T(n) = Θ(nlogba lg n) = Θ(n2lg n)

1. Write the recursion tree for the binary search algorithm.



1. Use substitution to prove that T(n) = 4T(n/2) – 1 is O(n2) (Note: Big-O, not Big-Theta)

T(n) ≤ 4T(n/2)2-1

= 4c(n/2)2-1

= cn2 – 1

≤ cn2