**CS430 Lecture 03 Activities**

Asymptotic Analysis (more details)

BIG-O Notation – Upper bound on growth of a runtime function

f(n) ∈ O(g(n)) “f(n) is big-O of g(n)”

If there exists C, no such that

0 < f(n) < Cg(n) when n> no

1a. Use the definition of big-O to show 2n^2 is big-O n^3 (find a C and no that works in the above)

1b. Use the definition of big-O to show T(n)=3n^3-4n^2+3lgn-n = O(n^3)



2. Use the definition of omega to show n^(1/2) = Omega(log n)



3a. Use the definition of theta to show 3n^3-4n^2+37n = Theta(n^3)

3b. Use the definition of theta to show n^2 +3n^3 = Theta(n^3)

Recursive Sorting – Mergesort

* divide and conquer (and combine) approach, recursive algorithm
* key idea: you can merge two sorted lists of total length n in THETA(n) linear time
* base case: a list of length one element is sorted

1. Demonstrate how you can merge two sorted sub-lists total n items with n compares/copies. How much memory do we need to do this? Write pseudocode to do this.

2 3 7 8 1 4 5 6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |

Mergesort(A, p, r) { // initial call Mergesort (A, 1, n)

if (p<r) {

q = (p+r)/2 // integer division

Mergesort(A, p, q) // recursively sort 1st half

Mergesort(A, q+1, r) // recursively sort 2nd half

Merge(A, p, q, r) // merge 2 sorted sub-lists

}

}

2. Demonstrate Mergesort on this data

3 41 52 26 38 57 09 49