Algorithm – I CS21003
Tutorial 1

July 28th, 2017

1. Put the following functions in order from lowest to highest in terms of their Θ classes. (Some of the functions may be in the same Θ class. Indicate that on your list also.)
(a) $f_1(n) = n \log n$,
(b) $f_2(n) = n^{3/2}$,
(c) $f_3(n) = 1000$
(d) $f_4(n) = \sqrt{n(n + \log n)}$,
(e) $f_5(n) = 3^n$,
(f) $f_6(n) = 2^{n+2}$,
(g) $f_7(n) = 0.0001$.

2. Prove or disprove the following:
(a) $f(n) + g(n) = O(f(n)g(n))$
(b) $\log((2n)!) = \Theta(\log(n!))$

3.a. Consider the modified binary search algorithm so that it splits the input not into two sets of almost-equal sizes, but into three sets of sizes approximately one-third. Write down the recurrence for this ternary search algorithm and find the asymptotic complexity of this algorithm.

b. Consider another variation of the binary search algorithm so that it splits the input not only into two sets of almost equal sizes, but into two sets of sizes approximately one-third and two-thirds. Write down the recurrence for this search algorithm and find the asymptotic complexity of this algorithm.

c. Similarly as above, what is the recurrence relation for k-ary search while splitting like in a and b and find out asymptotic time complexity

4. Assume you have an array A[1..n] of n elements. A majority element of A is any element occurring in more than n/2 positions. Assume that elements cannot be ordered or sorted, but can be compared for equality. Design an efficient divide and conquer algorithm to find a majority element in A (or determine that no majority element exists).

5. Solve the following recurrence relations by Master Theorem
a) $T(n) = 81\,T(n/9) + n^4 \log n$
b) $T(n) = 4T(n/3) + n^{\log_3 4}$

6. You are given k sorted lists each of size n. Describe a divide-and-conquer algorithm to merge the k lists into a single sorted list of size kn. What is the running time of your algorithm? What about a naïve algorithm?

7. Let A be a sorted array of n distinct elements. An array B is formed by cyclically right-shifting the array A by some k cells. Given B (but not A or k), determine how the shift amount k can be computed in O(log n) time.