Back to Home

Midterm I Review for CS3343

Fall 2011

1. **Analyzing Algorithms:** Best case, worst case runtime
   - **Example:** best case and worst case runtime for a given code snippet(Homework 2 Problem 2)
2. **Asymptotic Notation (O, Ω, θ, o)**
   - Prove by Definition (you have to **find out a value of c>0 and n_0 > 0** so that for all n≥n_0 the inequality satisfies)
     - f(n) belongs to O(g(n))
     - f(n) belongs to Ω(g(n))
     - f(n) belongs to θ(g(n)) [you need to prove both O and Ω]
   - Prove using **limit theorm**
     - Apply limit theorm on the functions and decide from the value
     - Ranking of functions(Example: Homework 2 problem 1)
3. **Recursion:** For a given psedocode you have to come up with a recurrence
   - **Divide & Conquer**
     - You can call an algorithm **Divide and Conquer** only if the size of subproblems can be written as n/b where b>1
   - **Regular Recursion**
     - subproblems can be of size n-1, n-2, n-3 etc.
4. **Recursion Tree:** Come up with a guess using Recursion Tree method
   - Given T(n) = aT(n/b) + f(n)
   - a = #of subproblems = #of children at each node
   - n/b = subproblem size
   - height of the tree = log_b (n)[log of n base b]
5. **Solving Recurrence:** Two ways has been discussed so far for solving a recurrence
   - **big-Oh induction**
     - you will have a **guess/claim**
     - state your **inductive hypothesis** for all k<n
     - replace k with a value[**the size of subproblem in the right hand side of recurrence**] < n
     - plugin the value you got in previous step into your recurrence
     - find out your **desired** and **residual** part
     - **residual part must be<0**, so you will end up with a condition most likely on **c**
     - Verify if it works for your **base case**
     - select a value of **c**(which satisfies the base case and the condition you got from residual<0) and **n_0**(base case value of n)
   - **Master Method**
     - find the value of a and b
     - compute n^(log of a base b) and compare with O(f(n))
     - please **clearly write which case it is and the value of epsilon**(for case I and III), k (for case II) and check **regularity condition** and give the value of c if it is **case III**
6. **Heaps and Heap Sort**
   - Building Heap will always take O(n) time whether it is min heap or max heap
     - Here is a useful link for better understanding: Build Heap

- **FindMin** in MinHeap and **FindMax** in MaxHeap will take O(1) time but all other operations like Extract Min in MinHeap, Extract Max in Max Heap, Insert, Delete will take O(log n) time.

7. **Probability, Random variables and Expected values**
   - **Sample Questions**
     - Define Sample Space and Random Variables for given problem
     - Compute expected values of your random variables using definition of expectation/linearity of expectation
     - **Example:** Consider a game "Rolling Die". In this game one has to roll a fair four sided die two times and will win 10$ if he get same number in both times otherwise has to pay1$. What is the expected win/loss of this game? [Clearly define your **Sample Space** and **Random Variable**] Consider the same problem but the die is biased and probability of getting an even number{2,4} is twice the probability of getting an odd number{1,3}
7. **Randomized Algorithms**

Important Homework Problems

1. Show by definition of **O, Ω, θ, o,** (Homework 1: Problem 2)
2. For given code snippet **analyze the runtime**
3. Ranking of functions
4. For a given algorithm come up with a **Recurrence**
5. use **recursion tree** to guess the solution of a Recurrence
6. verify given solution using **big-Oh Induction**
7. For given recurrence relation solve it using **Master Theorem**.
8. Probability and Randomized Algorithms(Homework 5 problem 1,3)