CSE 100: Algorithm Design and Analysis. Midterm 1 key topics.
Spring 2023

Midterm 1 will be on Ch 1-4.
Disclaimer: I do not guarantee that this list is complete.

1. Ch1: Not much. Algorithm description. You can describe your algorithm by either plain English or pseudocode or by their combination. It should be formal. Conciseness will be a plus though.

2. Ch2:

   (a) RAM (Random-Access Machine): Recall the three key features.

   (b) Proving correctness using *loop invariant* and/or *induction*

      i. Loop Invariant: Used for iterative algorithms (for, while... loops). e.g. Insertion sort. Three steps. Initialization, maintenance, termination. Because of time constraints, you may be asked to write down only the loop invariant. More examples from discussion.

      ii. Induction: Use for recurrence. e.g. Merge sort.

   (c) *Worst case* vs average case running time.
   e.g. Suppose there are 2^n inputs to a certain problem. The running time of algorithm A is exactly 2^n for exactly one of the inputs, and 1 for any other input. Then, the running time is $O(1)$ since $2^n * \frac{1}{2^n} + 1 * (1 - \frac{1}{2^n}) \leq 2$. Is this statement correct?
   e.g. Suppose we have $\langle 1, 2, ..., n\rangle$ with probability $1 - 1/n$ and $\langle n, n-1, n-2, ..., 1\rangle$ with probability $1/n$. Then, what is the average running time of Insertion sort?

   (d) A glimpse at divide-and-conquer (merge-sort) and its runtime analysis.

   (e) You may see questions on some pseudocodes. e.g. Assuming that you have Merge, write a pseudocode for Mergesort. e.g. What goes wrong with the pseudocode with the base case removed?

3. Ch3:

   (a) Formal definition of asymptotic notations, O, Ω, Θ. E.g., prove $100n + 50n \log n = O(n^2)$ using the formal definition of O.

   (b) Their properties. e.g. Transitivity.

   (c) Ordering the given functions in asymptotically non-decreasing order.

4. Ch4:

   (a) How to solve recurrences.

      i. Solve the given recurrence using the recursion tree method. It is important that your solution shows the key quantities that yield the solution: Tree depth (log base could matter in some cases); Each subproblem size at depth d; Number of nodes at depth d; workload per node at depth d; and total workload at depth d.

ii. Solve the given recurrence using the Master theorem. The Master theorem will be given to you. Just need to state which case applies, the solution. If not applicable, state N/A.

iii. Solve the given recurrence using the substitution method.

(b) Examples

i. Max Subarray Problem. How to handle the crossing case.

ii. Matrix Multiplication. Only high-level structures of the pseudo-codes of the naive divide-and-conquer based algorithm and Strassen's algorithm.

(c) Can you write a recurrence for RT of an algorithm? e.g. Merge Sort, Max Subarray Problem. Matrix Multiplication.

5. Questions on Pseudocodes.

6. There may be a bonus problem worth 25+ points. The problem is for students who challenged themselves hard enough to study more beyond what was covered in class. It will be graded rigorously. It's not for collecting partial points.