CSE 100: Algorithm Design and Analysis. Midterm exam 2 key topics.
Spring 2023

Disclaimer: I do not guarantee that the following list is complete. I had planned to assign 18% weight to midterm 2. Because we couldn't finish ch12 and it will be covered by another exam, I will likely lower this exam's weight.

1. FYI: T/F questions (8 problems, each worth 2pts). Pseudo-codes from lab assignments. Two pseudo-code problems. One bonus problem (15 pts).

2. Ch06. Illustration of a heap operation on a given input: max-heapify, increase-key, building max-heap, etc.. Run time of heap operations. No need to study loop invariants. What is an in-place sorting algorithm? What are such sorting algorithms? What are not?

3. Ch07. Running time of deterministic/randomized quick sort. Revisiting worst-case RT vs average RT: e.g. The deterministic quick sort has an average RT of $\Theta(n \log n)$ when the input is a uniform random permutation of n elements. Yet, it has a run time of $\Omega(n^2)$ for some inputs. For any input, the randomized quick sort has an expected run time of $\Theta(n \log n)$. Run time analysis of randomized quicksort.

4. Ch08.

   (a) Any comparison based sorting algorithms require $\Omega(n \log n)$ time. Can you draw the decision tree of a sorting algorithm on small inputs? Understanding the decision tree of a sorting algorithm; why and how it gives a $\Omega(n \log n)$ lower bound for any comparison based sorting algorithm. E.g. What does the length of a path from the root to a leaf node mean?

   (b) The assumptions made in counting, radix, bucket sort to break this lower bound barrier.

   (c) Counting sort. What does stable sorting mean?

   (d) Radix sort. When do you use it? How do you implement it? Sorting n binary numbers, each of $100 \log_2 n$ bits. Sorting vectors, YY/MM/DD in lexicographic orders.

5. Ch09. Randomized Selection: how you make recursive calls based on the partition result and the order of the element to be found. Deterministic Selection. Understanding the run time analysis of both.

6. Ch11. Hashing. Direct addressing vs. Hashing. Universal Hash Family. Given a hash family, can you tell why it is bad? Resolving collisions using chains. Hash applications (e.g. discussion session problems).