1. Complexity
   • Be able to analyze and compare the time complexities of various algorithms using Big-O notation.
   • Be able to determine which is the best structure (from a list) for a given application.

2. Lists, Stacks, and Queues
   • Be able to work with these structures, using abstract operations or implementing new operations as needed or determine which is the best structure for a given application.

3. Recursion/Induction
   • Be able to prove the correctness of a recursive procedure for binary trees using induction, like the problem on the midterm.

4. Trees
   • Be able to show how to insert items into a splay tree
   • Be able to show how to insert items into a B+-tree

5. Hashing
   • Be able to show how separate chaining works on given data.
   • Be able to show how open addressing works with various collision-handling schemes (linear probing, quadratic probing, double hashing, rehashing or some given scheme) on given data.
   • Be able to determine when hashing is needed in the solution of an application problem.
   • Be able to analyze the complexity of given hashing schemes or algorithms that use them.

6. Heaps
   • Be able to show how to add items to binary min-heaps.
   • Be able to show how to do deleteMin operations.
   • Be able to determine when to use binary heaps (min or max) for some given application.
7. Union-Find (Up Trees)
   • Be able to show how to do union operations.
   • Be able to show how to do find operations.
   • Be able to determine when this is the best structure to use for some application.

8. Graphs and Digraphs
   • Be able to work with all the variations: directed graphs, undirected graphs, weighted and unweighted graphs, labeled and unlabeled graphs, etc.
   • Be able to use the two different representations we covered: adjacency matrices and adjacency lists.
   • Be able to show how the following algorithms work on given data:
     – breadth-first and depth-first traversal
     – topological sort
     – the Floyd-Warshall matrix algorithm for determining the minimum costs of all paths among vertices
     – the Dijkstra algorithm for finding the shortest path
     – the Kruskal algorithm for finding the minimal spanning tree of a weighted graph
     – the backtracking tree search algorithm for subgraph isomorphism.

9. Sorting
   • Be familiar with the algorithms and their complexities