1. AVL Tree Operations: (big O analysis & illustration of the insertion & deletion process)
2. Search and traversal – the same as BST. BigO analysis.
3. Insert a node
   1. search + add + rebalance. (understand how to conduct rebalance and be able to illustrate.)
   2. **LL and RR only one rotation, LR or RL needs two rotations**
4. Delete a node – the deleted node has no children; one child; or two children.
   1. search + remove + rebalance
5. Graph – each element can have multiple successors and multiple predecessors
6. Concepts: directed graph; undirected graph; vertex; edges; cycle; acyclic; dag; planar; clique;
7. Relationship between vertex and edges – given n vertices, maximum number of edges (directed/undirected graph)?
8. Graph representation:
   1. Operations: Given a graph, know how to represent it with adjacency matrix & adjacency list. Given an adjacency matrix or adjacency list, know how to draw the graph.
   2. Understand when to use adjacency matrix and when to use adjacency list. **If E << V^2 (sparse) use a list, but if E ~ V^2 either is good**
      1. When you care more about a given vertex’s neighbors – adjacency list
      2. When you care more about whether an edge exists or not – adjacency matrix (very few ones, not many edges, far from V^2, so sparse graph)
9. Graph traversal – Worklist algorithm
   1. Be able to do depth-first traversal – stack based worklist
   2. Be able to do Breadth-first traversal – queue based worklist
10. Sorting – understand the sorting category; coding for basic sorting algorithms: selection sort, insertion sort and bubble sort; be able to illustrate the sorting process; big O and stability analysis.
11. Selection based sorting
    1. Selection sort **(unstable, but can be made stable)**
    2. Heap sort **(unstable)**
12. Insertion based sorting
    1. Insertion sort **(stable)**
    2. Shell sort **(unstable)** works well for large scale data sets
13. Exchange based sorting
    1. Bubble sort **(stable)**
    2. Quick sort **(unstable)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Best case | Average case | Worst case | Stability |
| Selection sort | O(n^2) | O(n^2) | O(n^2) | Unstable (default) |
| Heap sort | O(nlogn) | O(nlogn) | O(nlogn) | Not stable |
| Insertion sort | O(n) | O(n^2) | O(n^2) | Stable |
| Shell sort | O(nlogn) | Quadratic | Quadratic | Not stable |
| Bubble sort | O(n) | O(n^2) | O(n^2) | Stable |
| Quick sort | O(nlogn) | O(nlogn) | O(n^2) | Not stable |
| Merge sort | O(nlogn) | O(nlogn) | O(nlogn) | stable |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SET** | **Unsorted Array** | **Sorted Array** | **Hash**  **Table** | **Unsorted Linked List** | **Sorted Linked List** | **BST** | **AVL tree** |
| Search | O(n) | O(log n) | O(n)  (expected O(1)) | O(n) | O(n) | O(h)  (Log(n) <= h <= n) | log(n) |
| Add | O(n) + O(1) = O(n) | O(log n)+O(n) = O(n) | O(n)  (expected O(1)) | O(n) + O(1) = O(n) | O(n) + O(1) = O(n) | O(h)+ O(1) = O(h) | log(n) |
| Remove | O(n) + O(1) = O(n) | O(log n)+O(n) = O(n) | O(n)  (expected O(1)) | O(n) + O(1) = O(n) | O(n) + O(1) = O(n) | O(h)+ O(1) = O(h) | log(n) |
| Min/Max | O(n) | O(1) | O(**m**) | O(n) | O(1)  (assuming fast access to tail) | O(h)  (Log(n) <= h <= n) | log(n) |