DASC 5300: Fundamentals of Computing
University of Texas at Arlington
Fall 2023
Alex Dillhoff

Exam 2: Data Structures & Algorithms

Complexity Analysis

Outcomes

1. Know how to compute the running time $T(n)$ of an algorithm.
2. Identify the rate of growth of a function.
3. Be able to identify the best, worst, and average case complexity of an algorithm.
4. Be able to identify both the space and time complexity of an algorithm.

Suggested Practice

1. Read Chapter 2 of *Data Structures and Algorithms in Python.*
2. State the running time of the worst and best-case scenarios for the BubbleSort algorithm.
3. State the running time of the worst and best-case scenarios for the MergeSort algorithm.
4. State the big-Oh complexity of the algorithms above given its rate of growth.

Linked Lists

Outcomes

1. Know how to implement a linked list.
2. Know how to traverse a linked list.
3. Know how to insert and delete nodes in a linked list.
4. Know the running time of the operations above.

Suggested Practice

1. Read Chapter 3 of *Data Structures and Algorithms in Python.*
2. Implement a linked list in Python.

3. Implement a function that inserts a node at the end of a linked list.
4. Implement a function that deletes a node from a linked list.
5. Implement a function that reverses a linked list.
6. Implement a function that returns the kth to last element of a linked list.

Stacks and Queues

Outcomes

1. Know how to implement a stack and queue.
2. Know the running time of push, pop, enqueue, and dequeue.
3. Know the difference between implementing a stack or queue with an array versus a linked list.

Suggested Practice

1. Implement a stack in Python.
2. Implement a queue in Python.
3. Implement a function that checks if a string is a palindrome using a stack.

Hash Maps

Outcomes

1. Know how to implement a hash map.
2. Know the running time of the operations of a hash map.
3. Know how to handle collisions in a hash map.

Suggested Practice

1. Implement a hash map in Python.
2. Be able to insert values into a hash table and handle collisions.

Red-Black Trees

Outcomes

1. Know how to implement a red-black tree.
2. Know the running time of the operations of a red-black tree.
3. Know how to insert and delete nodes in a red-black tree.

Suggested Practice

1. Implement a red-black tree in Python.
2. Implement a function that inserts a node into a red-black tree.
3. Implement a function that deletes a node from a red-black tree.
4. Implement a function that checks if a red-black tree is balanced.

Graphs

Outcomes

1. Know how to implement a graph.
2. Know how to traverse a graph.
3. Know how to implement a graph using an adjacency matrix and adjacency list.
4. Know the running time of the operations of a graph.

Suggested Practice

1. Implement a function that traverses a graph using breadth-first search.
2. Implement a function that traverses a graph using depth-first search.
3. Implement a function that checks if a graph is bipartite.

Minimum Spanning Trees

Outcomes

1. Know how to find the MST of a graph using Prim's and Kruskal's algorithms.
2. Know the running time of the operations of Prim's and Kruskal's algorithms.

Suggested Practice

1. Implement Prim's algorithm in Python.
2. Implement Kruskal's algorithm in Python.

Shortest Path Algorithms

Outcomes

1. Know how to find the shortest path of a graph using Dijkstra's and Bellman-Ford algorithms.
2. Know the running time of the operations of Dijkstra's and Bellman-Ford algorithms.

3. Know how to handle negative edge weights in a graph.
4. Know how to handle negative cycles in a graph.

Suggested Practice

1. Implement Dijkstra's algorithm in Python.
2. Implement Bellman-Ford algorithm in Python.
3. Implement a function that checks if a graph has a negative cycle.