CSE 137: Data Structures Course Outline

Chapter	Content	Comment

1 (Fundamentals)	☞ Pseudocode Writing
	☞ Flow Chart Drawing
	☞ Sorting
	  ↬ Bubble Sort
	  ↬ Insertion Sort
	  ↬ Selection Sort
	  ↬ Merge Sort
	  ↬ Quick Sort
	  ↬ Radix Sort
	  ↬ Bucket Sort
	☞ Searching
	  ↬ Linear Search
	  ↬ Binary Search
	☞ Complexity analysis	

2 (Recursion, Stack and Queue)	☞ Recursion
	  ↬ Ackerman Function
	  ↬ Tower of Hanoi
	☞ Queue
	  ↬ Push
	  ↬ Pop
	  ↬ Front/Peek
	  ↬ Linear Queue
	  ↬ Circular Queue
	  ↬ Using Queue of STL
	  ↬ Implementation using Array
	  ↬ Implementation using Pointer
	  ↬ Priority Queue
	  ↬ Dequeue
	☞ Stack
	  ↬ Push
	  ↬ Pop
	  ↬ Top
	  ↬ Using Stack of STL
	  ↬ Implementation using Array
	  ↬ Implementation using Pointer
	☞ Problems
	  ↬ Permutation
	  ↬ Combination
	  ↬ Parenthesis Balance using Stack
	  ↬ Palindrome Checker using Stack and Queue	

3 (Binary Tree, Ternary Tree, BST, AVL Tree, TST, Heap, BIT, Segment Tree, RMQ)	☞ Binary Tree
	  ↬ Representation using Array
	  ↬ Representation using Pointer
	  ↬ Traversal
	  ▪ In-order
	  ▪ Pre-order
	  ▪ Post-order
	☞ Binary Search Tree	

	↳ Representation ↳ Basic Operations ▪ Creating ▪ Insertion ▪ Deletion ▪ Querying/Searching ▪ Traversing ☞ AVL Tree ↳ Rotation ↳ Insertion ☞ Heap ↳ Min-Heap ↳ Max-Heap ↳ Fibonacci-Heap ↳ Heap Sort ☞ Ternary Search Tree ☞ Binary Indexed Tree/Fenwick tree ☞ Segment Tree ☞ Range Minimum Query (RMQ) ☞ Expression Conversion and Evaluation ↳ In-fix to post-fix expressions Conversion ↳ Post-fix expression Evaluation	
4 (Graph)	☞ Graph Representation ↳ Adjacency List ↳ Adjacency Matrix ☞ Basic Operations on Graph ↳ Node/edge Insertion ↳ Node/edge Deletion ☞ Traversing a graph ↳ Breadth First Search (BFS) ↳ Depth First Search (DFS)	
5 (Graph Algorithms and Techniques)	☞ Topological Sort ☞ Strongly Connected Components (SCC) ☞ Euler Path ☞ Articulation Point ☞ Articulation Bridge ☞ Bi-connected Components ☞ Graph-bicoloring ☞ Floodfill ☞ Dijkstra's Shortest Path Algorithm ☞ Bellman-Ford Algorithm and Negative Cycle Detection ☞ Floyd-Warshall all pair shortest path Algorithm ☞ Johnson's Algorithm ☞ Shortest Path in Directed Acyclic Graph ☞ Minimum Spanning Tree ↳ Prim's Algorithm ↳ Kruskal's Algorithm	
7 (Miscellaneous)	☞ Disjoint Set (Union Find) ☞ Huffman Coding ☞ Set Operations ↳ Set Representation using bitmask	

	↳ Set/Clear bit ↳ Querying Status of a bit ↳ Toggling bit values ↳ LSB ↳ Application of Set Operations ☞ String – Abstract Data Types (ADT) ↳ Concatenation of Two Strings ↳ The Extraction of Substrings ↳ Searching a string for a matching substring ↳ Parsing ☞ Trie Tree ☞ Suffix Tree ☞ Suffix Array	

CSE 138: Data Structures Lab Course Outline

Lab No	Content	Comment
0 (Basics)	☞ Basics of C++ ↳ Constructor ↳ Destructor ↳ Implementation of function of a structure externally ↳ Memory Declaration using `new` keyword ↳ Memory Deletion using `delete` keyword ☞ Basic Terminology ↳ front ↳ rear ↳ top ↳ enqueue/push ↳ dequeue/pop ↳ head ↳ tail ↳ overflow ↳ underflow	
1 (Singly Linked List)	☞ Implementation of Singly Linked List ☞ Implementation of different functions of Singly Linked List ↳ `void insertVal(int val);` ↳ `void insertValBeforeTail(int val);` ↳ `void insertAtHead(int val);` ↳ `void insertAfterHead(int val);` ↳ `void insertAtPos(int val, int pos);` ↳ `int findVal(int val);` ↳ `int findValAtPos(int pos);` ↳ `void traverse();` ↳ `void deleteVal(int val);` ↳ `void deleteOne(int val);` ↳ `void deletePos(int pos);` ↳ `void deleteAll();` ↳ `void deleteHead();` ↳ `void deleteTail();`	
2 (Doubly Linked List)	☞ Implementation of Doubly Linked List ☞ Differences between Singly Linked List and Doubly Linked List ☞ Implementation of different functions of Doubly Linked List ↳ `void insertValAtTail(int val);` ↳ `void insertValBeforeTail(int val);` ↳ `void insertAtHead(int val);` ↳ `void insertAfterHead(int val);` ↳ `void insertAtPos(int val, int pos);` ↳ `void insertAtRevPos(int val, int pos);` ↳ `int findValFromHead(int val);` ↳ `int findValFromTail(int val);` ↳ `int findValAtPosFromHead(int pos);` ↳ `int findValAtPosFromTail(int pos);` ↳ `void traverse();` ↳ `void reverselyTraverse();`	

	↳ `void deleteVal(int val);` ↳ `void deletePosFromHead(int pos);` ↳ `void deletePosFromTail(int pos);` ↳ `void deleteAll();` ↳ `void deleteHead();` ↳ `void deleteTail();`	
3 (Queue)	☞ Implementation of Linear Queue using Array ↳ Enqueue/Push ↳ Dequeue/Pop ↳ Handling Overflow ↳ Handling Underflow ☞ Implementation of Circular Queue using Array ↳ Enqueue/Push ↳ Dequeue/Pop ↳ Handling Overflow ↳ Handling Underflow ↳ Differences with Linear Queue ☞ Implementation of Queue using Pointer ↳ Initialization of front and rear ↳ Enqueue/Push ↳ Dequeue/Pop ↳ Handling Underflow	
4 (Stack)	☞ Implementation of Stack using Array ↳ Enqueue/Push ↳ Dequeue/Pop ↳ Handling Overflow ↳ Handling Underflow ☞ Implementation of Stack using Pointer ↳ Initialization of top ↳ Enqueue/Push ↳ Dequeue/Pop ↳ Handling Underflow	
5 (Basic Sorting)	☞ Bubble Sort Review ↳ Visualization ↳ Implementation ↳ Complexity Analysis ▪ Best case ▪ Worst case ▪ Average Case ↳ Analyzing Best Case and Optimization ☞ Insertion Sort ↳ Visualization ↳ Implementation ↳ Complexity Analysis ▪ Best case ▪ Worst case ▪ Average case ↳ Analyzing Best Case and Optimization ☞ Selection Sort ↳ Visualization ↳ Implementation	

	↳ Complexity Analysis ▪ Best case ▪ Worst case ▪ Average case ↳ Why this sorting algorithm cannot be improved more?
6 (Merge Sort)	☞ Visualization of Merge Sort ☞ Implementation ☞ Complexity Analysis ↳ Best case ↳ Worst case ↳ Average case ☞ What is the drawbacks of this sorting algorithm? ☞ Optimized Implementation using two global arrays ☞ Reducing Space Complexity using Linked List
7 (Recursion and Binary Search)	☞ Implementation of Ackerman Function ☞ Tower of Hanoi ☞ Permutation ☞ Combination ☞ Implementation of Binary Search
8 (Quick Sort)	☞ Visualization of Quick Sort ☞ Implementation ☞ Complexity Analysis ↳ Best case ↳ Worst case ↳ Average case ☞ Comparison with Merge Sort ☞ Optimization and Randomization of Quick Sort
9 (Expression Conversion and Evaluation)	☞ Infix to postfix expression conversion ☞ Postfix Expression Evaluation ☞ Analysis of operators having same precedence and associativity
10 (Introduction to Graph and Graph Algorithm)	☞ Representation of Graph (Node, Edge) ↳ Adjacency Matrix ↳ Adjacency List ↳ Traversing Adjacency matrix/list ↳ 4-adjacent ↳ 8-adjacent ☞ Finding a path from a node to another node
11 (Binary Search Tree, Heap and AVL Tree)	☞ Binary Search Tree (BST) ↳ Insertion ↳ Deletion ↳ Searching ↳ Complexity Analysis ☞ Ternary Search Tree (TST) ☞ Min-Heap/Max-Heap ↳ Creation ↳ Insertion ↳ Deletion ☞ Applications of Heap ☞ Binary Heap ↳ Array Representation: Index Calculation

	☞ Binomial Heap
	☞ Fibonacci Heap
	☞ Leftist Heap
	☞ K-ary Heap
	☞ Heap Sort
	☞ AVL Tree
	  ↳ Insertion
	  ↳ Deletion
	  ↳ Rotation	

12 (Counting Sort, Radix Sort)	☞ Basics of Counting Sort ☞ Radix Sort ☞ Bucket Sort ☞ Complexity Analysis	

13 (Graph Applications)	☞ Breadth First Search (BFS)
	  ↳ Visualization
	  ↳ Implementation
	  ↳ Complexity Analysis
	☞ Depth First Search (DFS)
	  ↳ Visualization
	  ↳ Implementation
	  ↳ Complexity Analysis
	☞ Dijkstra Algorithm
	☞ Disjoint Set/ Union Find
	☞ Minimum Spanning Tree (MST)
	  ↳ Prim's Algorithm
	  ↳ Kruskal's Algorithm
	☞ Topological Sort
	  ↳ Implementation
	  ↳ One solution
	  ↳ All Possible Solution
	  ↳ Complexity Analysis
	☞ Floyd-Warshall Algorithm
	  ↳ Implementation
	  ↳ Binary Solution
	  ↳ Weighted Solution
	  ↳ Complexity Analysis	

14 (Advanced Graph Topics and Miscellaneous)	☞ B-Tree ☞ Articulation Bridge ☞ Articulation Point ☞ Bi-connected Component ☞ Segment Tree ☞ Lazy Propagation ☞ Range Minimum Query (RMQ) ☞ Strongly Connected Component (SCC) ☞ Directed Acyclic Graph (DAG) ☞ Bellman-Ford Algorithm ↳ Trie Tree (Array, Pointer Implementation) ☞ Suffix Array, Suffix Tree ☞ Aho–Corasick algorithm ☞ Huffman Coding	