DATA STRUCTURES vs ABSTRACT DATA TYPES (ADTs) vs ALGORITHMS

- ADTs – what you want to do
    - Represent a set of behaviours on a domain of data types
- Data structures – how you will implement it
    - Ways of organizing data to implement an ADT
- Algorithms
    - Using ADTs to achieve something

Contiguous vs. Linked Data Structures:

- Contiguously allocated – single slabs of memory
    - Arrays, matrices, heaps, hash tables
- Linked – distinct chunks of memory connected by pointers
    - Lists, trees, graphs

ABSTRACT DATA TYPES (ADTs)

Stacks

- LIFO retrieval
- Operations:
    - Push(x, s): insert item x at top of stack s
    - Pop(s): Remove and return top of stack s
- Data Structure implementation:
    - Arrays
    - Linked lists

Queues

- FIFO retrieval
- Operations:
    - Enqueue(x, q): Insert x at back of q
    - Dequeue(q): Remove and return front of q
- Data Structure implementation:
    - Arrays
    - Linked lists

Priority Queue

- Retrieval: highest priority dequeued first
- Operations:
    - Insert(x, q): insert element with priority x into q
    - Find_min(q): return element with smallest value from q
    - Delete_min(q): remove item with smallest value from q
- Data Structure implementation:
    - Unsorted array/linked list

        - O(1) for insert, O(n) for extraction
    - Sorted array/linked list
        - O(n) for insert, O(1) for extraction
    - Heap (most efficient implementation of PQ)
        - O(logn) for insert, O(logn) for extraction, O(1) for returning min

Graphs

- Can have directed and weighted graphs
- Data Structure implementation:
    - Adjacency Matrix
    - Adjacency List
        - Each node has a list of connections
- Traversal Algorithms:
    - Breadth First Search
    - Depth First Search
- Shortest Path Algorithms:
    - Dijkstra's Algorithm
    - A* Algorithm

Dictionary/Map

- Collection of key-value pairs
- Operations:
    - Get(D, k): get value at k
    - Put(D, k, v): insert key-value pair k-v into D
- Data Structure implementation:
    - Hash Tables

DATA STRUCTURES

Array

Linked List (refer to lab 4 solutions)

Heap (refer to minpq lecture code)

- Complete tree (filled in from left to right)
- Parent < child (for min pqs)
- Stored as an array
- Operations:
    - Insert: insert item at last position, percolate up
    - Remove: swap first and last position, remove last item (min), percolate down
- Heapify
    - Starting from parent of last item and working down to first item, percolate down
- Heapsort

    - Heapify array, then remove min one at a time (and put into new array)

Binary Search Tree (BST) (refer to lecture code for C and lab 6 for python)

- All nodes in left subtree < root; right subtree > root
    - Need comparator function in struct of BST
- Operations:
    - Insert: compare item with root, recursively compare item with left/right subtree
        - Until root = NULL
    - Delete:
        - No children: delete
        - One child: replace node with child
        - Two children: swap node with next largest node (successor), delete node
            - To find successor: in right subtree, find left most descendant

AVL Tree (refer to lecture code)

- Balanced BST – for every node, height of right and left subtree differs by at most 1
- Must balance the tree after every insertion/deletion

Hash Tables (refer to lecture code)

- Array of table elements (key-value pairs)
- Requires hash function, hash(key, table_size) – returns what index in array to place item
- Closed vs Open hashing:
    - Closed – keeps data inside table (uses probing to avoid collisions)
        - Probe function, probe(i, key) – returns how much to add to current index to determine which index you should try next
    - Open – keeps data in a linked list at each index

ALGORITHMS

Graph Traversal Algorithms (refer to lab 5)

- Each node is either undiscovered, discovered, or processed (in that order)
    - Processed: visited all its edges
- Breadth First Search (BFS) – keeps queue of unprocessed nodes
    - Start with a node in Q
    - While Q is not empty:
        - Dequeue (processed the node)
        - Enqueue node's undiscovered connections
- Depth First Search (DFS) – keeps stack of unprocessed nodes (can also do recursively)
    - Start with a node in stack
    - While stack is not empty:
        - Pop (processed the node)
        - Push node's undiscovered connections

Shortest Path Algorithms (Graphs)

Dijkstra's Algorithm (refer to lab 5 solutions)

- Calculates shortest path between given vertex and all vertices in graph (and previous vertex)
- Have list of visited nodes (and their shortest path)
- Have PQ of unvisited nodes with their priority value being distance to starting node (algorithm visits the next most promising node)
- While PQ is not empty:
    - Dequeue to visit node, add data to visited node list
    - When visiting a node, iterate through each of its connections, update info in PQ

A* Algorithm

- Same as Dijkstra but uses a heuristic function to determine a node's priority in queue

OTHER

Shifting Bits

- Dividing/multiplying by 2
    - Equivalent of shifting decimal place in base 10 (div/mult by 10)
- Shift right is dividing: a >> 2
- Shift left is multiplying: a << 2

Strings (refer to lecture code and lab 3)