**Algorithm and Data Structure**

**Algorithm:** a collection of steps to solve a problem.

**Data Structure:** a named location that can be used to store and organize data.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Data Structure** | **Definition** | **Purpose** | **Space** | **Search** | **Insert** | **Delete** |
| Stack | LIFO: Last-In First-Out   * Stores objects into a sort of “vertical tower” * push() to add to the top * pop() to remove from the top | 1. Undo/redo features in text editors  2. Moving back/forward through browser history  3. Backtracking algorithms (maze, file directories)  4. Calling functions (call stack) |  |  |  |  |
| Queue | FIFO: First-In First-Out   * A collection designed for holding elements prior to processing * Linear data structure * add = enqueue, offer() * remove = dequeue, poll() | 1. Keyboard Buffer  2. Printer Queue  3. Used in LinkedLists, PriorityQueues, Breadth-first search |  |  |  |  |
| Priority Queue | A FIFO data structure that serves elements with the highest priorities first before elements with lower priority. |  |  |  |  |  |
| Linked List | * Stores Nodes in 2 parts (data + address) * Nodes are in non-consecutive memory locations * Elements are linked using pointers   Singly Linked List:  [data | address] -> [data | address]  Doubly Linked List**:**  [address | data | address]  <-> [address | data | address] | 1. Implement Stack/Queues  2. GPS navigation  3. Music playlist |  | O(n) | O(1) | O(1) |
| Dynamic Array | * In Java, known as ArrayList * Dynamic capacity 🡪 array size increases by factor of 3/2 every time you reach over capacity |  |  | O(1) |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |