Data Structures and Algorithms

Agenda

- Array vs Linked List
- Singly Circular Linked List
- Doubly Linear Linked List

Array vs Linked List

- A: Array size is fixed.
- L: Linked list size is dynamic (i.e. it can grow/shrink at runtime).
- A: Arrays are stored in contigous memory locations.
- L: Linked list nodes are not stored in contigous locations.
- A: Array elements can be accessed sequentially or randomly faster.
- L: Linked list elements can only be accessed sequentially.
- A: Insertion/deletion of the element in array will be slower (shifting of next elements).
- L: Insertion/deletion of the element in linked list is faster (no shifting of other elements).
- A: Needs less memory (compared to list).
- L: Needs more memory (data + overheads like next, head, ...)
- A: Arrays can be allocate statically as well as dynamically. (e.g. C++: int arr[5]; or int *arr = new int[n] 😉
- L: Linked list is always allocated dynamically (i.e. C++: new operator, malloc(), ...).

Queue using Linked List

- When linear queue is implemented using arrays, there is poor memory utilization. So circular queue is commonly implemented (with arrays). The queue is called circular, because front/rear can be incremented in circular fashion (i.e. 0, 1, 2, ..., n-1, 0, 1, ...).
- Whene queue is implemented using linked list, there is no issue of memory utilization. So there is no concept of circular queue required here.
- However queue can also be implemented using circular linked list.

Singly Circular Linked List

Doubly Linear Linked List