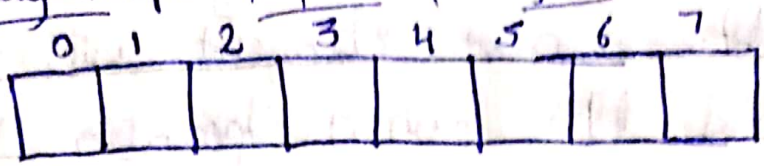
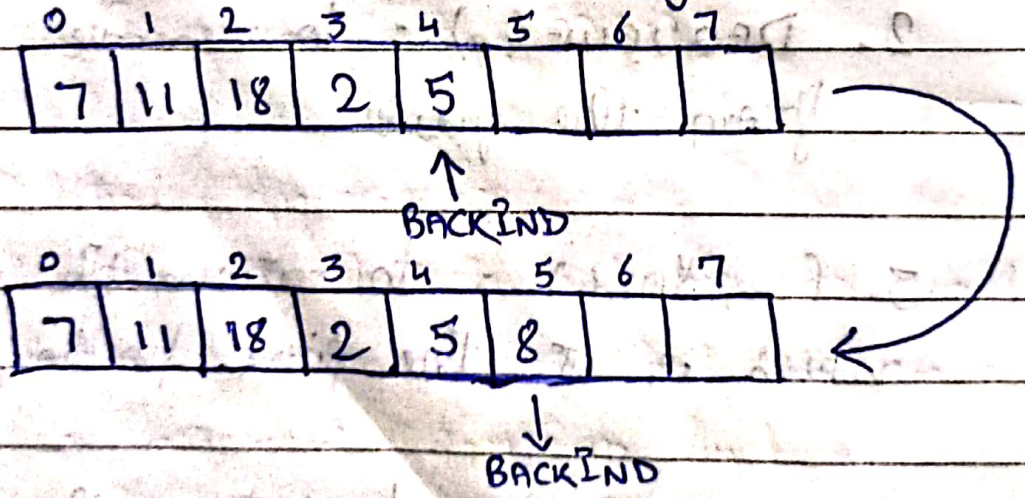
Array Implementation of Queue

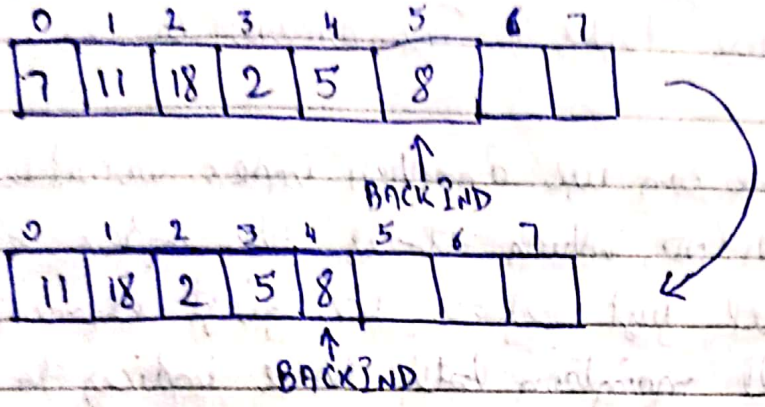


Here, we'll maintain index variable BACKIND, to store the index of the rearmost element. So, when we insert an element, we just increment the value of the BACKIND and insert the element at the current BACKIND value. Follow the array below to know how inserting works:



→ Now suppose we want to remove an element from the queue. And since a queue follows the FIFO discipline, we can only remove the element at the zeroth index, as that is the element inserted first in the queue. So, now we

will remove the element at the zeroth index and shift all the elements to its adjacent left.



→ But this removal of the zeroth element and shifting of other elements to their immediate left features O(n) time complexity.

① Insertion (enqueue)
- → Increment BACKIND by 1.
- → Insert the element.
- → Time complexity : O(1)

② Deletion (dequeue)
- → Remove the element at the zeroth index
- → Shift all other elements to their immediate left.
- → Decrement BACKIND by 1.

③ Here, our first element is at index 0, and the rearmost element is at index BACKIND.

④ Condition for queue empty BACKIND = -1.

⑤ Condition for queue full: BACKEND = SIZE-1.

→ Can there be a better way to accomplish these tasks? The answer is Yes.

→ We can use another index variable called FRONTEND, which stores the index of the cell just before the first element. We'll maintain both these indices to bring about all our operations. Let's now enlist the changes we'll see after we introduce this new variable:

① Insertion (enqueue):
→ Increment BACKEND by 1.
→ Insert the element.
→ Time complexity: O(1).

② Deletion (dequeue):
→ Remove the element at the zeroth index (no need for that in an array.)
→ Increment FRONTEND by 1.
→ Time complexity: O(1).

③ Our first element is at index FRONTEND+1, and the rearmost element is at index BACKEND.

④ Condition for queue empty: FRONTEND = BACKEND.

Page No. _____

Date | | | |

⑤ Condition for queue full : BACKIND = SIZE-1.