ALGORITHM FOR TRAVERSING A LINKED LIST

```
Step 1: [INITIALIZE] SET PTR = START
Step 2: Repeat Steps 3 and 4 while PTR != NULL
Step 3: Apply Process to PTR->DATA
Step 4: SET PTR = PTR->NEXT [END OF LOOP]
Step 5: EXIT
```

ALGORITHM TO SEARCH A LINKED LIST

```
Step 1: [INITIALIZE] SET PTR = START
Step 2: Repeat Step 3 while PTR != NULL
Step 3: IF VAL = PTR->DATA SET POS = PTR Go To Step 5 ELSE SET PTR = PTR->NEXT
 [END OF IF] [END OF LOOP]
Step 4: SET POS = NULL
Step 5: EXIT
```

ALGORITHM TO INSERT A NEW NODE IN THE BEGINNING OF THE LINKED LIST

```
Step 1: IF AVAIL = NULL, then Write OVERFLOW Go to Step 7 [END OF IF]
Step 2: SET New_Node = AVAIL
Step 3: SET AVAIL = AVAIL->NEXT
Step 4: SET New_Node->DATA = VAL
Step 5: SET New_Node->Next = START
Step 6: SET START = New_Node
Step 7: EXIT
```

ALGORITHM TO INSERT A NEW NODE AT THE END OF THE LINKED LIST

```
Step 1: IF AVAIL = NULL, then Write OVERFLOW Go to Step 10 [END OF IF]
Step 2: SET New_Node = AVAIL
Step 3: SET AVAIL = AVAIL->NEXT
Step 4: SET New_Node->DATA = VAL
Step 5: SET New_Node->Next = NULL
Step 6: SET PTR = START
Step 7: Repeat Step 8 while PTR->NEXT != NULL
Step 8: SET PTR = PTR ->NEXT [END OF LOOP]
Step 9: SET PTR->NEXT = New_Node
Step 10: EXIT
```

ALGORITHM TO INSERT A NEW NODE AFTER A NODE THAT HAS VALUE NUM

```
Step 1: IF AVAIL = NULL, then Write OVERFLOW Go to Step 12 [END OF IF]
Step 2: SET New_Node = AVAIL
Step 3: SET AVAIL = AVAIL->NEXT
Step 4: SET New_Node->DATA = VAL
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR
Step 7: Repeat Steps 8 and 9 while PREPTR->DATA != NUM
Step 8: SET PREPTR = PTR
Step 9: SET PTR = PTR->NEXT [END OF LOOP]
Step 10: PREPTR->NEXT = New_Node
Step 11: SET New_Node->NEXT = PTR
Step 12: EXIT
```

Algorithm to delete the first node from the linked list

```
Step 1: IF START = NULL, then Write UNDERFLOW Go to Step 5 [END OF IF]
Step 2: SET PTR = START
Step 3: SET START = START->NEXT
Step 4: FREE PTR
Step 5: EXIT
```

ALGORITHM TO DELETE THE LAST NODE OF THE LINKED LIST

```
Step 1: IF START = NULL, then Write UNDERFLOW Go to Step 8 [END OF IF]
Step 2: SET PTR = START
Step 3: Repeat Steps 4 and 5 while PTR->NEXT != NULL
Step 4: SET PREPTR = PTR
Step 5: SET PTR = PTR->NEXT [END OF LOOP]
Step 6: SET PREPTR->NEXT = NULL
Step 7: FREE PTR
Step 8: EXIT
```

ALGORITHM TO DELETE THE NODE AFTER A GIVEN NODE FROM THE LINKED LIST

```
Step 1: IF START = NULL, then Write UNDERFLOW Go to Step 10 [END OF IF]
Step 2: SET PTR = START
Step 3: SET PREPTR = PTR
Step 4: Repeat Step 5 and 6 while PRETR->DATA != NUM
Step 5: SET PREPTR = PTR
Step 6: SET PTR = PTR->NEXT [END OF LOOP]
Step7: SET TEMP = PTR->NEXT
Step 8: SET PREPTR->NEXT = TEMP->NEXT
Step 9: FREE TEMP
Step 10: EXIT
```

Algorithm to insert a new node in the beginning of the doubly linked list

```
Step 1: IF AVAIL = NULL, then Write OVERFLOW Go to Step 8 [END OF IF]
Step 2: SET New_Node = AVAIL
Step 3: SET AVAIL = AVAIL->NEXT
Step 4: SET New_Node->DATA = VAL
Step 5: SET New_Node->PREV = NULL
Step 6: SET New_Node->Next = START
Step 7: SET START = New_Node
Step 8: EXIT
```

Algorithm to insert a new node at the end of the doubly linked list

```
Step 1: IF AVAIL = NULL, then Write OVERFLOW Go to Step 11 [END OF IF]
Step 2: SET New_Node = AVAIL
Step 3: SET AVAIL = AVAIL->NEXT
Step 4: SET New_Node->DATA = VAL
Step 5: SET New_Node->Next = NULL
Step 6: SET PTR = START
Step 7: Repeat Step 8 while PTR->NEXT != NULL
Step 8: SET PTR = PTR->NEXT [END OF LOOP]
Step 9: SET PTR->NEXT = New_Node
Step 10: New_Node->PREV = PTR
Step 11: EXIT
```

Algorithm to insert a new node after a node that has value NUM

```
Step 1: IF AVAIL = NULL, then Write OVERFLOW Go to Step 11 [END OF IF]
Step 2: SET New_Node = AVAIL
Step 3: SET AVAIL = AVAIL->NEXT
Step 4: SET New_Node->DATA = VAL
Step 5: SET PTR = START
Step 6: Repeat Step 8 while PTR->DATA != NUM
Step 7: SET PTR = PTR->NEXT [END OF LOOP]
Step 8: New_Node->NEXT = PTR->NEXT
Step 9: SET New_Node->PREV = PTR
Step 10: SET PTR->NEXT = New_Node
Step 11: EXIT
```

Algorithm to delete the first node from the doubly linked list

```
Step 1: IF START = NULL, then Write UNDERFLOW Go to Step 6 [END OF IF]
Step 2: SET PTR = START
Step 3: SET START = START->NEXT
Step 4: SET START->PREV = NULL
Step 5: FREE PTR
Step 6: EXIT
```

Algorithm to delete the last node of the doubly linked list

```
Step 1: IF START = NULL, then Write UNDERFLOW Go to Step 7 [END OF IF]
Step 2: SET PTR = START
Step 3: Repeat Step 4 and 5 while PTR->NEXT != NULL
Step 4: SET PTR = PTR->NEXT [END OF LOOP]
Step 5: SET PTR->PREV->NEXT = NULL
Step 6: FREE PTR
Step 7: EXIT
```

Algorithm to delete the node after a given node from the doubly linked list

```
Step 1: IF START = NULL, then Write UNDERFLOW Go to Step 9 [END OF IF]
Step 2: SET PTR = START
Step 3: Repeat Step 4 while PTR->DATA != NUM
Step 4: SET PTR = PTR->NEXT [END OF LOOP]
Step 5: SET TEMP = PTR->NEXT
Step 6: SET PTR->NEXT = TEMP->NEXT
Step 7: SET TEMP->NEXT->PREV = PTR
Step 8: FREE TEMP
Step 9: EXIT
```

Algorithm to PUSH an element in a stack

```
Step 1: IF TOP = MAX-1, then PRINT “OVERFLOW” Goto Step 4 [END OF IF]
Step 2: SET TOP = TOP + 1
Step 3: SET STACK[TOP] = VALUE
Step 4: END
```

Algorithm to POP an element from a stack

```
Step 1: IF TOP = NULL, then PRINT “UNDERFLOW” Goto Step 4 [END OF IF]
Step 2: SET VAL = STACK[TOP]
Step 3: SET TOP = TOP - 1
Step 4: END
```

Algorithm for Peek Operation

```
Step 1: IF TOP =NULL, then PRINT “STACK IS EMPTY” Go TO Step 3 [END OF IF]
Step 2: RETURN STACK[TOP]
Step 3: END
```

Algorithm to PUSH an element in a linked stack

```
Step 1: Allocate memory for the new node and name it as New_Node
Step 2: SET New_Node->DATA = VAL
Step 3: IF TOP = NULL, then SET New_Node->NEXT = NULL SET TOP = New_Node ELSE SET New_node->NEXT = TOP SET TOP = New_Node [END OF IF]
Step 4: END
```

Algorithm to insert an element in a queue

```
Step 1: IF REAR=MAX-1, then; Write OVERFLOW Goto Step 4 [END OF IF]
Step 2: IF FRONT == -1 and REAR = -1, then SET FRONT = REAR = 0 ELSE SET REAR = REAR + 1 END OF IF]
Step 3: SET QUEUE[REAR] = NUM
Step 4: Exit
```

Algorithm to delete an element from a queue

```
Step 1: IF FRONT = -1 OR FRONT > REAR, then Write UNDERFLOW Goto Step 2 ELSE SET VAL = QUEUE[FRONT] SET FRONT = FRONT + 1 [END OF IF]
Step 2: Exit
```

Algorithm to insert an element in a linked queue

```
Step 1: Allocate memory for the new node and name it as PTR
Step 2: SET PTR->DATA = VAL
Step 3: IF FRONT = NULL, then SET FRONT = REAR = PTR SET FRONT->NEXT = REAR->NEXT = NULL ELSE SET REAR->NEXT = PTR SET REAR = PTR SET REAR->NEXT = NULL [END OF IF]
Step 4: END
```

Algorithm to delete an element from a linked queue

```
Step 1: IF FRONT = NULL, then Write “Underflow” Go to Step 5 [END OF IF]
Step 2: SET PTR = FRONT
Step 3: FRONT = FRONT->NEXT
Step 4: FREE PTR
Step 5: END
```

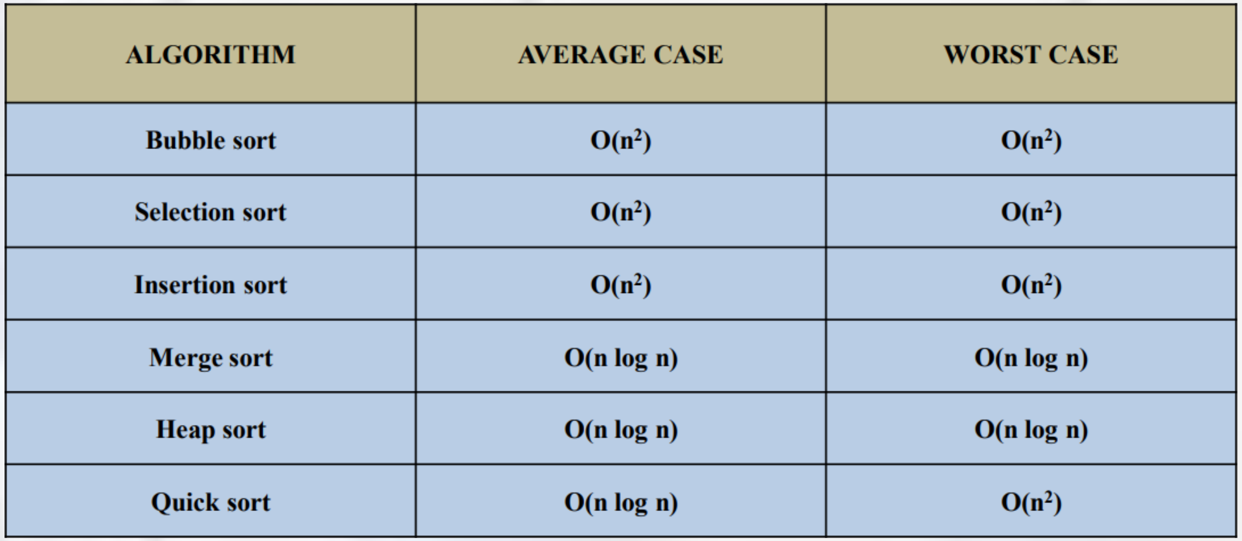
Algorithm to Insert an Element in a Circular Queue

```
Step 1: IF FRONT = 0 and Rear = MAX – 1, then Write “OVERFLOW” Goto Step 4 [END OF IF]
Step 2: IF FRONT = -1 and REAR = -1, then; SET FRONT = REAR = 0 ELSE IF REAR = MAX – 1 and FRONT != 0 SET REAR = 0 ELSE SET REAR = REAR + 1 [END OF IF]
Step 3: SET QUEUE[REAR] = VAL
Step 4: Exit
```

Algorithm to Delete an Element from a Circular Queue

```
Step 1: IF FRONT = -1, then Write “Underflow” Goto Step 4 [END OF IF]
Step 2: SET VAL = QUEUE[FRONT]
Step 3: IF FRONT = REAR SET FRONT = REAR = -1 ELSE IF FRONT = MAX -1 SET FRONT = 0 ELSE SET FRONT = FRONT + 1 [END OF IF] [END OF IF]
Step 4: EXIT
```

```
MERGE_SORT( ARR, BEG, END)
Step 1: IF BEG < END, then
            SET MID = (BEG + END)/2
            CALL MERGE_SORT( ARR, BEG, MID)
            CALL MERGE_SORT (ARR, MID + 1, END)
            MERGE (ARR, BEG, MID, END)
            [END OF IF]
Step 2: END
```



ALGORITHM	AVERAGE CASE	WORST CASE
Bubble sort	$O(n^2)$	$O(n^2)$
Selection sort	$O(n^2)$	$O(n^2)$
Insertion sort	$O(n^2)$	$O(n^2)$
Merge sort	O(n log n)	O(n log n)
Heap sort	O(n log n)	O(n log n)
Quick sort	O(n log n)	$O(n^2)$