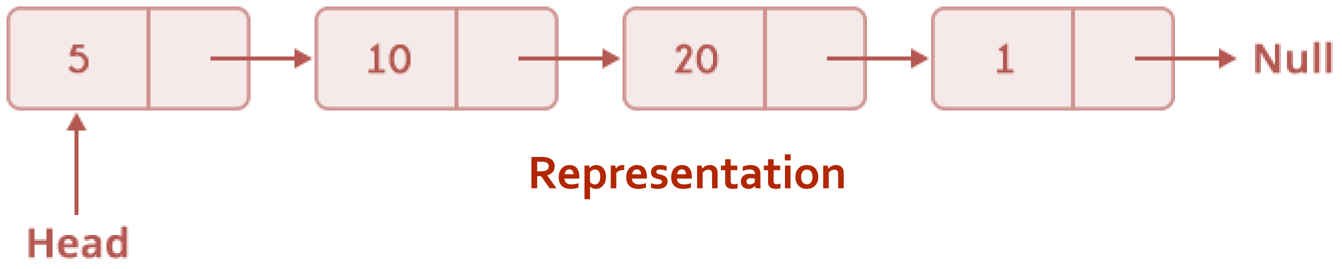
DATA STRUCTURES & ALGORITHMS

Linked List

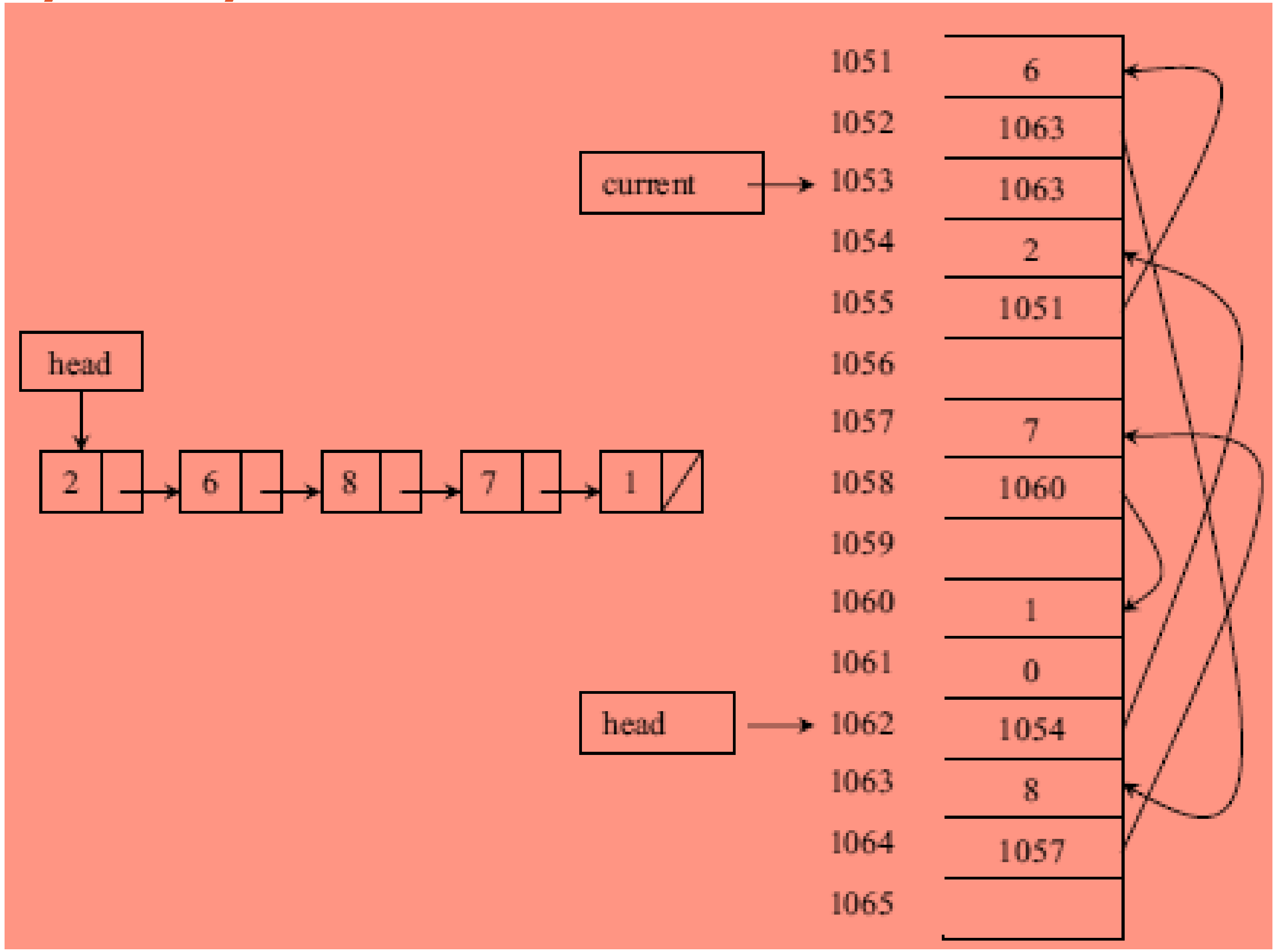
Instructor: Engr. Laraib Siddiqui

Linked list

- It’s a linear data structure, consist of nodes.
- Nodes includes:
  - ✓Data - elements
  - ✓Address/Link - address of another node.



Memory Layout

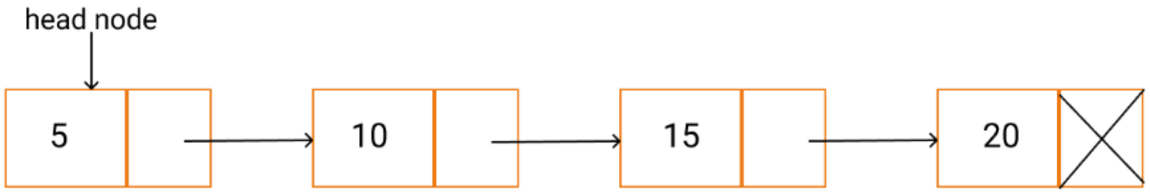


Linked list vs Arrays

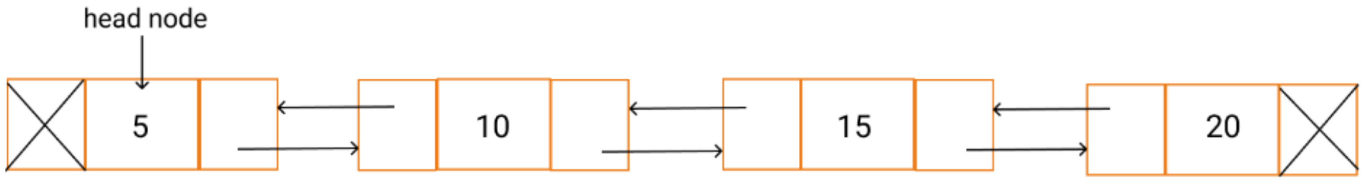
Array	Linked List
Size is fixed; resizing is expensive	Size is dynamic
Data can be access randomly	No random access
Insertion and Deletion operation takes more time.	Insertion and Deletion operations are fast in linked list.
Data is stored in consecutive location.	Data is randomly stored.

Types

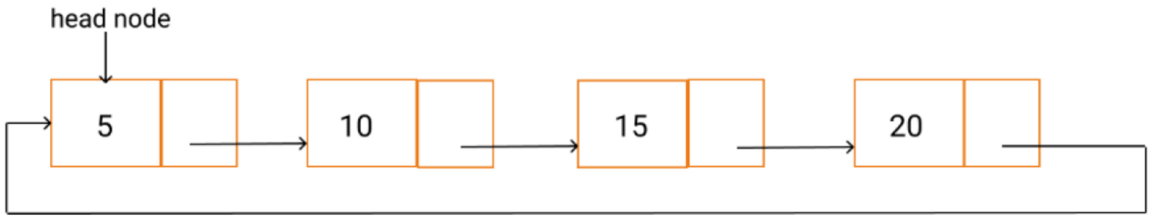
- Singly linked list – Item navigation is forward only.



- Doubly linked list – Items can be navigated forward and backward.



- Circular Linked List – Last item contains link of the first element as next and the first element has a link to the last element as previous.



Comparison

	Singly	Doubly
Strength	Consume less memory Easy to implement	Traversing can be done in both directions
Weakness	Access previous elements not easy	Consume more memory

Traversing

Let LIST be a linked list in memory. Following algorithm traverses LIST, applying an operation PROCESS to each element of LIST. The variable PTR points to the node currently being processed.

1. Set PTR: = START.
2. Repeat Steps 3 and 4 while PTR ≠ NULL.
3. Write: INFO [PTR]
4. Set PTR:= LINK[PTR].

   [End of Step 2 loop.]

5. Exit

Memory Allocation

- A special list call free pool/available space is maintained which consists of unused memory cells.
- This list provides unused memory for the new nodes.
- This also reuse the deleted nodes space.

Overflow and Underflow

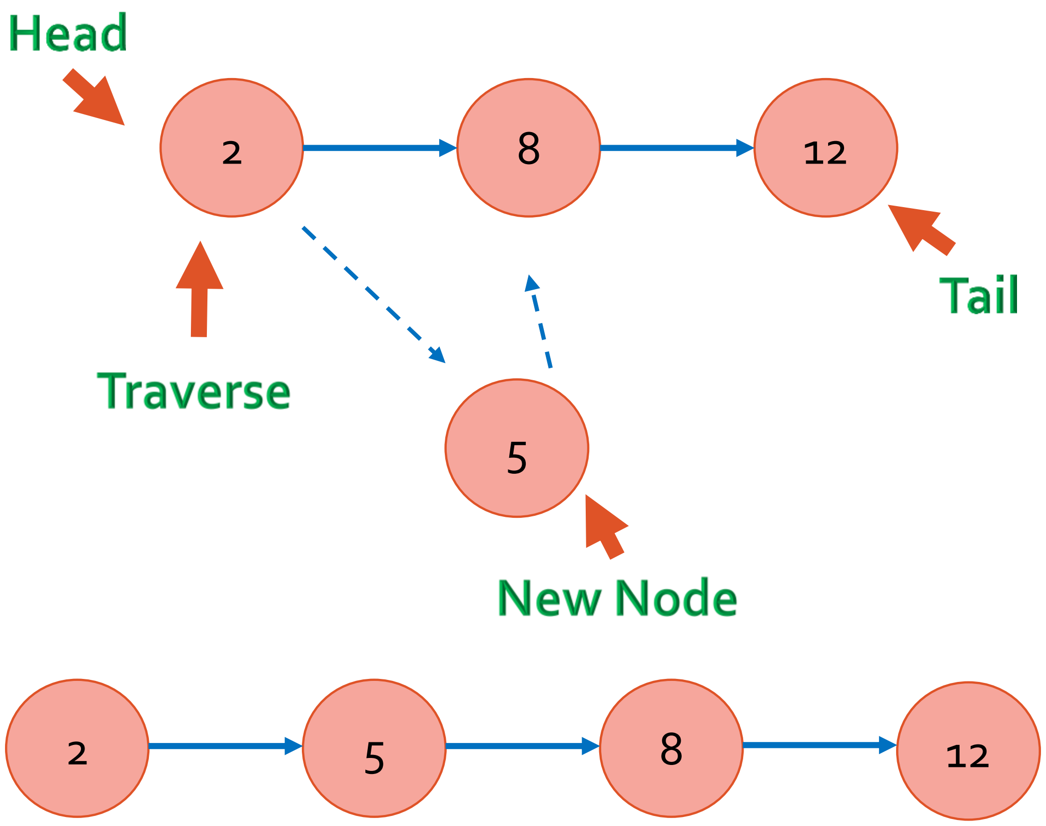
Overflow

- The situation where new data are to be inserted into a data structure but there is no available space.

Underflow

- The situation where one wants to delete data from a data structure that is empty.

Insertion



Insertion

INSLOC (INFO, LINK, START,AVAIL,LOC, ITEM)

This algorithm inserts ITEM so that ITEM follows the node with location LOC or inserts ITEM as the first node when LOC = NULL.

1. [OVERFLOW?] If AVAIL = NULL, then: Write: OVERFLOW, and Exit. — Check for availability
2. Set NEW:= AVAIL and AVAIL:= LINK[AVAIL].
3. Set INFO[NEW]: = ITEM

   Create new node

4. If LOC = NULL, then [Insert as first node.]

   Set LINK[NEW]:= START and START:= NEW.

   Insert first element

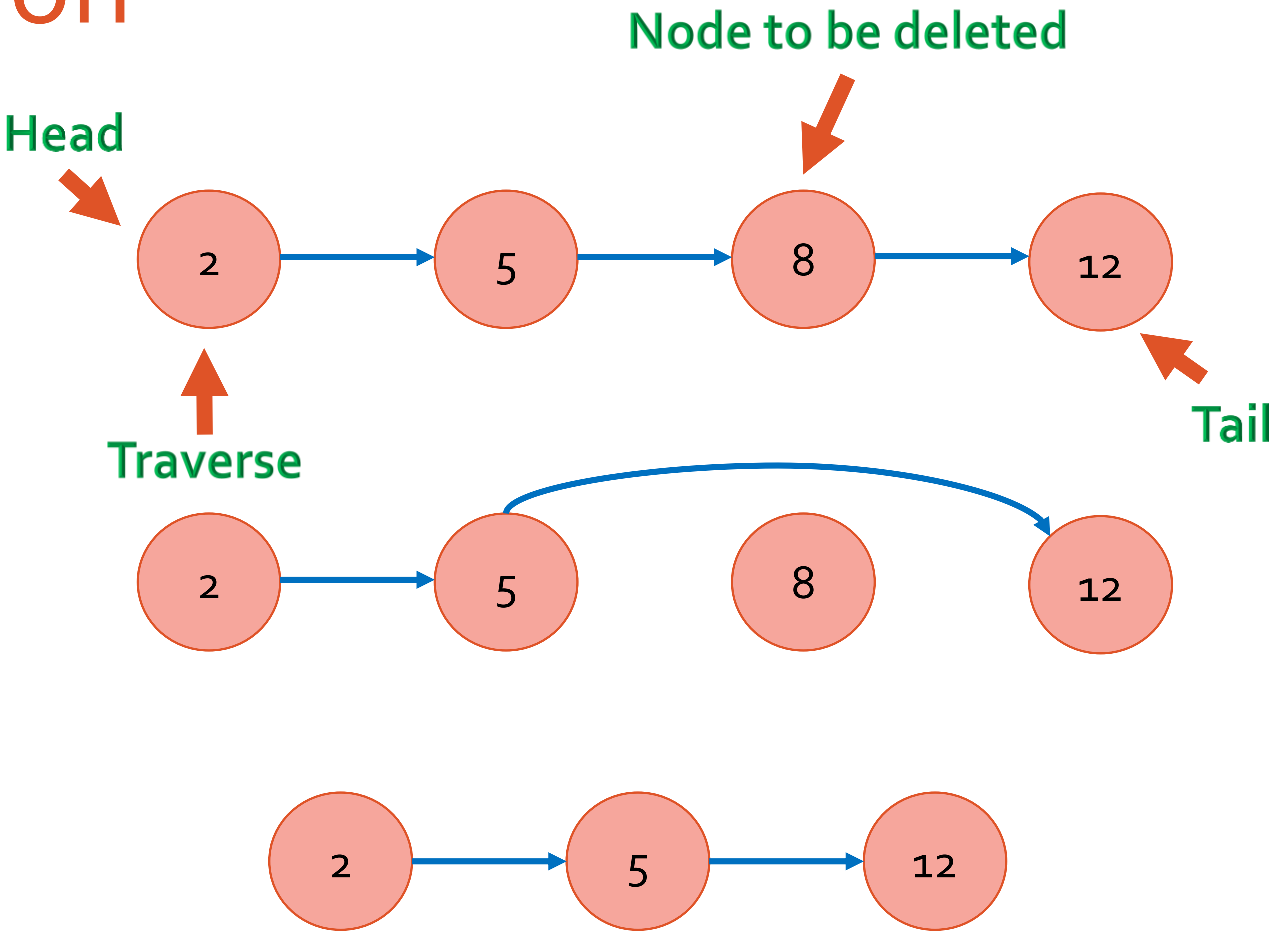
   Else: [Insert after node with location LOC.]

   Set LINK[NEW]:= LINK[LOC] and LINK[LOC]:=NEW.

   Insert after current node

   [End of If structure.]
5. Exit

Deletion



Deletion

DEL(INFO, LINK, START, AVAIL, LOC, LOCP)

This algorithm deletes the node N with location LOC. LOCP is the location of the node which precedes N or, when N is the first node, LOCP = NULL.

1. If LOCP = NULL, then:
   Set START:= LINK[START]. [Deletes first node.]

   Else:
   Set LINK[LOCP]:= LINK[LOC]. [Deletes node N.]

   [End of If Structure.]

2. [Return deleted node to the AVAIL list.]
   Set LINK[LOC]:= AVAIL and AVAIL:= LOC.

3. Exit.

Complexity

	Singly	Doubly
Access	O(n)	O(n)
Search	O(n)	O(n)
Delete	O(1)	O(1)
Insert	O(1)	O(1)

Applications

- Queue & stack implementation.
- Model real world objects.
- Model repeating event cycles.
- Separate chaining.
- Adjacency list for graphs.