Data Structures and Algorithms

Rao Muhammad Umer
Lecturer,
CS and IT Department,
The University of Lahore.
Web: raoumer.github.io

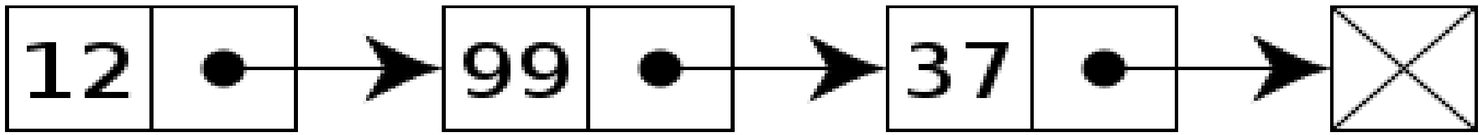
outline

Linked Lists

- What is Linked List?
- Linked List Operations
  - Traversal
  - Insertion
  - Deletion
  - Searching
  - Sorting
  - Merging
  - Reversing
- Circular Linked List
- Doubly Linked List
- Time Complexity Comparison of Linked list
- Applications of Linked list

Linked List

- Linked list also store similar elements in memory
- Linked list elements are not stored at contiguous location.
- The elements are linked using pointers.



Linked List

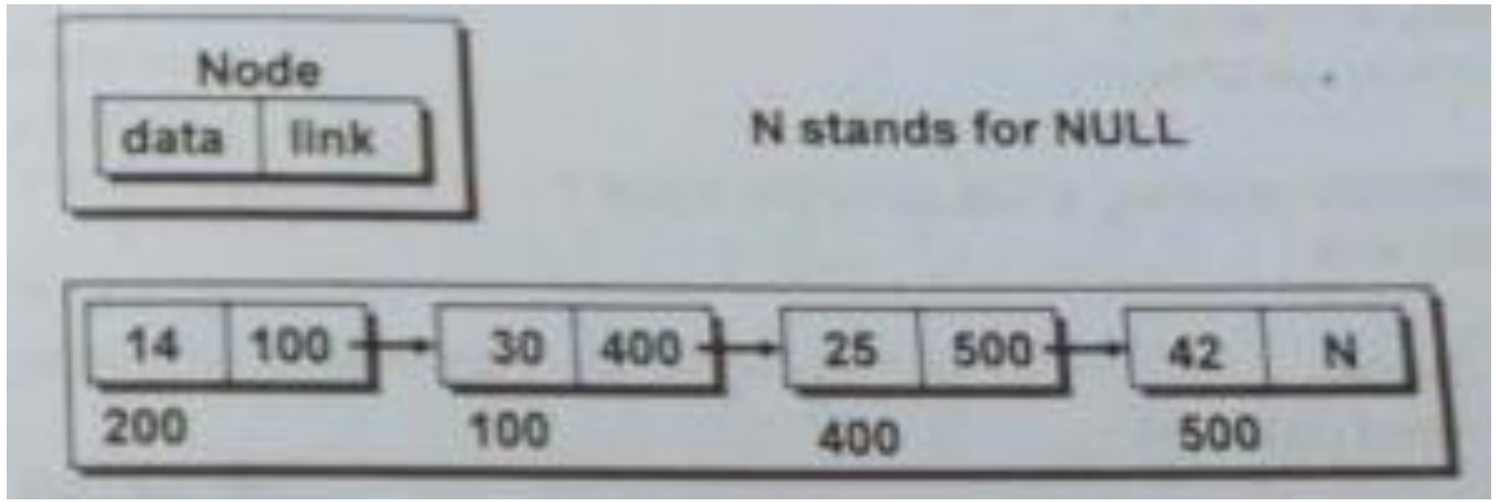
Arrays Limitations:

- Fixed Dimension / size
- Contiguous memory location
- No ease of insertion / deletion operation

Linked List:

- No max size of linked list (Dynamic size)
- No shortage of memory
- Easy insertion and deletion operation

Memory Representation of Linked List



Operations on Linked List

- **Traversal:** Processing each element in the Linked List
- **Search:** Finding the location of an element with a given value
- **Insertion:** Adding a new element to an a Linked List
- **Deletion:** Removing an element from an Linked List
- **Sorting:** Organizing the elements in some order (ascending or descending)
- **Merging:** Combining two Linked List into a single Linked List
- **Reversing:** Reversing the elements of an Linked List

Operations on Linked List

Building a Linked List

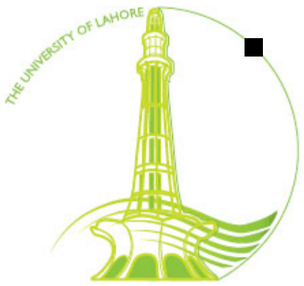
- See Animation of building a Linked List

Addition of a node

- At beginning: See Animation of addition at beginning in Linked List
- At end: See Animation of addition at end in Linked List
- At anywhere: See Animation of addition at anywhere in Linked List
- See source code in C++ of addition operation

Deletion of a node

- See Animation of deleting an element in Linked List
- See source code in C++ of deletion operation



Operations on Linked List

Reversing a Linked List

- See Animation of reversing a Linked List
- See source code in C++ of reverse operation

Merging two Linked List

- See Animation of merging two Linked List
- See source code in C++ of merging operation

Ascending order Linked List

- See Animation of merging two Linked List
- See source code in C++ of merging operation

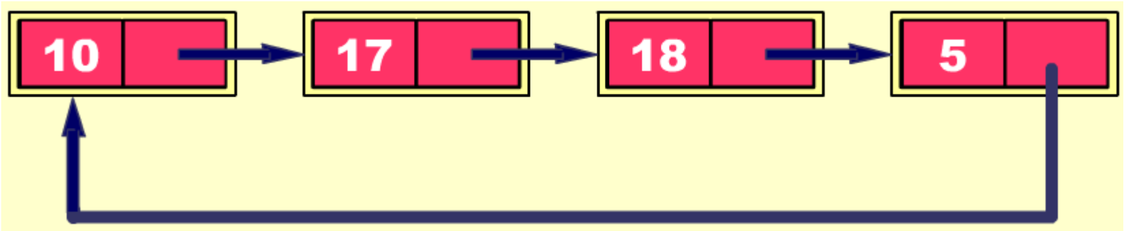
Operations on Linked List

Sorting a Linked List

- By changing data: See Animation of reversing a Linked List
- By changing links: See Animation of reversing a Linked List
- See source code in C++ of sort operation

Circular Linked List

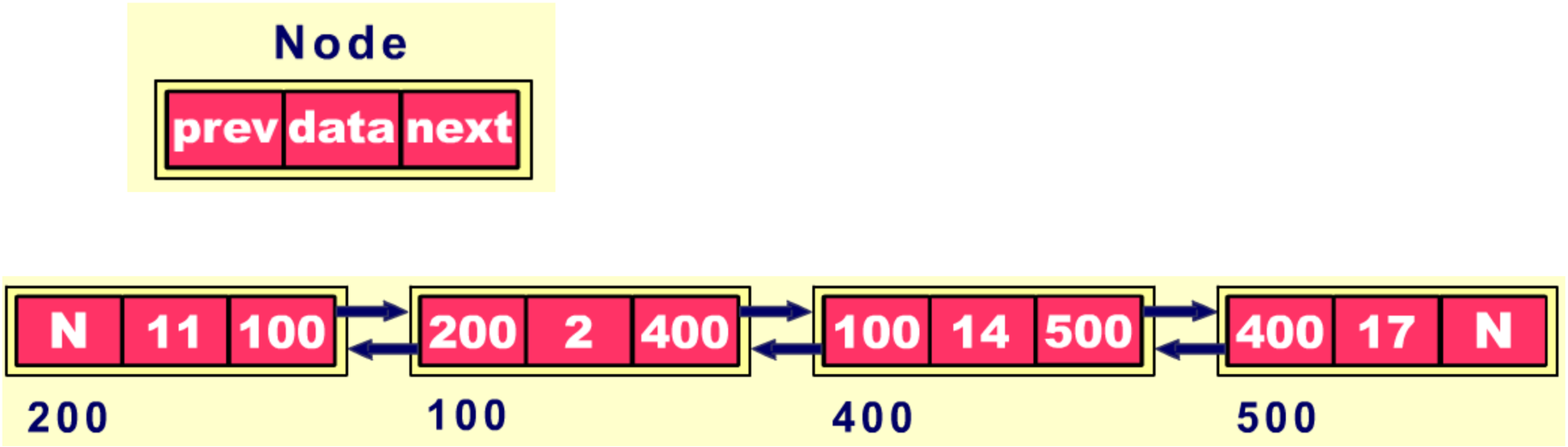
- **Circular Linked List**
  - Change the structure of linear list such that **link** field of last node contains a pointer back to the first node rather than **NULL**.
  - Does not have a **first** or **last** node
  - Can be used to represent as a **stack** and a **queue**



- **Operations on Circular Linked List as a Queue**
  - Addition in empty list: See Animation of reversing a Linked List
  - Addition in non-empty list: See Animation of reversing a Linked List
  - Deletion of a node: See Animation of reversing a Linked List
  - See source code in C++ of sort operation

Doubly Linked List

Doubly Linked List



Doubly Linked List

Doubly Linked List

- Building a Doubly Linked List:
  - See Animation of building a doubly linked List
  - See source code in C++ of building a doubly linked list operation
- Addition in Doubly Linked List:
  - At beginning: See Animation of addition at beginning in doubly Linked List
  - At end: See Animation of addition at end in doubly Linked List
  - At anywhere: See Animation of addition at anywhere in doubly Linked List
  - See source code in C++ of deletion operation
- Deletion of a node from Doubly Linked List:
  - From beginning: See Animation of deletion from beginning in doubly Linked List
  - From end: See Animation of deletion from end in doubly Linked List
  - From anywhere: See Animation of deletion from anywhere in doubly Linked List
  - See source code in C++ of deletion operation

Time Complexity comparison of list data structures

- **Linked Lists vs. Dynamic Arrays:**

	Linked list	Array	Dynamic array	Balanced tree	Random access list	hashed array tree
Indexing	Θ(*n*)	Θ(1)	Θ(1)	Θ(log n)	Θ(log n)[4]	Θ(1)
Insert/delete at beginning	Θ(1)	N/A	Θ(*n*)	Θ(log n)	Θ(1)	Θ(*n*)
Insert/delete at end	Θ(1) when last element is known; Θ(*n*) when last element is unknown	N/A	Θ(1) amortized	Θ(log *n*)	Θ(log *n*) updating	Θ(1) amortized
Insert/delete in middle	search time + Θ(1)	N/A	Θ(*n*)	Θ(log *n*)	Θ(log *n*) updating	Θ(*n*)
Wasted space (average)	Θ(*n*)	0	Θ(*n*)[8]	Θ(*n*)	Θ(*n*)	$\Theta(\sqrt{n})$

https://en.wikipedia.org/wiki/Linked_list

Applications of Linked List Data Structure

- **Linked List's Applications:**
  - **FAT:** See Animation of FAT(File Allocation Table) using linked list
  - Suppose you need to program an application that has a pre-defined number of categories, but the exact items in each category is unknown.
    - What Data Structure should we use? Arrays or Linked List?
  - Linked list is used in a lot of data structures, e.g. Stacks , Queues , Dequeues , Graphs, Hash tables ,etc.
  - Circular linked list is commonly used for scheduling processes in operating system.

Acknowledgement

- Mostly Slides taken from Book: **“Data Structures through C++”** by Yashavant P. Kanetkar
- https://en.wikipedia.org/wiki/Linked_list