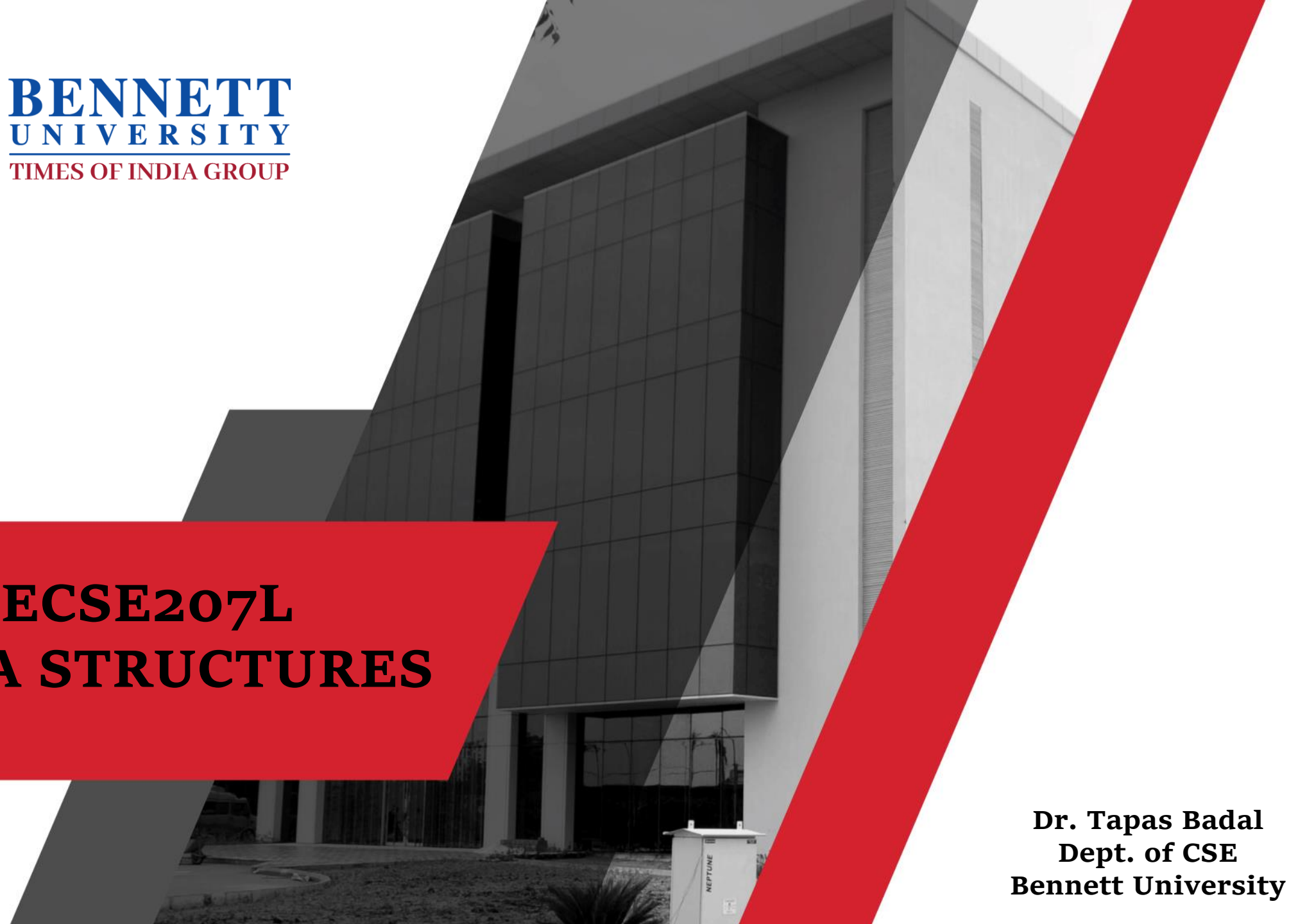
BENNETT UNIVERSITY
TIMES OF INDIA GROUP




# ECSE207L
# DATA STRUCTURES

**Dr. Tapas Badal**
**Dept. of CSE**
**Bennett University**

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

# LINKED LISTS

# Issues with Arrays



- Array size has to be fixed in the beginning.

- Sorted arrays: O(log n) time to search
- Inserting a new element
  - create a hole – shift large number of elements
  - large number of operations, worst case: O(nm)
- Deleting a new element
  - close a hole – shift large number of elements
  - large number of operations, worst case: O(nm)

# Handling Lists



- Consider a list of integers
- { 16, 8, 10, 2, 34, 20, 12, 32, 18, 9, 3 }
- It can be thought of as an element 16 followed by another list
- 16 --{ 8, 10, 2, 34, 20, 12, 32, 18, 9, 3 }

- 16 --{some list}

# Handling Lists



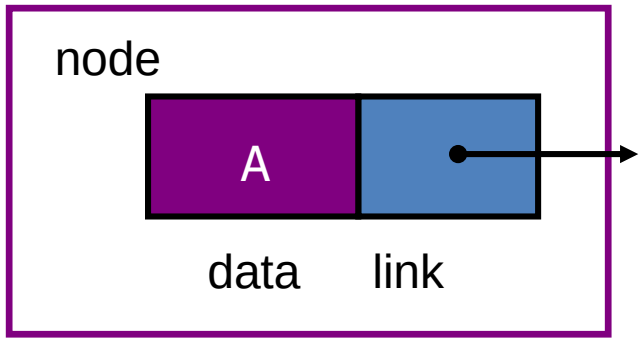
- The second list can also be thought of an element 8 followed by a list
- 16--8-- { 10, 2, 34, 20, 12, 32, 18, 9, 3 }
- 16--8--{another list}

- Thus any list can be thought of as an element followed by a list
- We can come up with a recursive definition of a linked list where each element is linked to the next one

# Linked Lists



- A linked list is a linear data structure where each element is a separate object.
- Each element of such a list needs to comprise of two items
  - The data and
  - A reference to the next node.
- **Node:** object that holds the data and refers to the next element.
- **Data:** component may actually consist of several fields.
- For example in a linked list of students, the data could be student's name and CGPA.

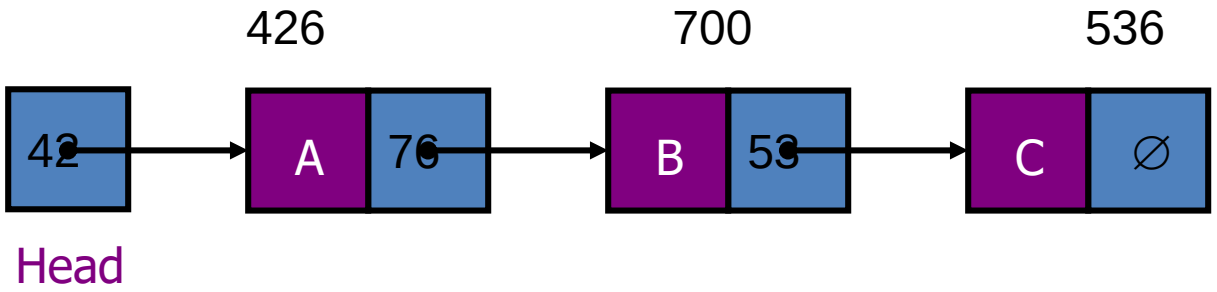


# List stored as an array A[ ]

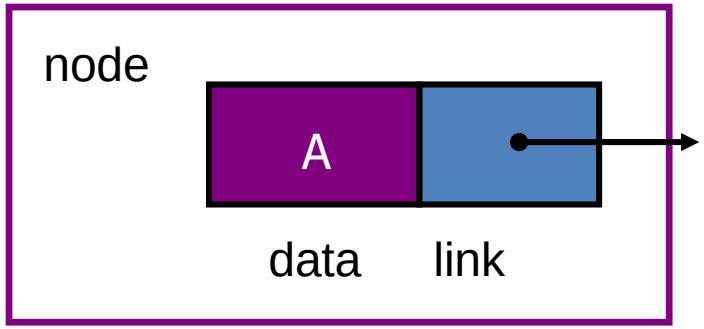


| Memory Address | Array Index | List Contents |
|---|---|---|
| 3200 | A[0] | 36 |
| 3202 | A[1] | 42 |
| 3204 | A[2] | 20 |
| 3206 | A[3] | 16 |
| 3208 | A[4] | 38 |
| 3210 | A[5] | 40 |
| 3212 | A[6] | 12 |
| 3214 | A[7] | 54 |
| 3216 | A[8] | 82 |

# Implementing lists using Linked Lists




- A *linked list* is a series of connected *nodes*
- Each node contains at least
  - A piece of data (any type)
  - Link to the next node in the list
- *Head*: points to the first node
- Links generated by system
- The last node points to NULL



# linked list is an computer memory

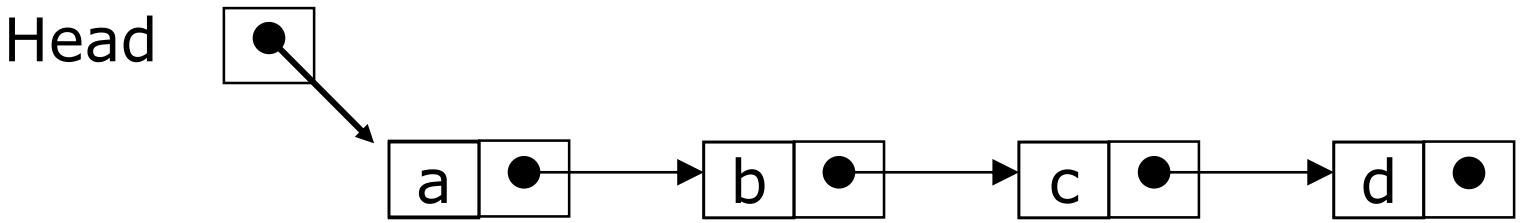
BENNETT UNIVERSITY
TIMES OF INDIA GROUP

| Memory Address | List contents | Next link |
|---|---|---|
| 2020 | 36 | 450 |
| 450 | 42 | 3600 |
| 3600 | 20 | 4200 |
| 4200 | 16 | 4231 |
| 4231 | 38 | 760 |
| 760 | 40 | 5555 |
| 5555 | 12 | X |
| | | |
| | | |

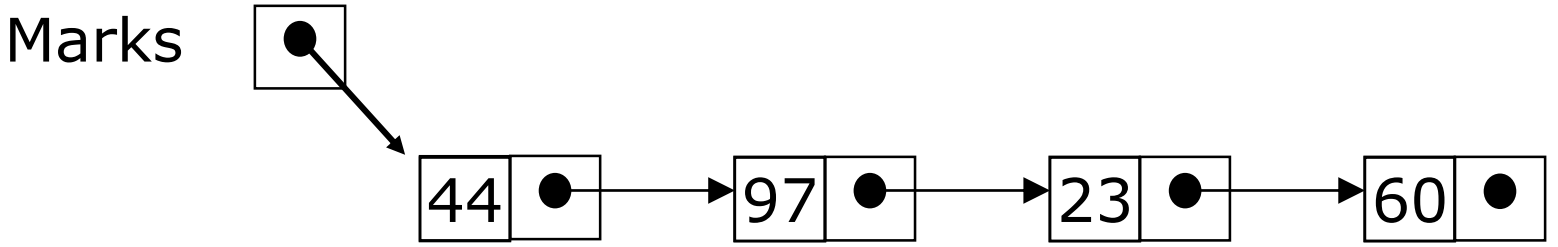
# Anatomy of a linked list

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

- A linked list consists of:
  - A sequence of nodes




- Each node contains data (a value) and a link (pointer) to the following node The last node contains a null link The list is referred to by pointer to first node, called the head/front


Marks
44
97
23
60

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

# List ADT: operations that we want to do on a list

# List ADT : popular choices

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

| | |
|---|---|
| create_list | { 16, 8, 10, 2, 34, 20, 12} |
| Insert_front | { 55, 16, 8, 10, 2, 34, 20, 12} |
| Insert_rear | { 16, 8, 10, 2, 34, 20, 12, 55} |
| Delete_front | { 8, 10, 2, 34, 20, 12} |
| Delete_rear | { 16, 8, 10, 2, 34, 20 } |
| Delete_kth | { 16, 8, 10, 2, 20, 12} |
| Print_list | 16-8-10-2-34-20-12 |
| Count_list | 7 |
| check_if_empty | No |
| Clear_list | { } |

BENNETT UNIVERSITY
TIMES OF INDIA GROUP

# THANKYOU

@csebennett

cse_bennett