IBA

IBA FCS
Faculty of Computer Science

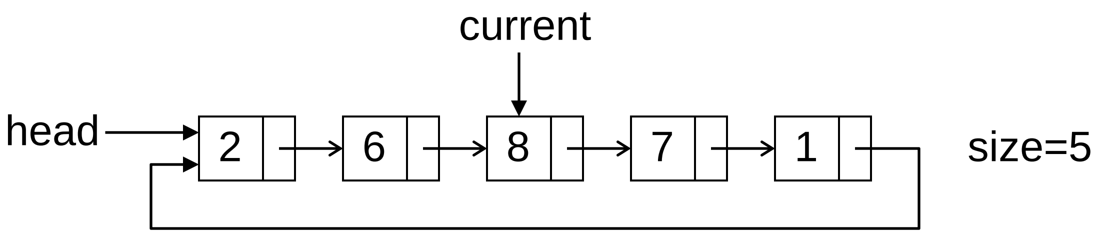
CSE 247
Data Structures

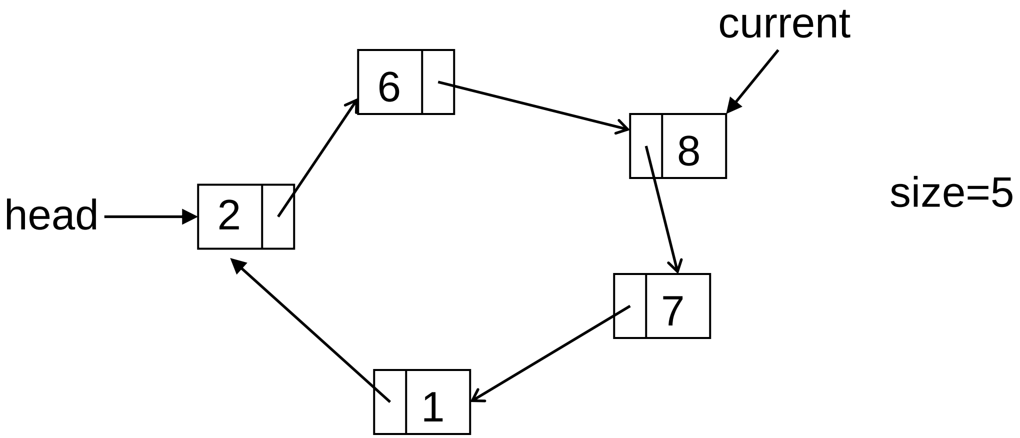
Circular Linked list

- A Circular Linked List is a special type of Linked List
- It supports traversing from the end of the list to the beginning by making the last node point back to the head of the list

Circular Linked List

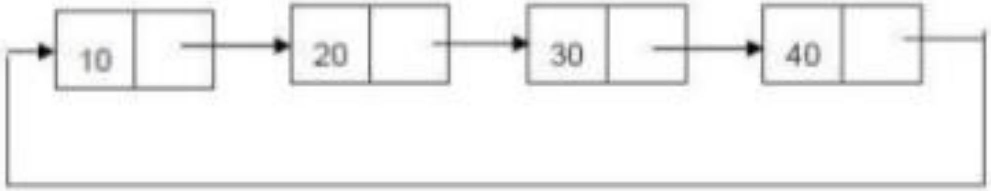
- Two views of a circularly linked list:



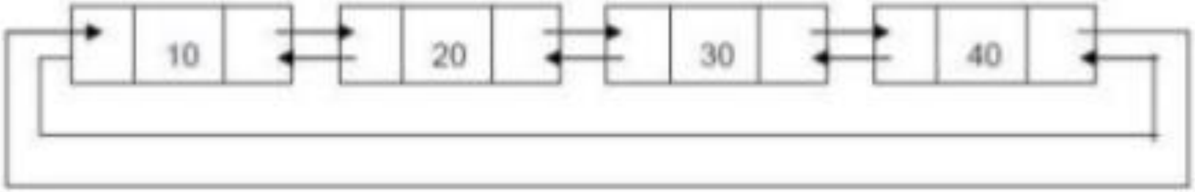


Types of circular linked list:

1. **Singly**: The last node points to the first node and there is only link between the nodes of linked list.



2. **Doubly**: The last node points to the first node and there are two links between the nodes of linked list.



Issues with circular list

- How do you know when you’re done?
  - Make sure you save the head reference.
  - When (cur.next == head) you’ve reached the end
- How circular link list is an efficient data organization?
  - Only last pointer can access both last node and head node.
- How are insertion and deletion handled?
  - No special cases! (e.g first, middle , last). Just to maintain head node.
  - Predecessor to head node is the last node in the list.

Application

- A simple example is keeping track of whose turn it is in a multi-player board game. Put all the players in a circular linked list. After a player takes his turn, advance to the next player in the list. This will cause the program to cycle indefinitely among the players.
- Music play list
- A circular linked list can be effectively used to create a queue (FIFO) or a deque (efficient insert and remove from front and back) in a timesharing problem solved by the operating system.

Application

- Some problem domains are inherently circular (means rotating in a circle).
  - For example :
    - escalator, bus stops, railway station, airport , the squares on a Monopoly board can be represented in a circularly linked list.

- Some solutions can be mapped to a circularly linked list for efficiency.
  - For example:
    - a jitter buffer is a type of buffer that takes numbered packets from a network and places them in order, so that (for example) a video or audio player can play them in order.This can be represented in a circular buffer, without needing to constantly allocate and deallocate memory, as slots can be re-used once they have been played.

Array vs. Linked List

Following are the points in favor of Linked Lists.

1. The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage, and in practical uses, upper limit is rarely reached.
2. Inserting a new element in an array of elements is expensive, because room has to be created for the new elements and to create room existing elements have to shifted.
3. Deletion is also expensive with arrays until unless some special techniques are used and to remove cell in an array existing elements have to shifted.

So Linked list provides following two advantages over arrays

1. Dynamic size
2. Ease of insertion/deletion

Array vs. Linked List

Linked lists have following drawbacks:

1. Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do binary search with linked lists.
2. Extra memory space for a pointer is required with each element of the list.
3. Arrays have better cache locality that can make a pretty big difference in performance.

The Josephus Problem

The **Josephus Problem** (or **Josephus permutation**) is a theoretical problem related to a certain counting-out game.

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

- (One of many variations…)

  The founder of a startup is forced to lay off all but one employee. Not having any better way to decide, he arranges all employees in a circle and has them count off. The 10th employee in the circle is laid off, and the count begins again. The last person is not laid off.
- If there are N employees, where should you sit to avoid being laid off?

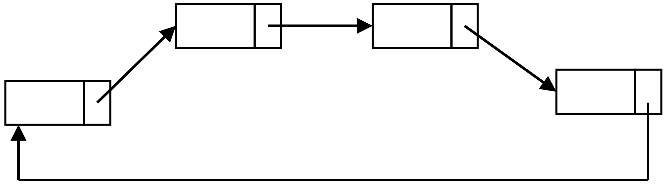
https://en.wikipedia.org/wiki/Josephus_problem

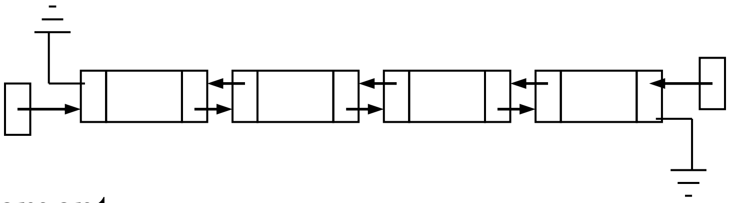
Solution

- Model the circle of employees as a circular linked list
- Implement the counting off process, and delete the 10th employee each time (or N could be a random number)
- After N-1 people are deleted, there should be only one employee left.
- That employee's original position number is the solution to the problem.

Summary of linked lists

- Types of linked lists:
  - Singly linked list
    - Begins with a pointer to the first node
    - Terminates with a null pointer
    - Only traversed in one direction
  - Circular, singly linked
    - Pointer in the last node points back to the first node





  - Doubly linked list
    - Two "start pointers" – first element and last element
    - Each node has a forward pointer and a backward pointer
    - Allows traversals both forwards and backwards
  - Circular, doubly linked list
    - Forward pointer of the last node points to the first node and backward pointer of the first node points to the last node