Data Structures

Header Linked List-2 Way Linked List

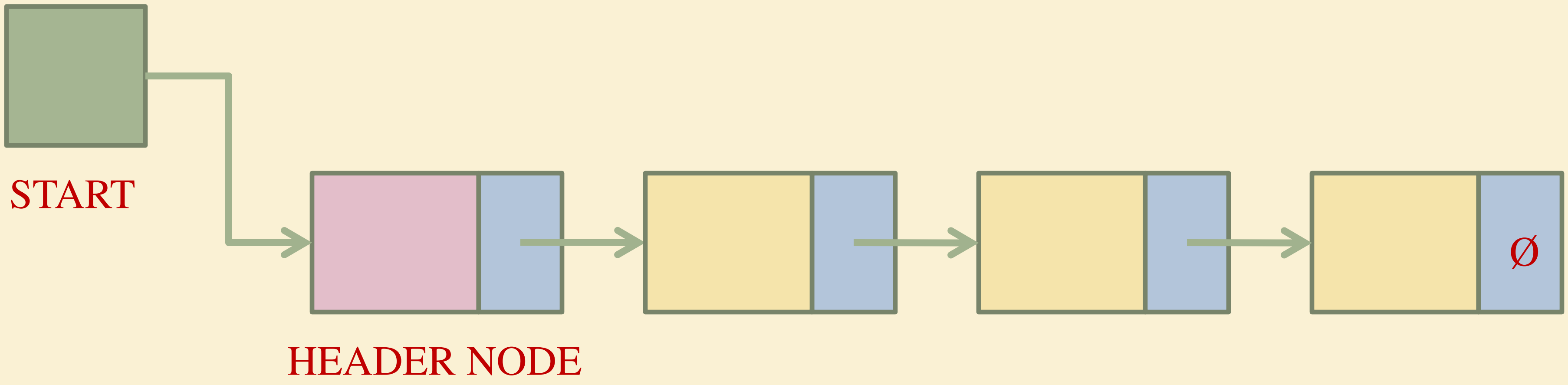
Lovely Professional University, Punjab

Outlines

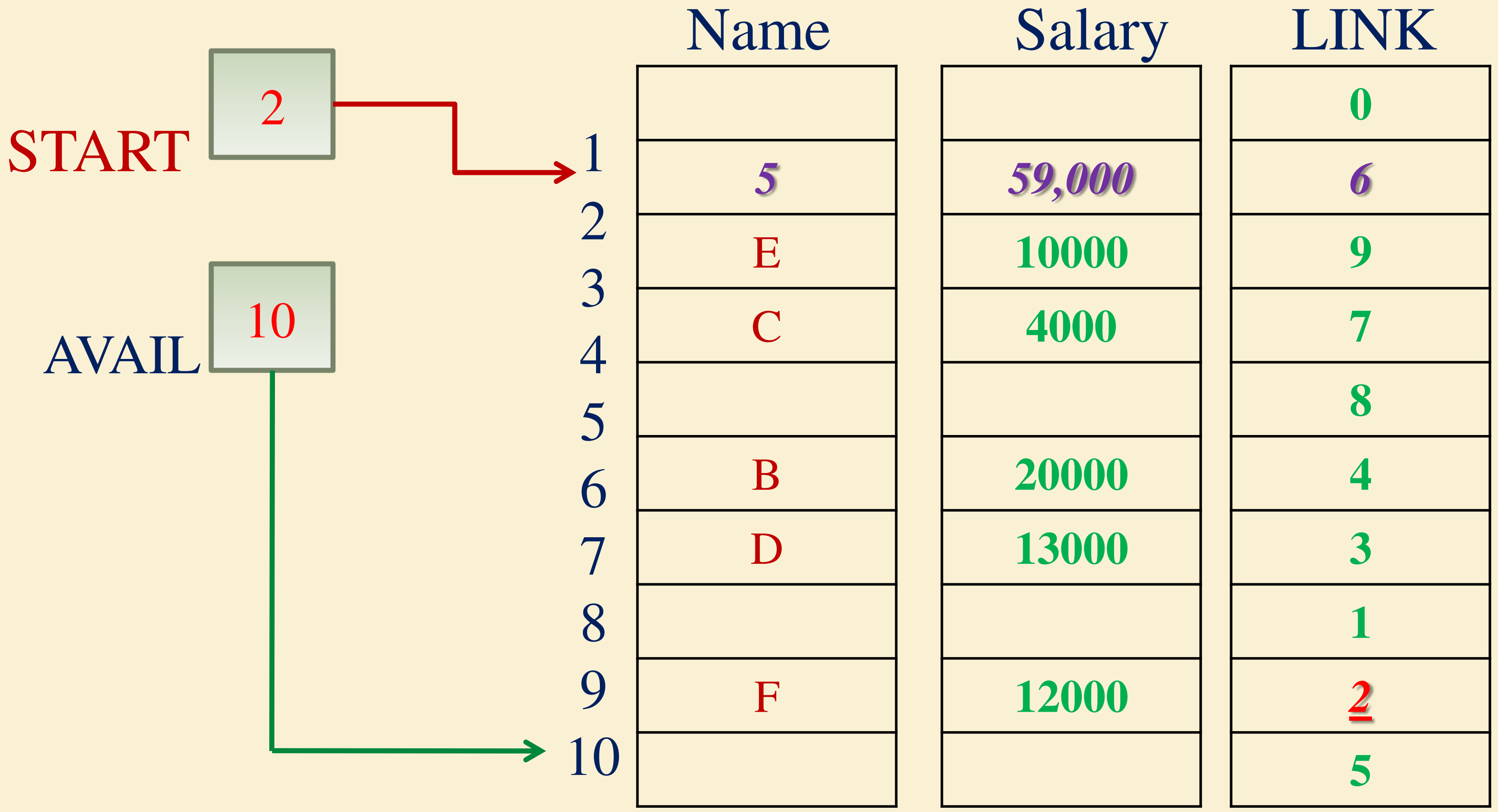
- Introduction
- Header Linked List
- Advantages of Header Linked List
- Types of Header Linked List
- Review Questions

Header Linked List

- A header linked list which always contains a special node, called the header node, at the beginning of the list.



Header Linked List



	Name	Salary	LINK
1			0
2	*5*	*59,000*	*6*
3	E	10000	9
4	C	4000	7
5			8
6	B	20000	4
7	D	13000	3
8			1
9	F	12000	2
10			5

Advantages of Header Linked List

- Header linked list contains a special node at the top.
- This header node need not represent the same type of data that succeeding nodes do.
- It can have data like, number of nodes, any other data...
- Header node can access the data of all the nodes in the linked list.
- The header node is never deleted in the list and always point to the first actual node in the list.

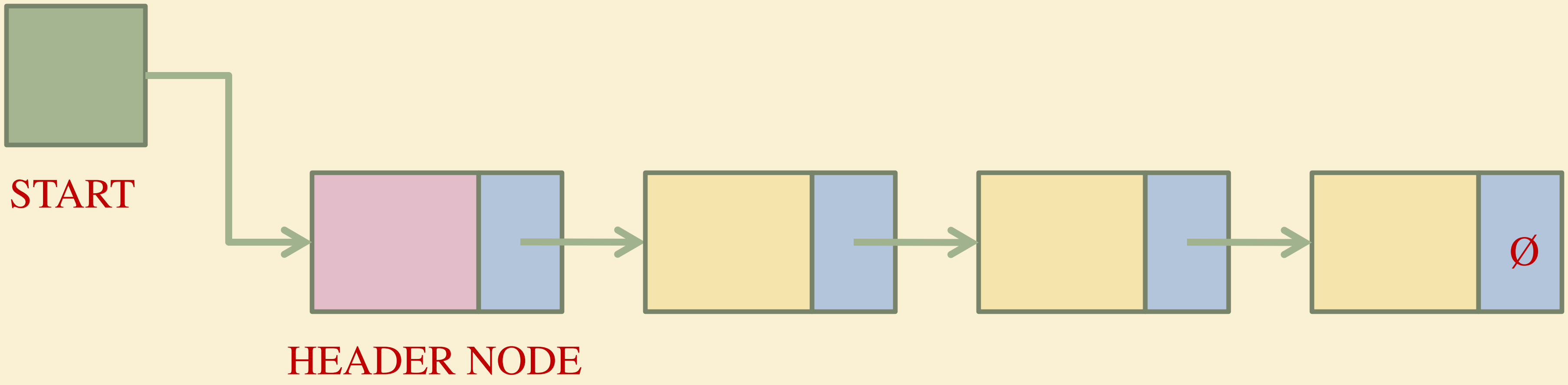
Types of Header Linked List

- A **Grounded header list** is a header list where the last node contains the null pointer.

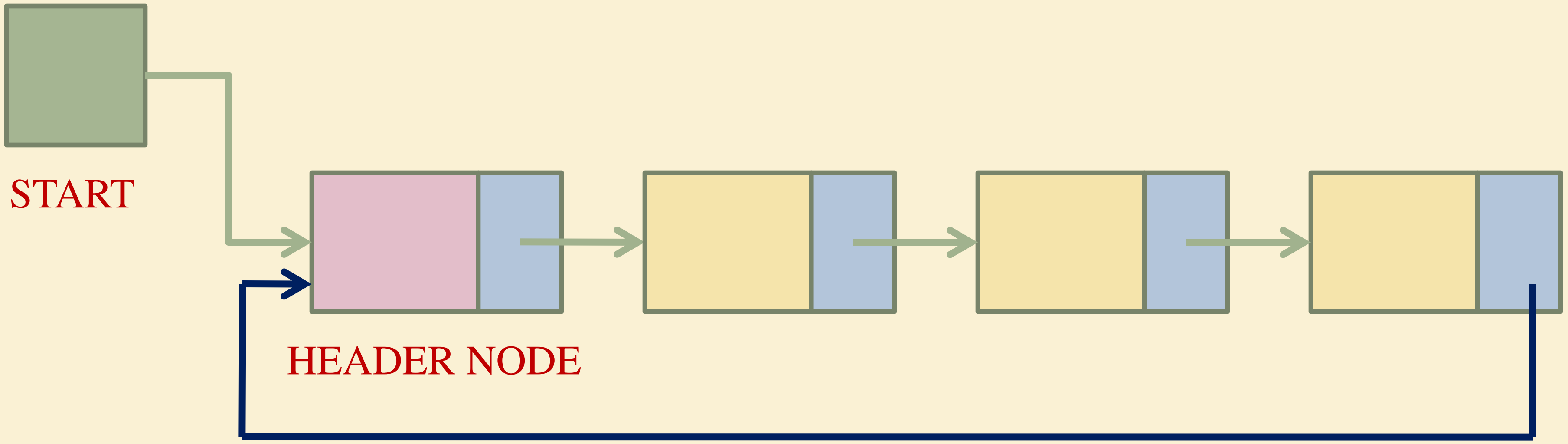
- A **Circular header list** is a header list where the last node points back to the header node.

Note:

- Unless otherwise stated or implied, header list will always be circular list.
- Accordingly, in such a case, the header node also acts as a sentinel indicating the end of the list.



Grounded Header List



Circular Header List

Circular Linked List

- The main drawback of singly linked list is that from a given node, we can access all the nodes that follow it but not the one preceding it.
- To overcome this we make slight modification in the single linked list by replacing the null pointer of the last node of list with the address of its first node.
- This is called the **Circular Linked List**

- The operations performed on a circular linked list are similar to those of a single linked list except that we need to remember that the last node does not contain null pointer but points to the first node.
- Eg: for deleting the first node of a circular linked list we need to change the link part of the last node and for that we need to traverse the list to reach the last node.
- This problem can be solved using the TAIL pointer. That points to the last node of the list.
- Now we can directly access the last node and first node

underflow Condition

- If Link [START] = NULL,
  then, Grounded Header List is Empty.

- If Link [START] = START,
  then, Circular Header List is Empty.

Characterstics of Header Linked List

- The main advantage of using header node in the linked list is that we can avoid special testing involved while inserting and deleting nodes from the linked list.
- As we don't need to check the whether the list is empty or not because header node give us that information.
- To perform the basic operations on a header linked list, the algorithm that we have discussed previously need to be re-written so as to account for presence of header node. As for traversing ,in order to point to first node

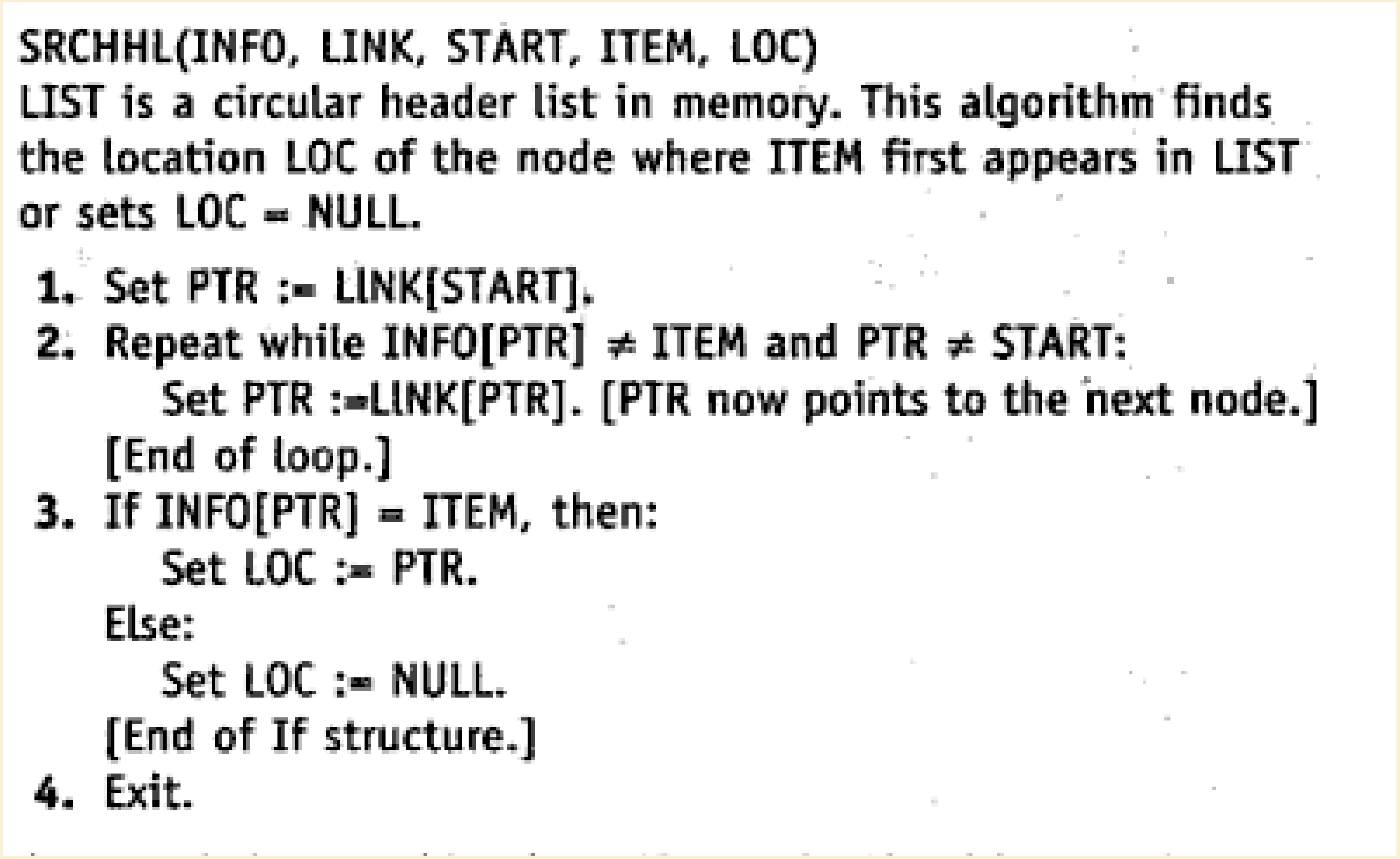
  PTR:=LINK [START]

Traversing a Circular Header List

- Algorithm (Traversing a Circular Header list)

```
1.  Set PTR = LINK [START]. [Initialize pointer PTR]
2.  Repeat step 3 and 4 while PTR ≠ START.
3.          Apply PROCESS to INFO[PTR].
4.          Set PTR = LINK [PTR].   [PTR points to next node]
    [End of Step 2 Loop.]
5.  EXIT
```

Searching in circular Linked List



SRCHHL(INFO, LINK, START, ITEM, LOC)
LIST is a circular header list in memory. This algorithm finds the location LOC of the node where ITEM first appears in LIST or sets LOC = NULL.

```
1. Set PTR := LINK[START].
2. Repeat while INFO[PTR] ≠ ITEM and PTR ≠ START:
       Set PTR :=LINK[PTR]. [PTR now points to the next node.]
   [End of loop.]
3. If INFO[PTR] = ITEM, then:
       Set LOC := PTR.
   Else:
       Set LOC := NULL.
   [End of If structure.]
4. Exit.
```

Deletion in circular Linked List

```
1.PTR:=LINK[START]   [Set PTR to first node]
2.LOCP:=START     [Set PREVLOC to header node]
3. Repeat while PTR!=START and ITEM!=INFO[PTR]
4. LOCP:=PTR  and PTR:=LINK[PTR]          [update pointers]
[End of step 3 loop]
4.if ITEM=INFO[PTR]
            LOC:=PTR                             [update LOC,now points to target node]
    Else:
            Write:Item not found
    [end of if structure]
5. LINK([LOCP]):=LINK[LOC]                       [Delete node]
6. LINK[LOC]:=AVAIL                               [Return node to free storage list]
7.AVAIL:=LOC
8.Exit.
```

Use of Header Linked List

- Header Linked lists are frequently used for maintaining *Polynomials* in memory.

Review Questions

- What is Header Node?
- How a Linked List is different from Header linked list?
- What is Grounded Header list and circular header list?

Two-Way List/Doubly Linked List

Outlines

- Introduction
- Two-Way List
- Two-Way Header List
- Operations on Two-Way List
  - Traversing
  - Searching
  - Deleting
  - Inserting
- Review Questions

Introduction

- If a list can be traversed in only one direction, it is called One-way List.
- List Pointer variable START points to the first node or header node.
- Next-pointer field LINK is used to point to the next node in the list.
- Only next node can be accessed.
- Don’t have access to the preceding node.

OPERATIONS ON DOUBLY LINKED LIST

Traversal

- Doubly linked list can be traversed both ways and the traversing algorithm is same as the traversing the linked list,we start using START pointer and follow the chain of link using the pointers FORW in succession
- Similary in backward traversal we traverse in the backward direction using END pointer and follow the chain of BACK pointers in succession

Searching

- It involves the searching in of the LOC ,and usually we search using the forward progression only same as the way when searching in the single linked list.
- But if our list is sorted and we are searching for the ITEM which may be at the end of the list then we may use the TAIL pointer for ease of access.

Two-Way List

- A Two-Way list is a linear collection of data elements, called nodes, where each node N is divided into three parts:

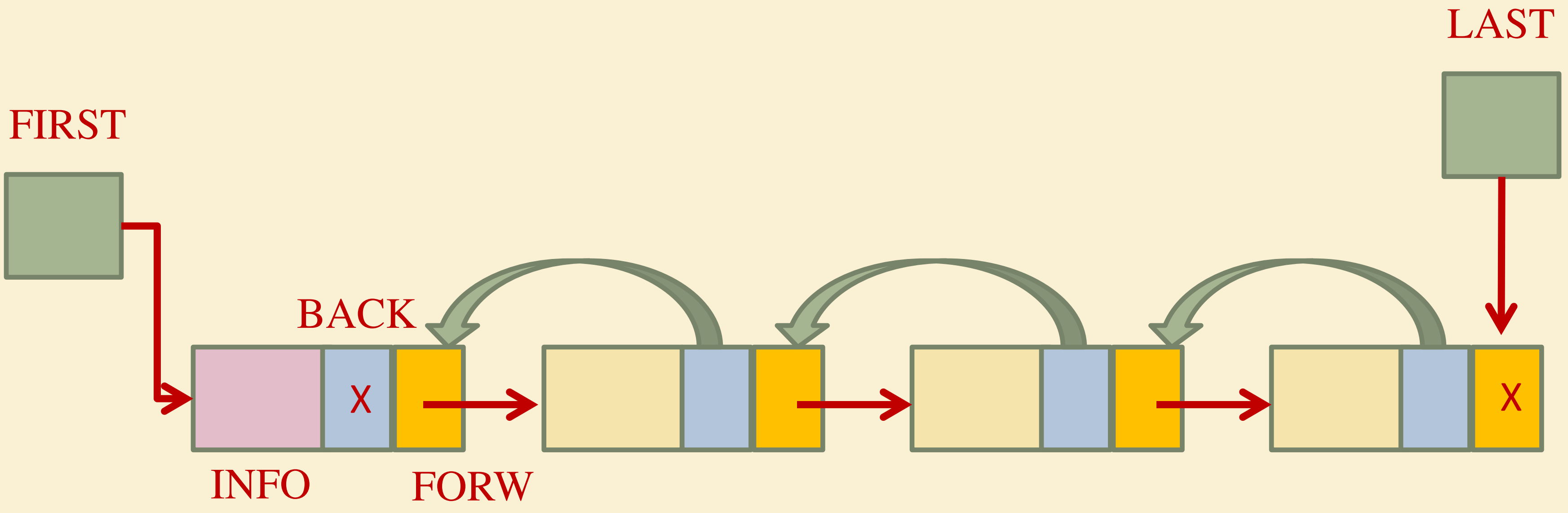
1. An information field *INFO* which contains the data of N.

2. A pointer field *FORW* which contains the location of the next node in the list.

3. A pointer field *BACK* which contains the location of the preceding node in the list.

Two-Way List …

- The Two-Way list requires two list pointer variables:
  1. FIRST: points to the first node in the list.
  2. LAST: points to the last node in the list.



Key Points

- If LOCA and LOCB are the locations of node A and node B respectively, then

FORW [LOCA] = LOCB, iff

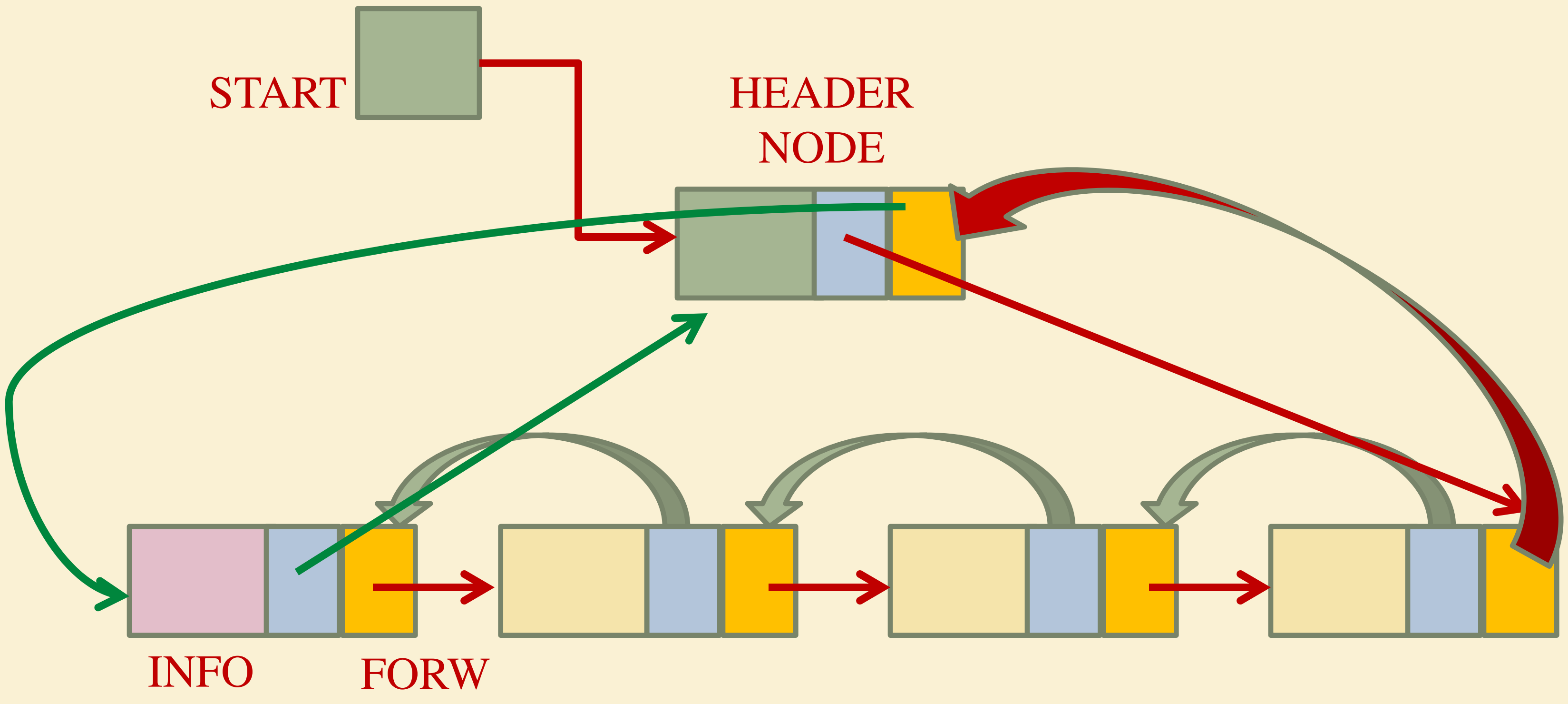
BACK [LOCB] = LOCA

- Two way lists are maintained in memory by means of linear arrays as in one way lists.

- But now we require two pointer arrays BACK and FORW.

Two-Way Header List

- It is circular because the two end nodes point back to the header node.
- Only one list pointer variable START is required.



Insertion in a Two-way List

- INST_TWL (INFO, FORW, BACK, START, AVAIL, LOCA, LOCB, ITEM)

```
1. [OVERFLOW?] If AVAIL = NULL, then: Write: Overflow and Exit.

2. [Remove Node from AVAil List and copy the Item into it.]
    Set NEW = AVAIL, AVAIL = FORW [AVAIL],
        INFO [NEW] = ITEM.

3. [Insert Node into the list.]
    Set FORW [LOCA] = NEW, FORW [NEW] = LOCB,
        BACK [LOCB] = NEW, and BACK [NEW] = LOCA.

4. Exit.
```

Deletion in a Two-way List

- DEL_TWL (INFO, FORW, BACK, START, AVAIL, LOC)

```
1. [Delete Node.]
   Set FORW [BACK [LOC] ] = FORW [LOC] and
   BACK [FORW [LOC] ] = BACK [LOC].

2. [Return Node to AVAIL list.]
   Set FORW [LOC] = AVAIL and AVAIL = LOC.

3. Exit.
```

Review Questions

- What is the advantage of Two-way List over Linked List?
- How Traversing is done in TWL?
- What is the difference between Deletion in Linked List and TWL?
- How TWL is more efficient than Linked List in case of Searching?