Lecture 8
Linked List ,
Traversing Operation of Linked List

Prepared By:

Afsana Begum
Lecturer,
Software Engineering Department,
Daffodil International University.

Objectives

- Introduction with link list
- Characteristics of link list
- Garbage collection
- Over flow and Under flow
- Doubly linked lists
- Basic operations of Link list
- Traversing of Link List

Linked lists

•A linked list, or one way list, is a linear collection of data elements, called nodes, where the linear order is given by means of pointers.

•Dynamically allocate space for each element as needed.

Node

In linked list

Each node of the list contains the data item

a pointer to the next node

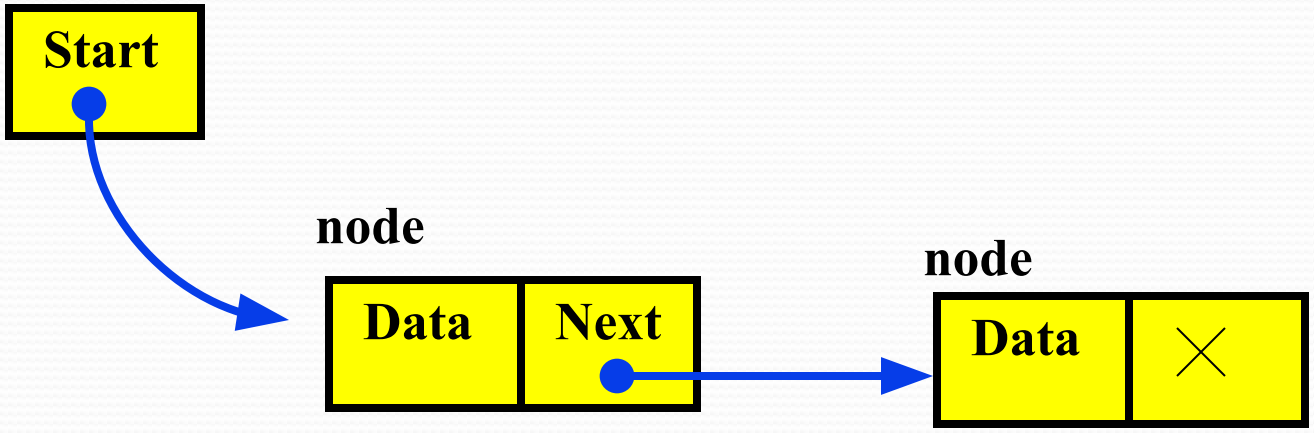
Collection structure has a pointer to the list Start

Initially NULL

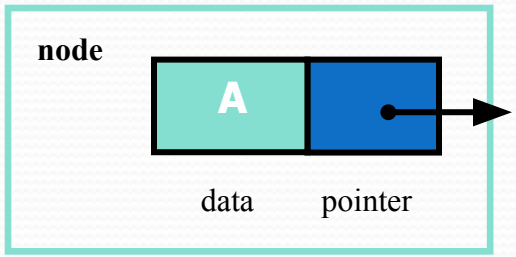
Linked lists (con…)

•A linked list, or one way list, is a linear collection of data elements, called nodes, where the linear order is given by means of pointers.
•Each node is divided into two parts:
•The first part contains the information of the element, and
•The second part, called the link field or next pointer field, contains the address of the next node in the list.
•The pointer of the last node contains a special value,called the null pointer.
•A list pointer variable – called START which contains the address of the first node.
•A special case is the list that has no nodes, such a list is called the null list or empty list and is denoted by the null pointer in the variable START.

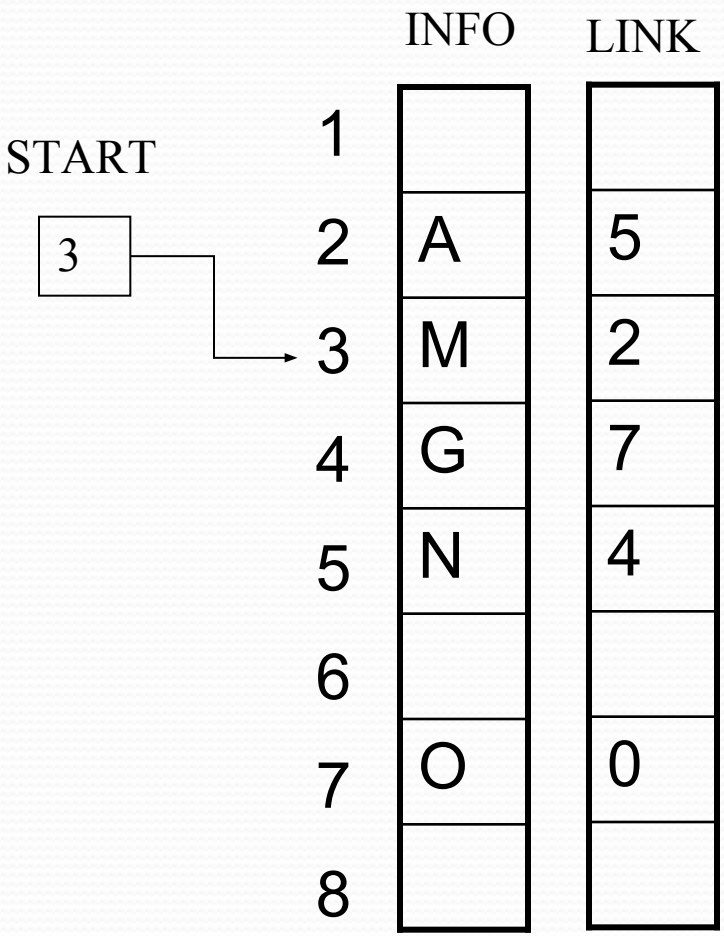
Linked lists (Con…)



Linked list with 2 nodes



Linked lists



START=3,

INFO[3]=M LINK[3]=2,

INFO[2]=A LINK[2]=5,

INFO[5]=N LINK[5]=4,

INFO[4]=G LINK[4]=7,

INFO[7]=O LINK[7]=0, NULL value, So the list has ended

linked lists

•A linked list organizes a collection of data items (elements) such that elements can easily be added to and deleted from any position in the list.

•Only references to near two elements are updated in insertion and deletion operations.

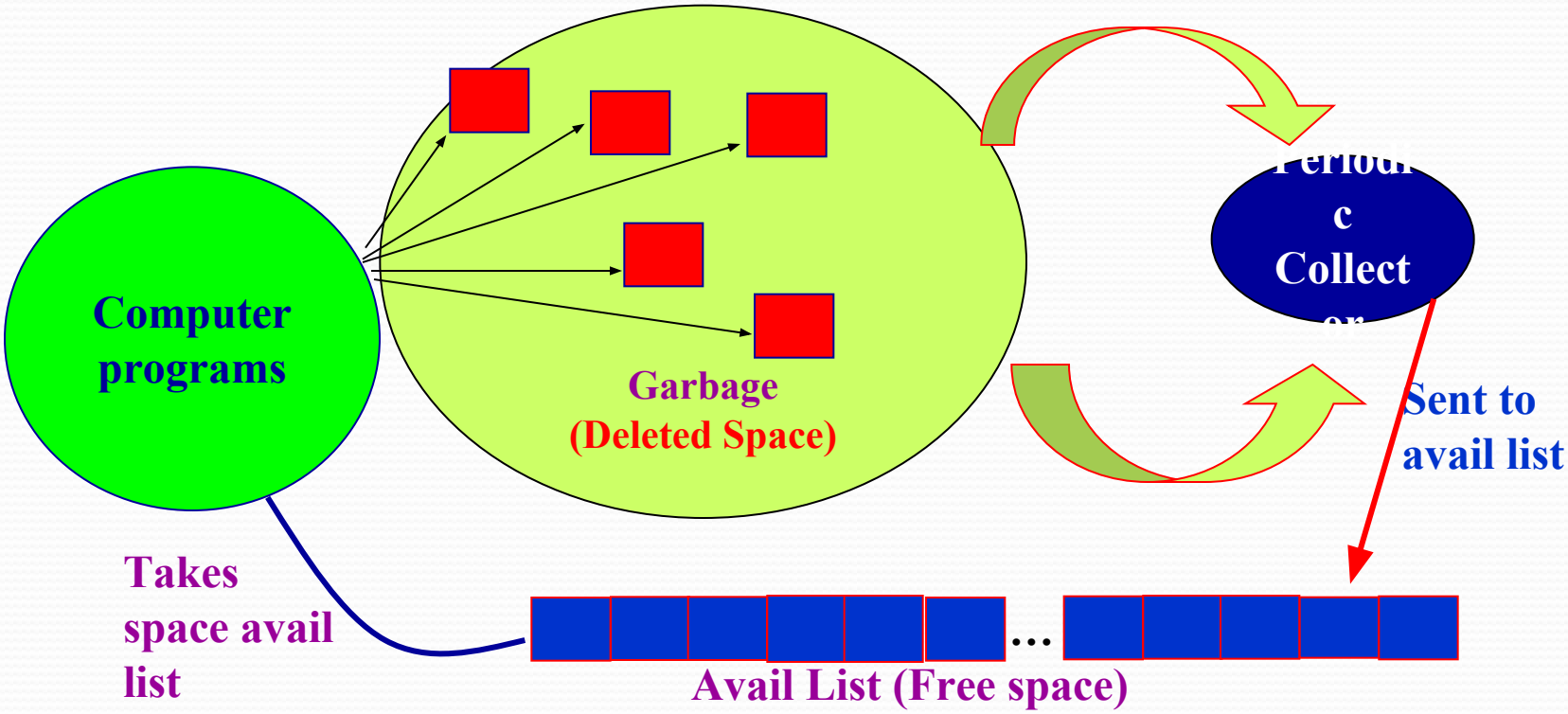
•There is no need to copy or move large blocks of data to facilitate insertion and deletion of elements.

•Lists grow dynamically.

For the above reason link list is great to study.

Memory allocation: Garbage collection

- Memory space can be reused if a node is deleted from a list
  - i.e deleted node can be made available for future use
- The operating system of a computer may periodically collect all the deleted space on to the free storage list. Any technique which does this collection called garbage collection

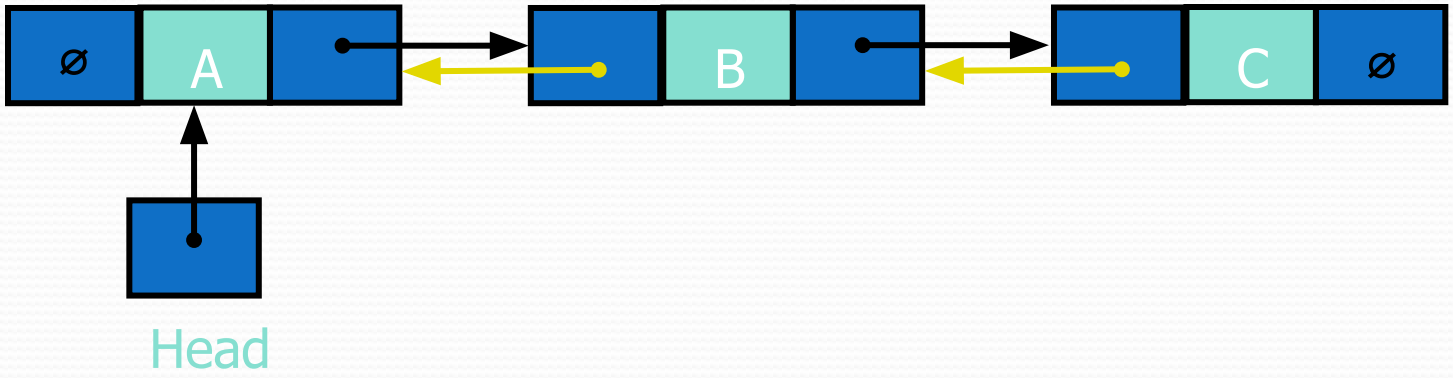


Overflow and Underflow

- **Overflow**:
  - Sometimes data are inserted into a data structure but there is no available space.
  - This situation is called overflow
  - Example: In linked list overflow occurs when
    - AVAIL= NULL and
    - There is an insertion operation
- **Underflow**:
  - Situation:
    - Want to delete data from data structure that is empty.
  - Example: In linked list overflow occurs when
    - START = NULL and
    - There is an deletion operation

Variations of Linked Lists

- ***Doubly linked lists***
- *A linked list in which each node has three parts :one information and 2 pointers:*
- *An information field which contains the data of node.*
- *a forward pointer (a pointer to the next node in the list) and*
- *a backward pointer (a pointer to the node preceding the current node in the list) is called a doubly linked list. Here is a picture:*



Each node points to not only successor but the predecessor
There are two NULL: at the first and last nodes in the list

–Advantage: given a node, it is easy to visit its predecessor. Convenient to traverse lists backwards.

The primary **disadvantage** of doubly linked lists are that

(1) Each node requires an extra pointer, requiring more space, and

(2) The insertion or deletion of a node takes a bit longer (more pointer operations).

Basic Operations

You all know the basic operations?

Yes

Insertion
Deletion
Traversing
Searching
Sorting
Marging

Traversing a linked lists

LIST be a linked list in memory stored in linear arrays INFO and LINK with START pointing to the first element and NULL indicating the end of LIST.

We want to traverse LIST in order to process each node exactly once.

Pointer variable PTR points to the node that is currently being processed.

LINK[PTR] points to the next node to be processed.

Thus update PTR by the assignment

PTR : =LINK[PTR]

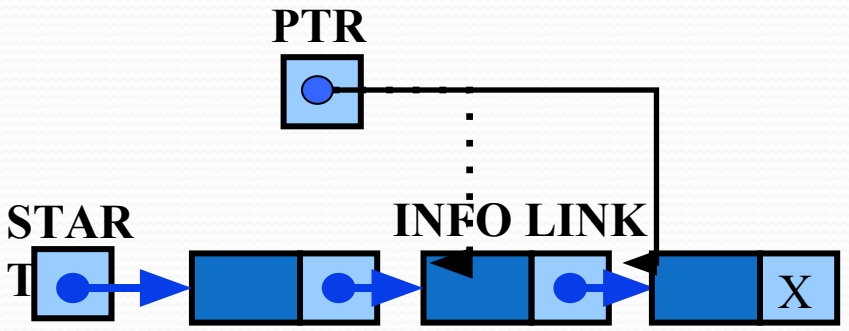


Fig : PTR : = LINK[PTR]

Question?