Chapter 4
LIST

Data Structure & Algorithm

Compiled by :

Ankit Bhattarai
Email: ankit.bca@kathford.edu.np

Syllabus

Unit	Contents	Hours	Remarks
1.	Introduction to Data Structure	2	
2.	The Stack	3	
3.	Queue	3	
4.	List	2	
5.	Linked Lists	5	
6.	Recursion	4	
7.	Trees	5	
8.	Sorting	5	
9.	Searching	5	
10.	Graphs	5	
11.	Algorithms	5	

Credit : 3

Outlines

- Introduction
- Static & dynamic list structures
- Array Implementation of List
- Queues as a list

Introduction to List

- A **list** is a sequence of related data item. It can be ordered collection of integer items like 30, 40, 75, 80 or it can be collection of structures likes

```
    struct student
            {
            int age;
            char name[20];
            }
```

- Stacks and queues are special types of lists called **restricted list** because insertion and deletion are restricted to occur only at the ends of list.
- In simple words, a list is a series of linearly arranged finite elements (numbers) of same type.

Introduction to List

Applications of List:

- ✓ Lists can be used to store a list of data elements, they can grow & shrink dynamically.
- ✓ Implementation of stacks, queues & graph.
- ✓ Applications like Music player, image viewer & web browser (previous & next).

Basic Operations in List:

- ✓ Creating a list
- ✓ Traversing the list
- ✓ Inserting an item in the list
- ✓ Deleting an item from the list
- ✓ Concatenating two lists into one

List as an ADT

Values:

A list contains sequence of zeros or more elements of given type T. The elements may be a_1, a_2,…….an with the following properties:

- n is called the size of the list
- If n = 0, we have an empty list (which has no element)
- If n > =1 then a1 is the first element & an is the last element.
- Elements are ordered according to their position a_i precedes a_{i+1}

List as an ADT

Operations:

i. Insert(x, p, L) : Insert x at the position p in the list L.
ii. Find(x, L): Return the position of first occurrence of x in L.
iii. Retrieve(p, L): Return the element at position p in L.
iv. Delete(p, L): Delete the element at position p of the list L.
v. Next(p, L): Returns the element of the position following p on list L.
vi. Previous(p, L): Returns the element at position preceding position p on list L.
vii. makeEmpty(L): Make the list L an empty list.
viii. First(L): Return the first position element of list L.

Implementation of List

1. Static Implementation (Array)
2. Dynamic Implementation (Linked List)

<u>Explanation:</u>

1. Static Implementation (Array)

- In Static implementation the size of the structure is fixed. The content of the data structure can be modified but without changing the memory space allocated to it.
- Array is used in static implementation of list.

Static Implementation: Array Vs List

- The no. of elements of an array is fixed when an array is created. In contrast, the no. of elements in the list can be varied by adding or removing elements.
- The index of a component of an array is unique & fixed. In contrast, the position of an element in a list can vary over time. If we remove the first element of the list, then elements which was previously second in the list now becomes the first.
- All the components of an array can be accessed easily in constant time. But the components of the list are accessed in variable time. The first element in the list can be accessed in shortest time while the last element is accessed in maximum time.

Static Implementation: Disadvantages of Array

1. Elements of arrays are always stored in contiguous memory.
    - Inserting or deleting an element in an array may require all of its elements to be shifted.
2. The size of array is always fixed.
    - You cannot add a new element beyond the end of the array.
    - Memory of the entire array is always reserved even though you only use part of array.
3. You must guess the expected maximum size of the list in advance.

Static Implementation: Algorithm for insertion in Static List

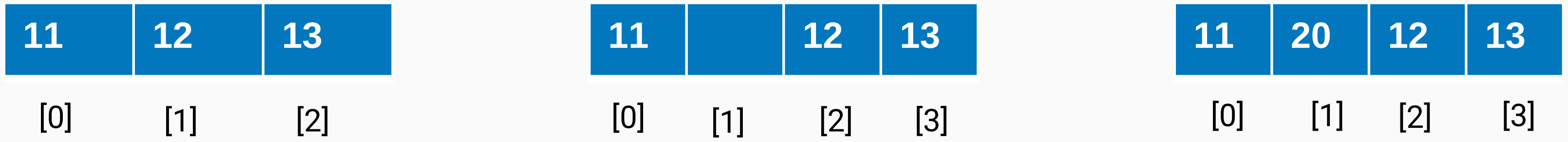
Step 1: Start

Step 2: Suppose the array to be **arr[max]**, **pos** be the postion at which the element is to be inserted.

Step 3: For insertion all the elements starting from the postion pos are shifted towards the right to make a vacant space where the element is inserted.

Step 4: Stop

Insert at pos =1 & element = 20



Static Implementation: Algorithm for deletion in Static List

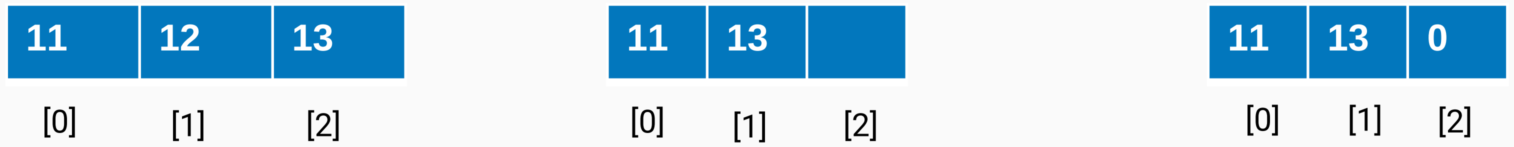
Step 1: Start

Step 2: Suppose the array to be **arr[max]**, **pos** be the postion at which the element is to be deleted.

Step 3: For deletion all the elements to the right of the element at position **pos** are shifted to the left & the last vacant space is filled with 0.

Step 4: Stop

Delete at pos =1



Dynamic Implementation of List

- In Dynamic implementation the size of the structure in not fixed and can be modified during the operations performed on it.
- Linked List is used as a dynamic list.
- A linked list is a data structure is a dynamic structure. It is an ideal technique to store data when the user is not aware of the number of elements to be stored.
- **Linked list** is a series of data items with each item containing a pointer giving a location of next item in the list.

Dynamic Implementation of List

- In other words, it is a collection of nodes where each node consists of at least two parts:

   i. **Information field or info field** : It contains the actual element to be stored.

   ii. **Linked or address field**: It contains one or two links that points to the next node or previous node in the list.

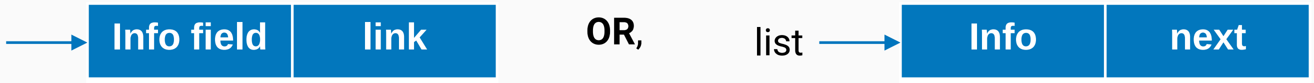
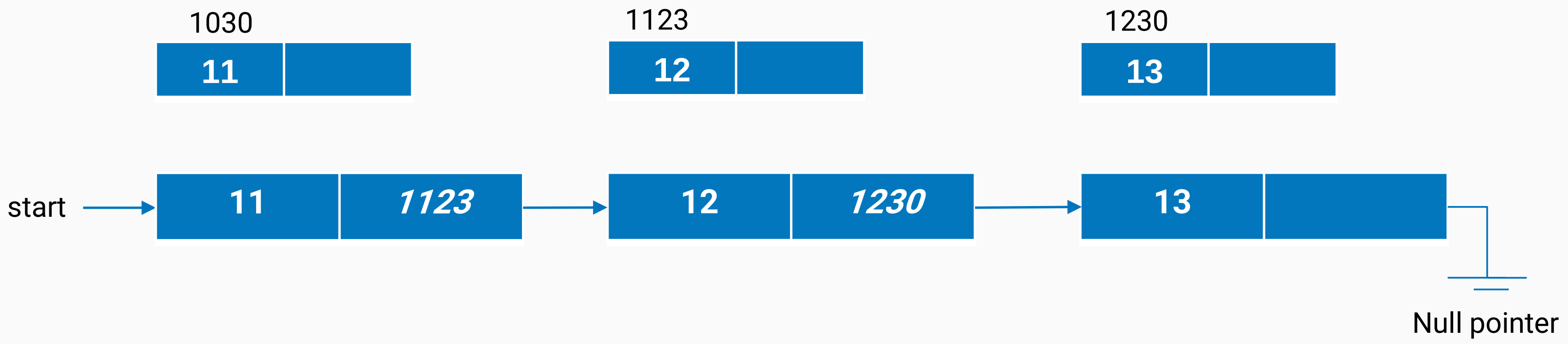


Figure. A node

✓ The memory for nodes of the linked list can be allocated dynamically whenever required.

Dynamic Implementation of List



Note:

- The first element of the list is known as **head** & the last element is known as **tail**.

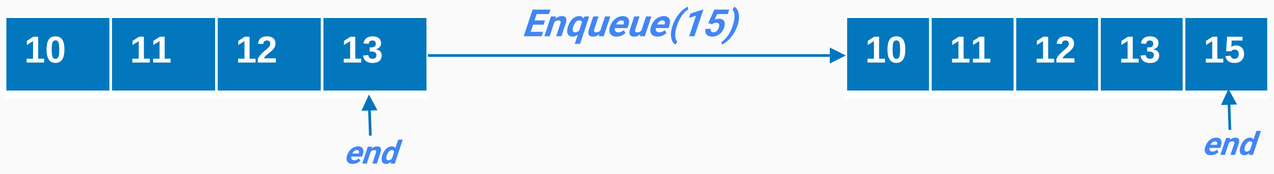
Queue as a List

- ✓ Let's, first understand **list operation**
  - Insert a new value at the start of the list.
  - Insert a new value at the end of the list.
  - Insert a new value at the nth position of the list.
  - Delete a value at the start of the list.
  - Delete a value at the end of the list.
  - Delete a value at the nth position of the list.
- ✓ Now recall **queue operation**
  - Two ends; insertion at rear end & deletion at front end
  - Enqueue: Insert at the end of the queue.
  - Dequeue: Delete at the front of the queue.

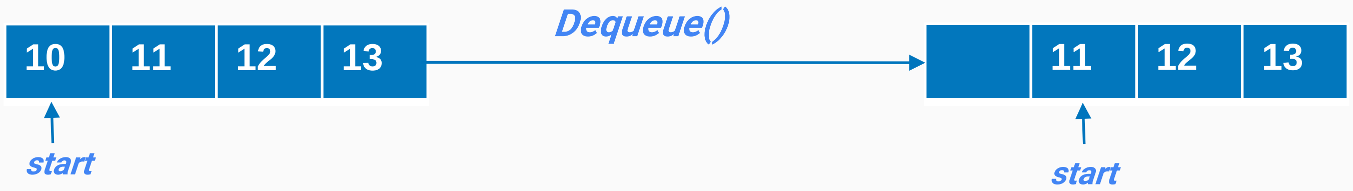
Queue as a List

✓ So, implementing queue as a list

- For enqueue operation, we can every time perform **insert a new value at the end of the list.**



- For dequeue operation, we can every time perform **delete a value at the start of the list.**



QUESTIONS TO FOCUS

1. Define list. Explain list as an ADT.
2. List down the applications & operation of list.
3. Difference between static & dynamic list.
4. Explain the static implementation of list.
5. What is linked list ? Why is it an example of dynamic list.
6. Write down the disadvantages of array. How does list differ to array ? Explain.
7. Explain Queue as a list.

THANK YOU

Any Queries ?

ankit.bca@kathford.edu.np