# Set Representation and Operations

Unit 3: Lecture 01

# Set Representation and Operations

A set is a **collection of distinct elements** (also called **members**): A={a,b,c}

Few **basic operations** on the sets are:

- Member/IsMember
- MakeNull
- Insert
- Delete
- Union
- Intersection
- Difference
- Find

# Representation of Sets

There are various ways to represent the sets:

- Bit Vector Representation
- List Representation
- Dictionary/Hash Table Representation
- Priority Queue Representation
- Tree and Trie Representation

# Representation of Sets: Bit Vector

A bit vector representation **marks the presence** of an element **by 1** and absence by 0: in the bit vector, $i^{th}$ bit is on if *i* is a member of the set

This representation is useful when the sets under discussion are subsets of a ***small universal set*** usually with integer values as the members

**Example:** The set {2, 3, 7, 9} is represented by the bit vector <0,1,1,0,0,0,1,0,1> where universal set is the set of natural numbers less than 10

**Advantage:** Member, Insert, Delete operations can be performed in constant time and Union, Intersection and Difference operations can be performed in time proportional to the size of the universal set

# Representation of Sets: List

List (linked) representation uses space proportional to the size of the set in which each member of the set is represented as an item of the list

List representation is **more general** as it can handle any type of member data which need not be a subset of some universal set (as compared to bit vector representation)

To perform various operations, the **ordering (linear)** of universal set is helpful, e.g., a sorted sequence

In a linear ordered set, the members are **comparable by the relation** < (named order by), e.g., e1<e2<e3…

The ordering has an **advantage** that the entire list is not to be searched to determine the presence of an element

# List Representation (contd.)

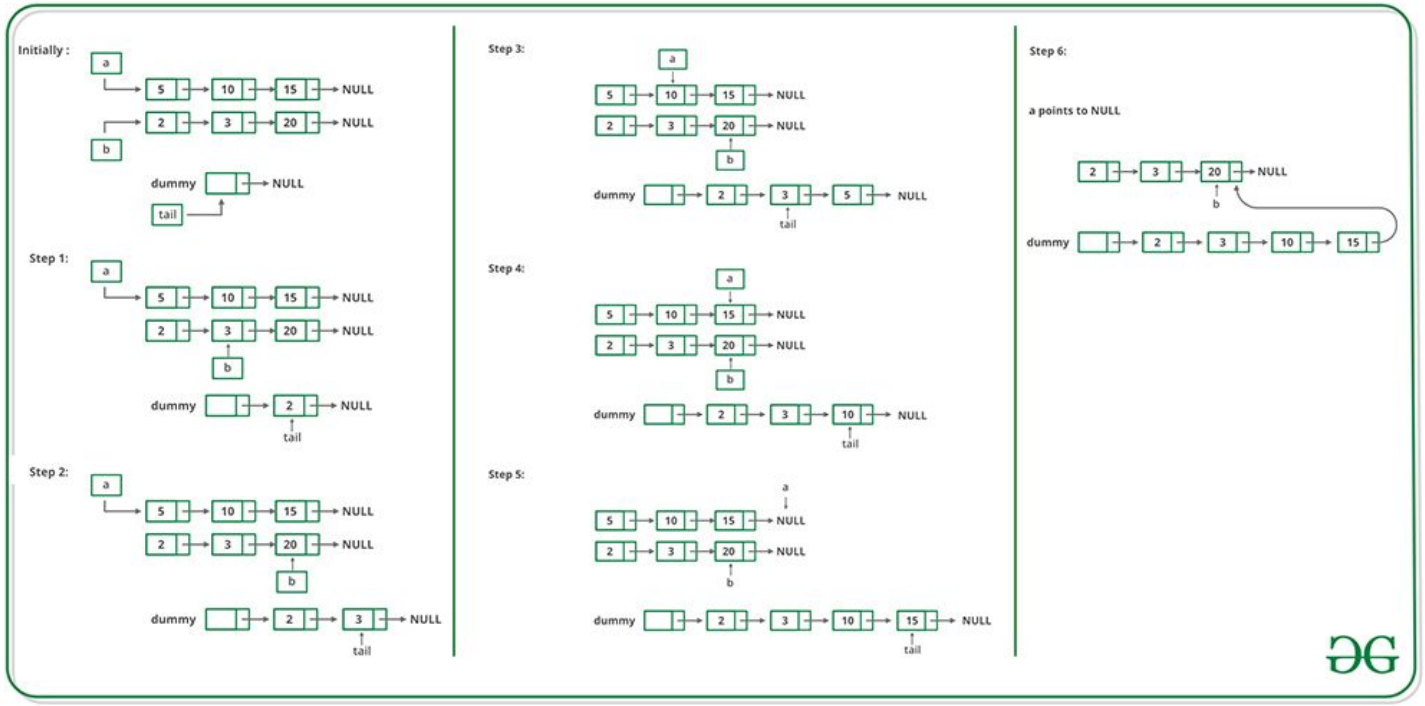
For various operations, the **unordered lists cause O($n^2$) steps** (comparing each element of one list with each of the other), which can be reduced with the help of ordered lists

For example, in **Intersection operation** on two lists L1 and L2 (linearly ordered), an element of first list is to be compared with an element of second list and continued if the next elements are comparable (cf. merge sort!), with the cost **O(n)**

Similar effect can be observed with **Union** and **Difference operations**

**Summary:** All the set operations are to be performed on sorted linked list(s)

# List Representation (contd.)




source:https://media.geeksforgeeks.org/wp-content/cdn-uploads/20210409164102/Merge-Two-Sorted-LinkedLists.jpg

# Representation of Sets: Dictionary/Hash Table

Often a set of ***current objects*** with periodic insertions and deletions are used in practice

A set with frequent Insert, Delete and Member operations is named as **dictionary**- a dictionary is associated with a set of keys and each key has a single associated value

A dictionary can be implemented by a **sorted/unsorted linked list**, by a **bit vector** (if the elements are integers) or a **fixed length array** (if the sets don't grow beyond the length of the array)

The array or list implementation requires **O(n)** steps to execute a single Insert, Delete or Member operation on a dictionary of n elements

# Dictionary/Hash Table Representation (contd.)

Another implementation of a dictionary is a **hash table** which requires **constant time per operation** (on the average) using a hash function

In worst case, hashing (hash table implementation) requires time proportional to the **size of the set** for each operation which may further be limited by a careful hash design

**Open/External Hashing**: allows the set to be stored in a potentially unlimited space, i.e., places no limit on the size of the set

**Closed/Internal Hashing**: allows a fixed space for storage and limits the size of the set

# Representation of Sets: Priority Queue

The priority queue representation is based on the **set model with** the operations **Insert** and **DeleteMin**

The operation DeleteMin is similar to the Delete operation however it is performed on the basis of **priority of each element** of the set. Accordingly it returns some element of **smallest priority** and **deletes that element** from the set

For each element *a*, the priority *p(a)* is a **real number**, which in general, is a member of some linearly ordered set

A priority queue may further be implemented with the help of an **array**, **list** or a **tree** in which the minimum element can be obtained easily (hashing is **not useful** in this case)

# Representation of Sets: Trees

A **tree** (rooted) is an efficient representation of the sets however it is **complex** and hence is often **appropriate for large sets**

The set members are **ordered** by a relation which helps to represent the members in some hierarchy

One of the members (usually the first one) is made the **root of the tree** and other members create the branches

Usually a **binary search tree**, **trie** or **balanced tree** is used to represent the sets

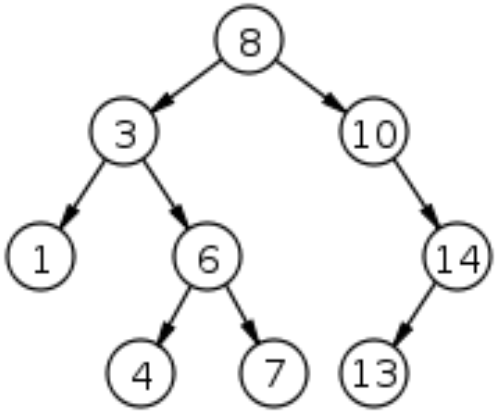
# Tree (BST) Representation (contd.)

A binary search tree (BST) is a binary tree in which the nodes are labeled with elements of a set and follow the **binary search tree property**:

*all the elements stored in the left subtree of any node are less than (using <, i.e., ordered by) the node and all the elements stored in the right subtree of any node are greater than the node*

A BST supports the set operations Member, Insert, Delete in **O(log n)** steps on the average where n is the number of elements in the set

# Tree (BST) Representation (contd.)




Source:https://upload.wikimedia.org/wikipedia/commons/thumb/d/da/Binary_search_tree.svg/180px-Binary_search_tree.svg.png

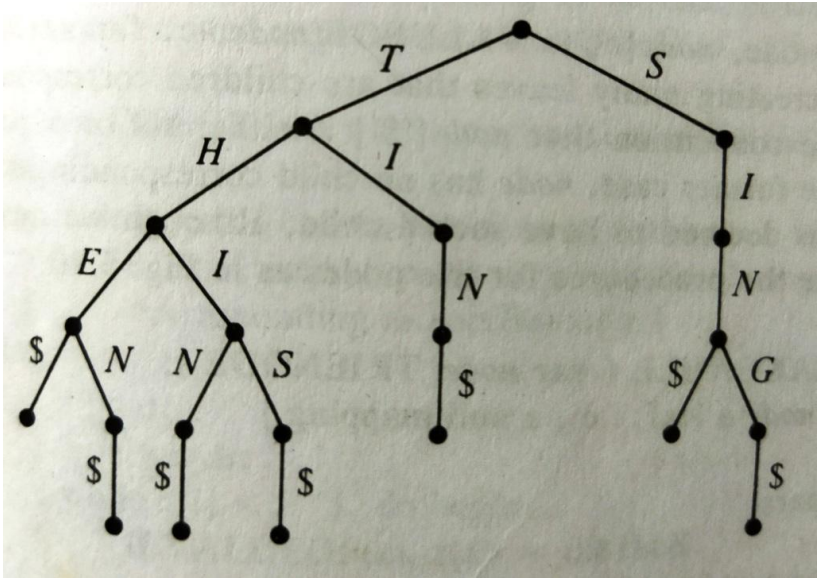
# Representation of Sets: Trie

The **set of character strings** is represented by a special structure called trie

In a trie, each path from the root to a leaf corresponds to one word in the represented set thus the nodes of the trie correspond to the prefixes of words in the set

For implementation, an endmarker symbol ($) is used to mark the end of the words in the trie corresponding to the leaf nodes- it restricts a prefix of a word to be a word itself

# Trie Representation (contd.)




Source: Aho, Hopcroft, Ullman (Book)

# Reference Book

Data Structures and Algorithms: Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, 10th Impression, Pearson Education, New Delhi