UNIT-II
DICTIONARIES

DICTIONARIES

A **dictionary** is a container of elements from a totally ordered universe that supports the basic operations of inserting/deleting elements and searching for a given element.

In this chapter, first, we introduce the abstract data type **Set** which includes dictionaries, priority queues, etc. as subclasses.

Sets:

The set is the most fundamental data model of mathematics.

A set is a collection of well defined elements. The members of a set are all different.

There are special operations that are commonly performed on sets, such as Union, intersection, difference.

1. The **union** of two sets S and T, denoted S ∪ T, is the set containing the elements that are in S or T, or both.
2. The **intersection** of sets S and T, written S ∩ T, is the set containing the elements that are in both S and T.
3. The **difference** of sets S and T, denoted S − T, is the set containing those elements that are in S but not in T.

For example:

Let S be the set {1, 2, 3} and T the set {3, 4, 5}. Then

S ∪ T = {1, 2, 3, 4, 5}, S ∩ T = {3}, and S − T = {1, 2}

Set implementation:

Possible data structures include

- Bit Vector
- Array
- Linked List
  - Unsorted
  - Sorted

Dictionaries:

A dictionary is a dynamic set ADT with the operations:

- Search(S, k) – an access operation that returns a pointer x to an element where x.key = k
- Insert(S, x) – a manipulation operation that adds the element pointed to by x to S
- Delete(S, x) – a manipulation operation that removes the element pointed to by x from S

Definition: A Dictionary is a collection of pairs of the form (K, E), where K is a key and E is the element associated with the Key K.

Used to store elements so that they can be quickly located using keys

Example: A dictionary may hold Bank Accounts

- Each Account is an Object that is identified by an Account number (Key).
- An Application wishing to operate on Account would have to provide the Account number as a Search Key

A Dictionary with duplicates is similar to the dictionary but it permits two or more (Key, Element) pairs to have the same Key.

Example: A Telephone Directory is an example of dictionary with duplicates.

Dictionary ADT: A dictionary isdynamic set ADT with the operations are given below is

AbstractDataType Dictionary

{

Instance:

- Dictionary is a collection of a key and value pair.

Operations:

- Create(D) – used for creating an empty Dictionary D.
- Makenull(D) – makes the null set to be the value for the dictionary D by deleting all of its elements.
- Insert(Key, Element) – insert an element with a specified key into
- the dictionary.

- Delete(Key) – delete or remove an element with a specified key.
- Find(Key) – returns the element associated with a specified key from the dictionary
- Size() – returns the number of elements in a dictionary. isEmpty() – tests whether dictionary is empty or not.

}

Implementation:

1. Fixed Length arrays
2. Linked lists: sorted, unsorted, skip-lists
3. Hash Tables: open, closed
4. Trees
   - Binary Search Trees (BSTs)
   - Balanced BSTs
     - AVL Trees
     - Red-Black Trees
   - Splay Trees
   - Multiway Search Trees
     - 2-3 Trees
     - B Trees
   - Tries

Let n be the number of elements is a dictionary D. The following is a summary of the performance of some basic implementation methods:

Worst case complexity of

	Search	Delete	Insert	min
Array	O(n)	O(n)	O(n)	O(n)
Sorted List	O(n)	O(n)	O(n)	O(1)
Unsorted List	O(n)	O(n)	O(n)	O(n)

Among these, the sorted list has the best average case performance.