Week 3

Algorithm design strategies:
Brute force – Bubble sort, Selection Sort, Linear Search.
Decrease and conquer - Insertion Sort.

Algorithm Design Strategies

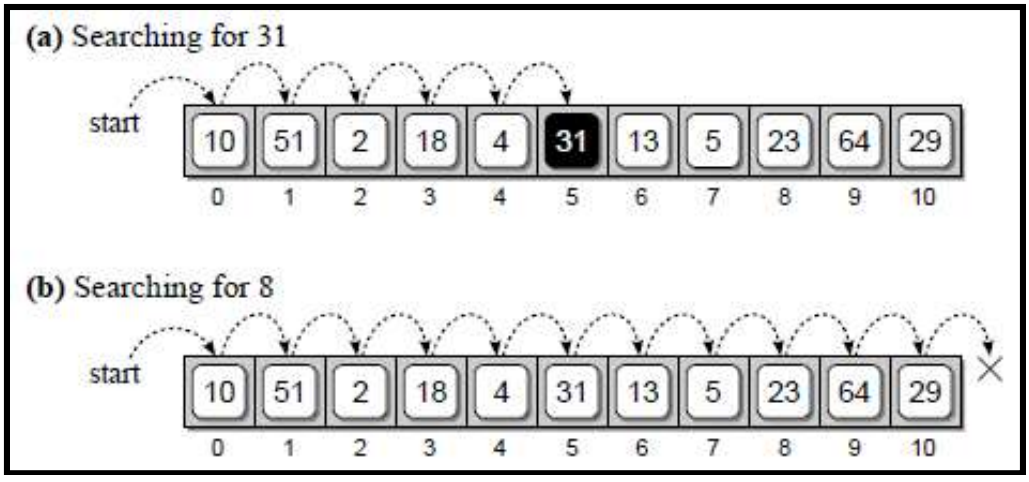
- **Algorithm Design Strategy is a general approach used to solve a given problem.**
- **Some of the common algorithm design strategies are listed below:**
  - ✓ **Brute Force**
  - ✓ **Decrease & Conquer**
  - ✓ **Divide & Conquer**
  - ✓ **Dynamic Programming**
  - ✓ **Backtracking**
  - ✓ **Greedy Method etc.**

Brute Force

- **Brute Force is an Algorithm Design Technique, in which all possible ways are tried to find the solution to a given problem.**

Linear Search

- The simplest solution to the search problem is the sequential or linear search.
- This technique iterates over the sequence, one item at a time, until the key item is found or end of the list reached.
- Linear or Sequential search gives Boolean value (True/False) as result, indicating successful or unsuccessful search.
- Here is an example: Fig (a): key item is found, Fig (b): Key item is not in the list.

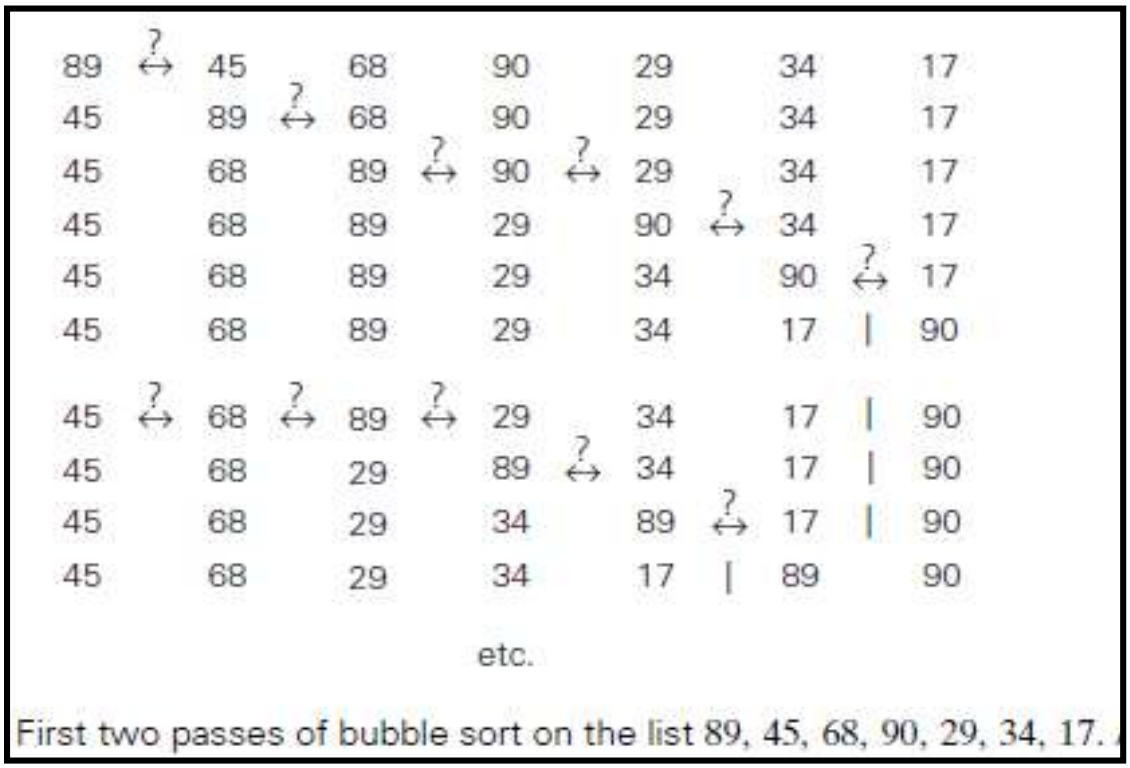


Python Function for Linear Search

```
def linearSearch(a, key):
    n = len(a)
    for i in range(n):
        if a[i] == Key:
            return True
    return False
```

Bubble Sort

- A simple solution to the sorting problem is the bubble sort.
- It is one of the most widely used sorting techniques. It is easy to understand and to implement the program.
- Bubble sort works as follows:
  - ✓ Compare adjacent elements of the list and exchange them if they are out of order.
  - ✓ By doing it repeatedly, we end up "bubbling up" or moving the largest element to the last position on the list.
  - ✓ The next pass moves the second largest element to its final position, and so on.
  - ✓ After n − 1 passes, the list is sorted.
- **Here is an example:**



First two passes of bubble sort on the list 89, 45, 68, 90, 29, 34, 17.

Python Function for Bubble Sort

```
def bubbleSort(a):
    n = len(a)
    for i in range(n-1):
        for j in range(n-i-i):
            if a[j] > a[j+1]:
                temp = a[j]
                a[j] = a[j+1]
                a[j+1] = temp
```

Selection Sort

- This technique selects elements in order and places them into their respective sorted positions.
- Assume we have an array A of size n elements. Now Selection sort mechanism first selects the smallest element in the array and places it at the array position A[0].
- Similarly selects smallest element in the array and place it at the second position in the array A[1].
- The process of selecting next smallest element and placing it at their respective positions continue till all elements in the array gets their respective positions.
- **Here is an example:**

89	45	68	90	29	34	**17**
17	45	68	90	**29**	34	89
17	29	68	90	45	**34**	89
17	29	34	90	**45**	68	89
17	29	34	45	90	**68**	89
17	29	34	45	68	90	**89**
17	29	34	45	68	89	90

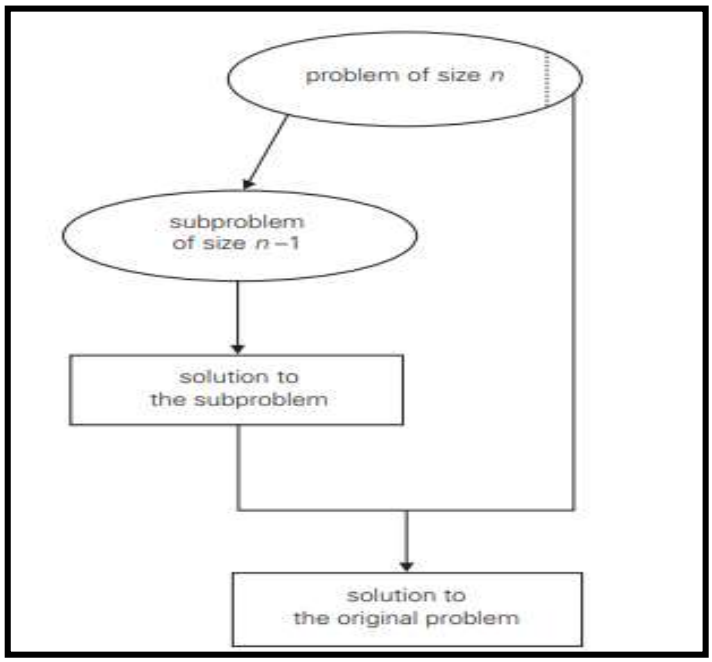
Python Function for Selection Sort

```
def selectionSort(a):
    n=len(a)
    for i in range(n-1):
        min=i
        for j in range(i+1,n):
            if a[j]< a[min]:
                min=j

        temp=a[i]
        a[i]=a[min]
        a[min]=temp
```

Decrease & Conquer

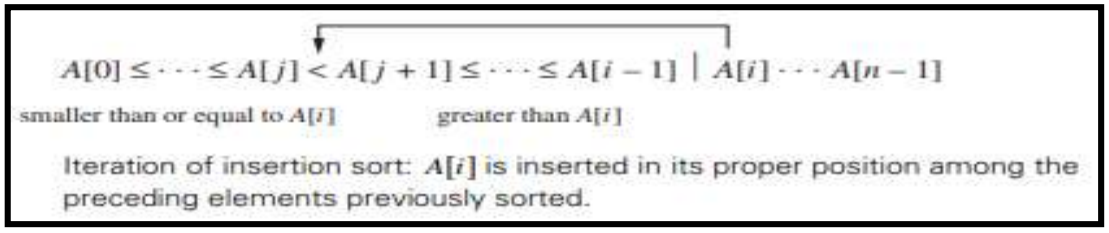
- **Decrease and Conquer is an Algorithm Design Technique, in which size of the input list is reduced on each iteration.**



Insertion Sort

- Insertion Sort works like playing Cards.
- This technique works very well when size of array is small and few array elements are already in sorted order. It sorts elements of unsorted array by inserting them into a sorted array.

- Assume that first element of an array is in its proper position. Now the elements from the unsorted array follow this sorted element after comparing with the sorted array elements and inserting into their respective position in sorted array.
- The process of repeated comparison and insertion continues till all the unsorted array elements are placed into their sorted positions.



Iteration of insertion sort: $A[i]$ is inserted in its proper position among the preceding elements previously sorted.

- **Here is an example:**

Unsorted List		***Sorted List***	***Insertion to take place***
a[0]	28	28	Assume it is sorted
a[1]	22	22,28	insert 22 before 28
a[2]	26	22,26,28	insert 26 between 22 and 28
a[3]	38	22,26,28,38	insert 38 after 28
a[4]	47	22,26,28,38,47	insert 47 after 38
a[5]	30	22,26,28,30,38,47	insert 30 between 28 and 38

Python Function for Insertion Sort

```
def insertionSort(a):
    n=len(a)
    for i in range(1,n):
      val=a[i]
      j=i-1
      while j>=0 and a[j]>val:
        a[j+1]=a[j]
        j=j-1
      a[j+1]=val
```

Program #1: Python Program to Linear Search.

```
def linearSearch(a, key):
    n=len(a)
    for i in range(n):
        if key == a[i]:
            return True
    return False

n=int(input("Enter Size of the List\n"))
a=list()
print("Enter List elements")
for i in range(n):
    num=int(input())
    a.append(num)
key=int(input("Enter Key element to be searched\n"))
print("\nSearch List: ", a)
print("Key:", key)
res=linearSearch(a,key)
if res==True:
    print("\nSuccessful Search")
else:
    print("\nUnsuccessful Search")
```

Output #1:

```
Enter Size of the List
5
Enter List elements
10
8
4
7
2
Enter Key element to be searched
7

Search List:  [10, 8, 4, 7, 2]
Key: 7

Successful Search
```

Output #2:

```
Enter Size of the List
5
Enter List elements
10
8
4
7
2
Enter Key element to be searched
12

Search List:  [10, 8, 4, 7, 2]
Key: 12

Unsuccessful Search
```

Program #2: Python Program to implement Bubble Sort.

```
def   bubbleSort(a):
    n=len(a)
    for i in range(n-1):
        for j in range(n-1-i):
            if a[j]> a[j+1]:
                temp=a[j]
                a[j]=a[j+1]
                a[j+1]=temp

n=int(input("Enter Size of the List\n"))
a=list()
print("Enter List elements")
for i in range(n):
    num=int(input())
    a.append(num)
print("\nUnsorted List")
for i in a:
    print(i,end="\t")
bubbleSort(a)
print("\n\nSorted List")
for i in a:
    print(i,end="\t")
```

Output #1:

```
Enter Size of the List
5
Enter List elements
8
6
4
5
2

Unsorted List
8       6       4       5       2

Sorted List
2       4       5       6       8
```

Program #3: Python Program to implement Selection Sort.

```
def selectionSort(a):
    n=len(a)
    for i in range(n-1):
        min=i
        for j in range(i+1,n):
            if a[j]< a[min]:
                min=j

        temp=a[i]
        a[i]=a[min]
        a[min]=temp

n=int(input("Enter Size of the List\n"))
a=list()
print("Enter List elements")
for i in range(n):
    num=int(input())
    a.append(num)
print("\nUnsorted List")
for i in a:
    print(i,end="\t")
selectionSort(a)
print("\n\nSorted List")
for i in a:
    print(i,end="\t")
```

Output #1:

```
Enter Size of the List
5
Enter List elements
8
6
4
5
2
Unsorted List
8       6       4       5       2
Sorted List
2       4       5       6       8
```

Program #4: Python Program to implement Insertion Sort.

```
def insertionSort(a):
    n=len(a)
    for i in range(1,n):
        val=a[i]
        j=i-1
        while j>=0 and a[j]>val:
            a[j+1]=a[j]
            j=j-1
        a[j+1]=val

n=int(input("Enter Size of the List\n"))
a=list()
print("Enter List elements")
for i in range(n):
    num=int(input())
    a.append(num)
print("\nUnsorted List")
for i in a:
    print(i,end="\t")
insertionSort(a)
print("\n\nSorted List")
for i in a:
    print(i,end="\t")
```

Output #1:

```
Enter Size of the List
5
Enter List elements
8
6
4
5
2

Unsorted List
8       6       4       5       2

Sorted List
2       4       5       6       8
```

Activity #3

1. **Develop Python Program to implement Bubble Sort to sort item in Descending order.**
2. **Develop Python Program to implement Selection Sort to sort item in Descending order.**
3. **Develop Python Program to implement Insertion Sort to sort item in Descending order.**
4. **Develop Python Program to implement Linear Search to search item in reverse order.**
5. **Apply Bubble sort, Selection sort and Insertion sort on following list:**

   10, 8, 9, 15, 7, 14