Unit-IV
Sorting and Searching Techniques

Syllabus:
Sorting and searching techniques:
Need of sorting and searching, sorting order & stability in sorting.

Sorting Techniques: Algorithms for Bubble sort, Selection sort, Insertion sort, Shell sort, Radix sort, Quick sort and Merge sort. Analysis of each sorting technique for best, worst and average case, Concept of Internal & External sorting.

Searching Techniques: Algorithms for Sequential search, Binary search, Fibonacci search & concept of Index Sequential search, analysis of each searching technique for best, worst and average case.

Searching:
Introduction:

"A process of finding particular element in the given list is called searching".

In many of the real life applications searching is required. e.g. In Banking System, say suppose bank administrator want to find account details of particular customer. To find these details, in search process, it will going to compare customer ID with each record in the bank customer database. If customer ID is matching with the customer ID in the database. Then it will get details of the employee so searching is important.

Types of Searching:
   a) Internal Searching
   b) External Searching

Need Of Searching and Sorting:
- Sorting is needed to arrange data in either ascending or descending order.
- When we arrange data in sorted order few of the searching algorithms will going to access that data fastly e.g. Binary search technique is better to find data in given data structure. But it works only on list of sorted data elements.
- Searching technique is needed to find particular data elements in the given list.

Prepared By: Sandip M.Walunj, Lecturer in Computer Engg. Dept., Sandip Foundation, Nashik
**Searching Techniques:**

There are four searching techniques listed as follows:

1) Linear Search
2) Binary Search
3) Fibonacci Search
4) Index Sequential Search

**1) Linear Search:**

**Introduction:**
- Linear search is one of the simplest searching techniques.
- Linear search works on sorted as well as unsorted lists.
- It is called linear search or sequential search because it searches key element with each element in the list one by one in sequence.
- Complexity of linear search is,
  
  Best Case Time Complexity : 1  
  Worst case Time complexity : n  
  Average case Time complexity : 1<= complexity <= n

- The complexity of linear search algorithm is directly proportional to the number of elements in the list so it is not better for list containing more number of elements. In such cases we have to go for another searching techniques like binary search, fibonacci search, hashing, index sequential search.
- Linear search algorithm is better for list containing less elements.

**Working Principle:**

It compares search key element with first element in the list, whether search key element found or not, if not found it will check for next element in the list in sequence until search key element not found or list not gets completely scanned.

It is called linear search or sequential search because for finding a particular value in a list, it checks key element with each element one at a time in sequence until the desired element not found or list can not get completely scanned.

Example:

Here, is a list of 'n' elements (n=5) and find key element 5 is present in the list or not. The diagramatic process is given below:

Carot symbol indicates the comparison of particular element with key element to search. Name of the list is Arr. This process will take 'n' iterations because their are toatal 5 elements in the list.

Prepared By: Sandip M.Walunj , Lecturer in Computer Engg. Dept., Sandip Foundation, Nashik
Iteration 1:
In this iteration compare key element with first element in the list. If it is equal to search key element, search gets terminated and if not equal then continue next iteration.

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 2 & 3 & 4 & 5 & 6
\end{array}
\]

^check Arr[0] equals to Key or not
if Yes, key element is found and it terminates search process
if No, Continue with next process

Iteration 2:
In this iteration compare key element with second element in the list. If it is equal to search key element, search gets terminated and if not equal then continue next iteration.

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 2 & 3 & 4 & 5 & 6
\end{array}
\]

^check Arr[1] equals to Key or not
if Yes, key element is found and it terminates search process
if No, Continue with next process

Iteration 3:
In this iteration compare key element with 3rd element in the list. If it is equal to search key element, search gets terminated and if not equal then continue next iteration.

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 2 & 3 & 4 & 5 & 6
\end{array}
\]

^check Arr[2] equals to Key or not
if Yes, key element is found and it terminates search process
if No, Continue with next process
**Iteration 4:**
In this iteration compare key element with 4th element in the list. If it is equal to search key element, search gets terminated and if not equal then continue next iteration.

<table>
<thead>
<tr>
<th>Arr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

^ check Arr[3] equals to Key or not
if Yes, key element is found and it terminates search process
if No, Continue with next process

**Iteration 5:**
In this iteration compare key element with 5th element in the list. If it is equal to search key element, search gets terminated and if not equal then continue next iteration.

<table>
<thead>
<tr>
<th>Arr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

^ check Arr[4] equals to Key or not
if Yes, key element is found and it terminates search process
if No, Continue with next process

**Algorithm:**

```
Linear_Search(Arr[], N, Key)
```

**Input:** Arr[], N, Key.
Here,
N = Total Number of elements stored in the list
Arr = List of 'N' Number of elements
Key = Element to search in the list

**Output:**
1) Element is found in the list
2) Element not found in the list

**Data structures used:**

I = Keep track of index of each element
flag = Keep track whether element found in the list or not.

Prepared By: Sandip M. Walunj, Lecturer in Computer Engg. Dept., Sandip Foundation, Nashik
Step 1: Start

Step 2:
Accept Size of the list from user i.e. N = 10.
Accept List of 'N' Size elements i.e. int Arr[10].
Accept key element to search from user i.e. Key = 9.

Step 3:
Initialize index variable of list to zero & set flag to zero. i.e. I = 0, flag = 0

Step 4: Check whether I < N, if No then stop
If Yes, then
a) check whether Arr[ I ] = = Key, if No go to step (c)
   if Yes, then
b) Increment flag by 1 i.e. flag = flag + 1
   c) Increment I by 1 i.e. I = I + 1
d) Go to step 4

Step 5: check flag = = 0,
If yes, display message "Element not found in the list".
If No, display message "Element found in the list"

Step 6: Stop

Program:
#include<stdio.h>
#include<conio.h>
void linear_search();
void main()
{
    clrscr();
    linear_search();
    getch();
}

void linear_search()
{
    int Arr[50],N,I,key,Flag=0;
    printf("\n\n\t Enter Size of the list : ");
    scanf("%d", &N);

    printf("\n\n\t Enter array elements : ");
    for(I=0;I<N;I++)
    {
        printf("\n\t Enter array elements %d : ",I+1);
        scanf("%d", &Arr[I]);
    }
```c
printf("\n\n\t Enter key element to search in a list : ");
scanf("%d", &key);

for(I=0; I<N; I++)
{
    if(Arr[I]==key)
    {
        Flag=1;
    }
}
if(Flag>0)
{
    printf("\n\n\t Element is found");
}
else
{
    printf("\n\n\t Element is not found");
}
```

**Advantages:**

1) Linear search can be implemented on sorted as well as unsorted data elements.
2) Linear search is better for list containing less number of elements.

**Applications of Searching and Sorting:**

Following are the applications of searching and sorting techniques.

1) Finding name of person in the telephone directory
2) Finding contact number of person in the mobile
3) Finding student details in the database
4) Payroll Management System of different organizations
5) Database Management Systems
6) Computerized Dictionaries
7) Searching particular thing on internet i.e. Google Search, Yahoo Search, MSN Search etc.

**1) Mobile Applications:**

In mobile phones to find particular persons contact number different searching techniques are used. We store mobile numbers and names in the mobiles. So to arrange names stored in mobile according to sorted order, different sorting algorithms are used.
2) **Database Management systems to maintain data properly**: 
To store particular employees data in sorted order. To search particular employee name in list of names these algorithms are used.

3) **Search engines like Google search engine, MSN search engine etc**
With help of these search engines, we will search whatever content required on the internet and make it available on our computer within fraction of moments.

4) **In online library System**: 
To arrange books in proper order & to check whether particular book is present in library or not we will develop that application.

5) **Banking systems**: 
To arrange customer records properly so as it will be accessible fastly and to check customer details whether it is present in the list or not.

**Sorting**:
“A process of arranging data in a systematic manner by considering order of data elements, that order may be ascending or descending is called sorting”. 
E.g. Arranging records in the student database according to ascending or descending order.

**Types of Sorting**:
- c) Internal Sorting
- d) External Sorting

*Sorting Techniques* :