Searching Techniques

bitCode technologies

learn.innovate.share

Searching Techniques

- Sequential Search
- Indexed Sequential Search
- Binary Search
- Fibonacci Search

Sequential Search

- **sequential search**, also known as linear search is suitable for searching a set of data for a particular value.
- It operates by checking every element of a list one at a time in sequence until a match is found.
- If the data are distributed randomly, on average N/2 comparisons will be needed.
- The best case is that the value is equal to the first element tested, in which case only 1 comparison is needed.
- The worst case is that the value is not in the list (or is the last item in the list), in which case N comparisons are needed.

Sequential Search

- The simplicity of the linear search means that if just a few elements are to be searched it is less trouble than more complex methods that require preparation such as sorting the list to be searched or more complex data structures, especially when entries may be subject to frequent revision.
- Another possibility is when certain values are much more likely to be searched for than others and it can be arranged that such values will be amongst the first considered in the list.

Example

```
for( int I = 0; I < n; i++ )
{
    if( arr[ I ] == key )
        break;
}
If( I == n )
  return not found;
Else
  return found;
```

Indexed Sequential Search

- Indexed sequential search improves search efficiency for a sorted file, but it involves an increase in the amount of space required.
- A table, called an index, is set aside in addition to the sorted file itself.
- Each element in the index consists of a key kindex and a pointer to the record in the file that corresponds to the kindex.
- The elements in the index as well as the elements in the file, must be sorted on the key.
- If the index is one eighth the size of the file, every eighth record of the file is represented in the index.

Binary Search

- A **binary search algorithm** is a technique for finding a particular value in a linear array, by ruling out half of the data at each step.
- A binary search finds the median, makes a comparison to determine whether the desired value comes before or after it, and then searches the remaining half in the same manner.
- A binary search is an example of a divide and conquer algorithm.

Binary Search

- The entries in the table are sorted in increasing order.
- The approximate middle entry of the array is located, and it's key value is examined.
- If it's value is high, then the key value is compared with the middle half of the first half and the procedure is repeated on the first half until the required item is found.
- If the value is too low, then the key value is compared with the middle entry of second half and the procedure is repeated until the required item is found.
- The process continues until the desired key is found or the search interval becomes empty.

Example

```
BinarySearch( A, value ) {

    low = 0 high = N - 1

     p = low+((high-low)/2) //Initial probe position

     while ( low <= high) {

       if ( A[p] > value )
             high = p - 1
        else if ( A[p] < value )
          low = p + 1
             else
             return p

       p = low+((high-low)/2) //Next probe position.

 }

 return not_found
```

bitCode technologies

Fibonacci Search

- The Fibonacci sequence is

  0 1 1 2 3 5 8 13

- Here the index will be taken according to the fibonacci sequence and a range in which the num falls will be selected and searched.

- This searching technique needs sorted array.