Searching Techniques

Prof. Pujashree Vidap

Linear Search

- Search for an item in an array one after another linearly fashion
  - Iterative
  - Recursive

```
Algorithm LinearSearchRec(A[ ], item,index, N){
If (index >=N)
    Return -1
Else if (item==A[index])
    Return index
Else
    Return LinearSearchRec(A,item,index+1,N)
    }
```

Sentinel Search

- Idea is replace last element with search element
- Run the loop to see there is copy of search element exists
- Quit the loop once element is found

```
Algorithm SentinelSearch(array, item,N) {
last = array[N-1];
array[N-1] = item;    // Here item is the search element.
i = 0;
while(array[i]!=item)
{
    i++;
}
array[N-1] = last;
if( (i < N-1) || (item == array[N-1]) )
    print "Item Found at " + i
else
   print "Item Not Found"
}
```

Binary Search

```
Algorithm Binarysearch( A, N,
Data)
{
//A is an array of N elements.
//Data is the element to be
//searched
  1. low := 0
  2. high := N – 1
  3.while(low<=high)
     {
     3.1  mid := (low + high)/2
     3.2  if (A[mid] = Data){
         3.2.1 write “Found”;
         3.2.2 exit ;
}
     3.3. If( Data < A[mid])
         3.3.1 high := mid – 1
     3.4.If( Data > A[mid])
         3.4.1 Set low := mid + 1
}
4.Print  “Not Found”
5.Exit
}
```

Binary Search using Recursion

```
Algorithm binsearch(A, low, high, Data, N)
{
   1.if(low=high) then{
            1.1  if(Data=a[low]) then return low;
                       else return -1;}
       else
            1.1 mid := (low + high)/2
            1.2  if (A[mid] = Data) then return mid;
            1.3 else if(x<a[mid] then
                      return binsearch(a,low,mid-1,Data,N)
                 else return binsearch(a,mid+1,high,data,N);
}
```

Fibonacii Search

- Applied on sorted arrays
- It uses Fibonacci series to determine the index position to be searched in the array.
- Steps
- 1.Find out a fibonacci number (F_m) that is greater than or equal to the size of the array. Then
- 2. Compare the item with the element at F_{m-1} position in the array.
- 3. If they are equal, search is successful, element is at $F_{m-1}+1$.
- 4. If the item is smaller, it is searched in the sublist left to F_{m-1}.
- 5. If the item is greater, it is searched in the sublist right to F_{m-1}.

- If *item* is not found, again the fibonacci number >= size of the sublist to be searched is taken, and the whole process is repeated until the desired element is found or the sublist is reduced to a single element that is not equal to *item*.

```
Algorithm Fibonacii_Search(A, n, item)
{
        int  flag, first, last, loc, index;
        flag=0;first=0;loc=-1; last=n-1;
        while( flag !=1  && first<=last)
        {
                index =return_finonacci(n);//returns $F_{m-1}$th Fibonacci no.
                if ( item = = A[index +first])
                {
                        flag=1;
                        loc =index;
                        break ;
                }
                else if (item  > A[index +first])
                {
                        first=first + index +1;
                }
                else
                {
                        last = first + index - 1 ;
                }
                n=last-first+1;
    }  //end of while
    if  (flag= =1)
          return  (loc+first+1);
    else
           return -1  ;//item not found
}
```

```
return_finonacci(n)
{
        int a, b, c;
        a =b = c =1;
        if (n ==0 || n ==1)
                return 0;
        else
        {
                while (c < n)
                {
                        c = a + b;
                        a  = b;
                        b =c;
                }
        }
        return a;
}
```

Analysis

- The complexity of Fibonacci search is $O(\log_2 n)$
- The performance of Fibonacci search is poor than binary search.
- However, binary search involves division operation , where as in Fibonacci search only addition and subtraction operation is involved. Average performance of Fibonacci search may be better than binary search on computers where division is more time consuming than addition or subtraction.

Differences with Binary Search:

- Fibonacci Search divides given array in unequal parts
- Binary Search uses division operator to divide range. Fibonacci Search doesn’t use /, but uses + and -. The division operator may be costly on some CPUs.
- Fibonacci Search examines relatively closer elements in subsequent steps. So when input array is big that cannot fit in CPU cache or even in RAM, Fibonacci Search can be useful.