Searching Techniques

Searching

- Linear Search
- Binary Search

Linear search Algorithm

Linear search(list, n, element)
Where list= Represents the list of elements
n = Represents the size of the list
element= Represents the value to search

Step 1: [Initialize]
K=1
Flag=1

Step 2: Repeat through step 3 for K=1,2,3…n

Step 3: If list[K]=element
(i) Flag=0
(ii) Output "search is successful"

Step 4: If Flag
Output "search is unsuccessful" and Exit

Step 5:Exit

Linsrc
h

Time complexity for linear search

- The time complexity of sequential search
- For successful search

  - worst case is O(n)
  - Best case is O(1)
- For unsuccessful search, O(n)

Binary Search

- Binary search works only on a sorted set of elements. To use binary search on a collection, the collection must first be sorted.
- need to know the start and end of the range. Lets call them Low and High.
- compare the search value K with the element located at the median of the lower and upper bounds. If the value K is greater, increase the lower bound, else decrease the upper bound.
- It is base on divide and conquer apporach.

Binary search Algorithm

Binary search(list, n, element)

Where list= Represents the list of elements
n = Represents the size of the list
element= Represents the value to search

Step 1: [Initialize]
low=1
high=n
flag=1
Step 2: Repeat through step 4 while (low<=high)
Step 3: mid=(low+high)/2
Step 4: If (element< list[mid])
then
high=mid-1
else If (element > list[mid])
then
low=mid+1
else If (element == list[mid])
Output “search is successful” and location of element is mid
flag=0
return
Step 5: If (Flag) Output “search is unsuccessful” and return
Step 5:Exit

binsrch

Time complexity for binary search

- The time complexity of binary search
  - For successful search:
  - worst case is O(log n)
  - Best case is O(1)
  - For unsuccessful search, O(n)
- For unsuccessful search:
  - worst case, best case and average case, O(log n)