Search Algorithms

Linear Search – O(n) search
Data: Array of n elements
Algorithm LinearSearch(target): returns integer

LinearSearch returns the index of the first occurrence of target in the Data array or -1 if target is not found.

```plaintext
probePos = 0
while(Data[probePos] != target and probePos < length of Data)
    probePos++
end while
if(probePos== length of Data
    return -1
else
    return probePos
end LinearSearch
```

Binary Search (iterative) – O(log n)
Data: Array of n elements
Algorithm BinarySearch(target): returns integer

BinarySearch returns the index at which target is found in the Data array or -1 if target is not found.

Requires: Data[i-1] ≤ Data[i] for 1 ≤ i < length of Data

```plaintext
start = 0
end = length of Data - 1
mid = (start + end + 1)/2
found = false
while(not found and end > start)
    if (Data[mid] == target)
        found = true
    else if(target < Data[mid])
        end = mid - 1
        mid = (start + end + 1)/2
    else
        start = mid + 1
        mid = (start + end + 1)/2
    end if
end while
if(found)
    return mid
else
    return -1
end BinarySearch
```
Binary Search (recursive) – $O(\log n)$

Data: Array of $n$ elements

Algorithm BinarySearch(target, start, end): returns integer

$$\text{mid} = (\text{start} + \text{end} + 1)/2$$

if(Data[mid] == target)
  return mid
else if (end <= start)
  return -1
else if(target < Data[mid])
  return BinarySearch(target, start, mid-1)
else
  return BinarySearch(target, mid+1, end)

end BinarySearch