***Data Structures: Searching***

# Searching

* Searching refers to the operation of finding the location LOC of ITEM in DATA
* Search is said to be successful if ITEM is found in DATA set otherwise search is said to be unsuccessful
* 2 types of search techniques
* Linear Search
* Binary Search

# Searching

* **Linear Search**
* **Time is proportional to *n***
* **We call this time complexity *O(n)***
* **Pronounce this “big oh” of *n***
* **Both arrays (unsorted) and linked lists**
* **Binary search**
* **Sorted array**
* **Time proportional to log*2 n***
* **Time complexity *O(*log *n)***
* ***Only used in arrays***

# Searching – Linear Search

* Linear search searches ITEM one by one in

DATA set. If item is found it returns its location otherwise set LOC to NULL and return

* If there are N number of elements in DATA

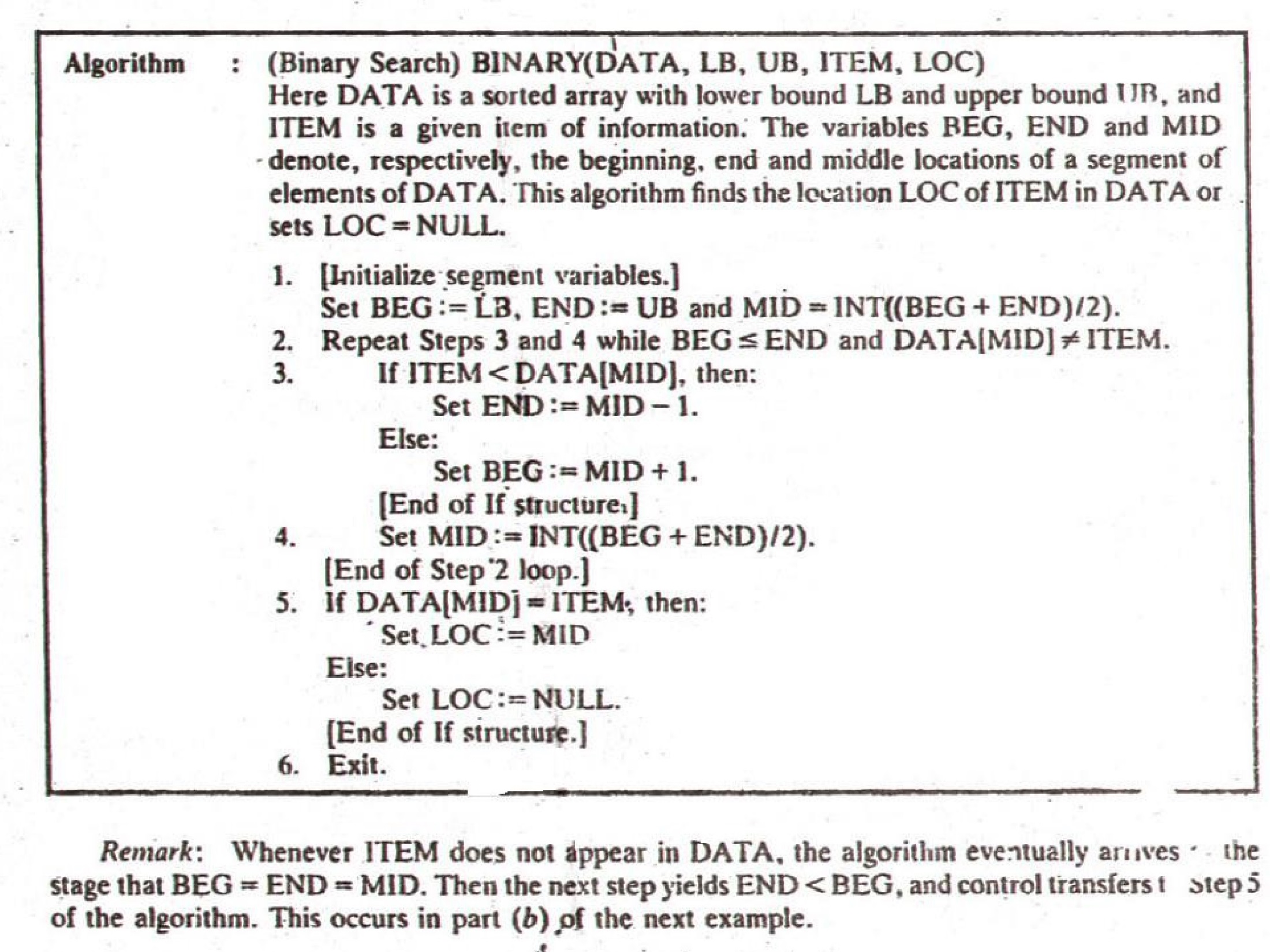
Linear search will compare each element of DATA with given ITEM until either search is successful or no more element to compare.

# Searching : Binary Search

* **Divide-an-Conquer**
* **Elements must be sorted**
* **Efficient Searching technique**
* **Application**
* **Searching in Telephone Directory**
* **Working**
* **Two pointers are used**
* **Begin: which points to 1st element in DATA block**
* **End: which points to last element in DATA block**
* **Another pointer MID points to middle value of the DATA block**
* **MID is determined as**
* **MID = CEIL(BEGIN + END)/2**
* **The given ITEM is compared with DATA[MID]**
* **If DATA[MID] = ITEM then**
* **LOC = MID and Location is returned**
* **Else if DATA[MID] < ITEM**
* **As already stated DATA is sorted then if ascending then item if exist in DATA can only be found on left of DATA[MID]**
* **There fore we need only to search the ITEM to left of mid value and discard all other elements**
* **Thus to do this all we have to do is to move END to point to end of the left portion of DATA[MID] i.e END = MID-1. No changes is required in BEGIN**
* **Else if DATA[MID] > ITEM**
* **As already stated DATA is sorted then if ascending then item if exist in DATA can only be found on right of DATA[MID]**
* **There fore we need only to search the ITEM to right of mid value and discard all other elements**
* **Thus to do this all we have to do is to move**

**BEGIN to point to start of the right portion of DATA[MID] i.e BEGIN = MID+1. No changes is required in END**

* **We continue with the search by dividing the DATA block until either search is successful or BEGIN > END at which the search terminates NULL is returned showing that search is unsuccessful**



# EXAMPLE

* **Data : 11 ,22, 30, 33, 40, 44, 55, 60, 66, 77,**

**80, 88, 89**

* **ITEM = 40**
* **Initially BEGIN = 1 and END = 13** • **MID = 7, DATA[MID] = 55**
* **Since 40<55 so**
* **END = MID -1 = 7-1 = 6 so new MID = 3, DATA[MID] = 30**
* **Since 40>30**
* **BEGIN = MID + 1=4 so**
* **New MID = 5, DATA[MID] = 40**
* **ITEM found so LOC = MID = 5**

***EXAMPLE cont..***

* **Same data elements**
* **ITEM = 85**
* **BEGIN = 1, END = 13, MID = 10, DATA[MID] = 77**
* **85>77 so**
* **BEGIN = 10+1 =11**
* **New MID = 12, DATA[MID] = 88**
* **85<88 so**
* **END =12-1 =11**
* **New MID = 11, DATA[MID] = 80**
* **So BIGIN = END = MID**
* **85 ≠ 80**
* **ITEM not found**

# Duplication Issue

* **Linear Search:**
* **If ITEM occurs more than one time in DATA then location for each occurrence should be returned.**
* **The algorithm for linear search assumes only one occurrence of ITEM**

# Search with insert and delete

* **Search and Insert**
* **The ITEM is searched by using any technique appropriate to given DATA**
* **If it is not found ITEM is inserted at the end of**

**DATA**

* **Search and Delete**
* **The ITEM is searched by using any technique appropriate to given DATA**
* **If it is found ITEM is deleted from the DATA**
* **If more than one time ITEM is found than all occurrence are deleted**