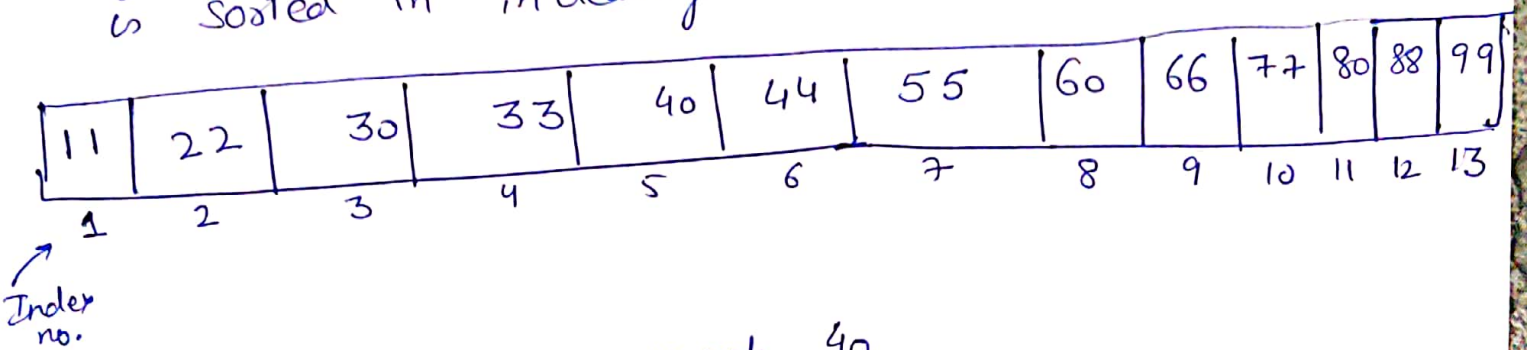
Binary Search

- ~~Consider~~ Array must be in sorted order.

So we consider an array named as DATA which is sorted in increasing numerical order such as



Say we want to search 40

Beg → denote the beginning location of the segment under consideration.

End → the end location of the segment under consideration.

Mid → middle location of a segment under consideration.

Initially Beg = 1 END = 13

therefore MID = INT((Beg + END)/2)

↳ type casting [for collecting integer value]

so MID = INT((1 + 13)/2) = 7

Compare 40 with Data[MID] ∴ Data[7] ≠ 55

⟹ Since 40 < 55,

END has its value changed by END = MID - 1

ie New END = 7 - 1 ⇒ 6

It means that now we have search our element between index 1 to index 6 ie in left sublist.

BEG = 1 END = 6 ∴ MID = 3

Now Compare 40 with Data[3] i.e (30)

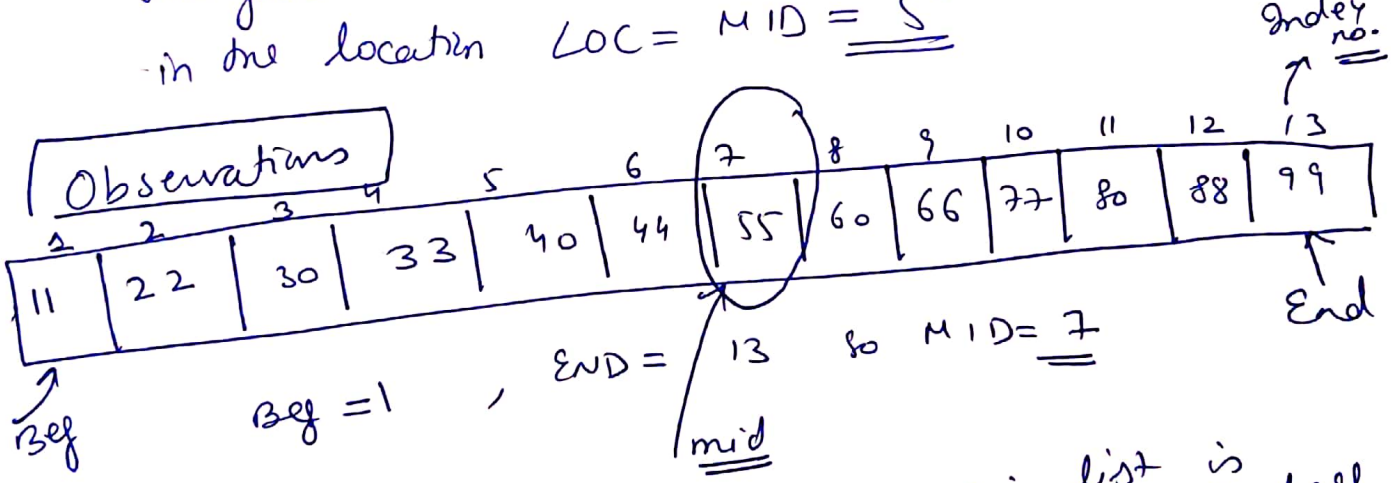
Since 40 > 30, Beg has its value changed. by Beg = MID+1 ⇒ 3+1 ⇒ (4)

Beg = 4 END = 6 ∴ MID = 5

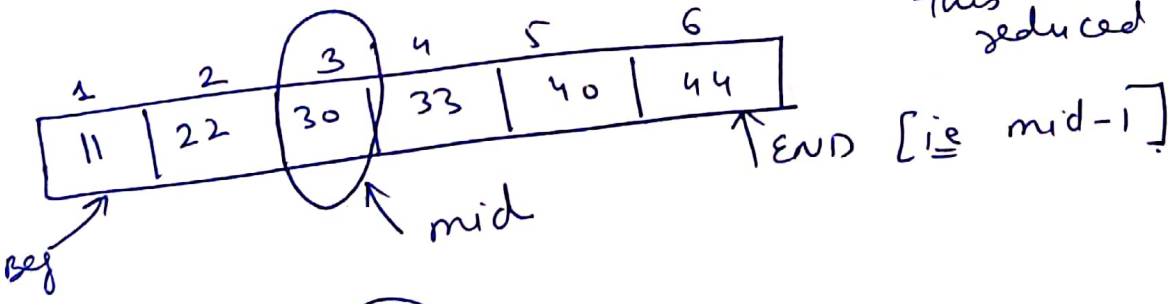
since Data[MID] ≠ 40

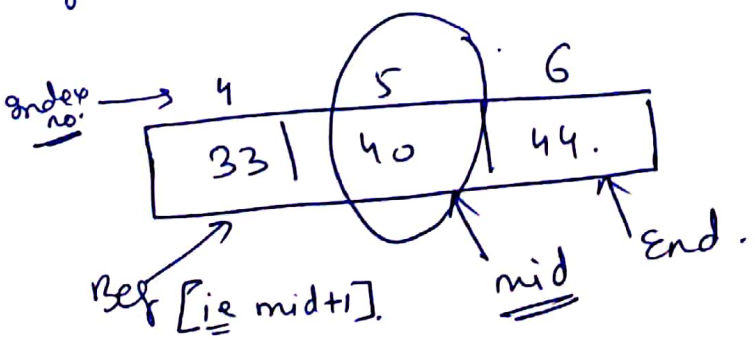
Since 40 = 40

therefore we have found item or element in the location LOC = MID = 5



This list is reduced to half



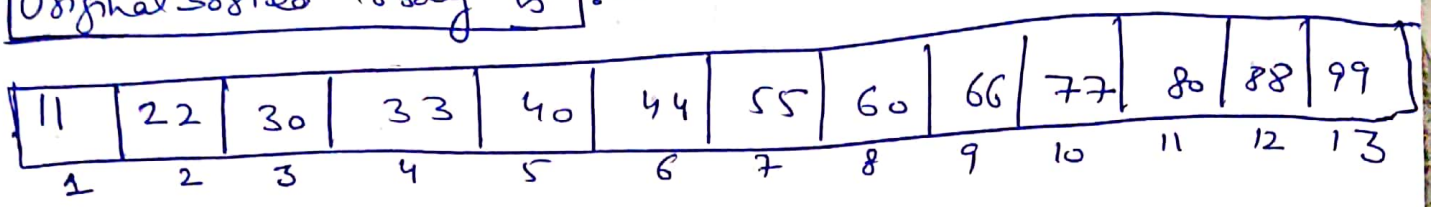


Again the list is reduce to half.

So we find the element with location as (5)

Say we want to search item = 85

Original sorted array is :-



Beg = 1 END = 13 ∴ MID = 7

data[MID] = 55

Compare 85 > 55

Beg has its value changed ie MID+1 ⇒ 8

Beg = 8 END = 13 ∴ MID = 10

Data[MID] = 77

Compare 85 > 77 so Beg = MID+1 ⇒ 11

Beg = 11 END = 13 MID = 12

data[MID] = 88

Compare 85 < 88 so END is changed

& New END = MID-1 ⇒ 11

Beg = 11 END = 11 MID = 11

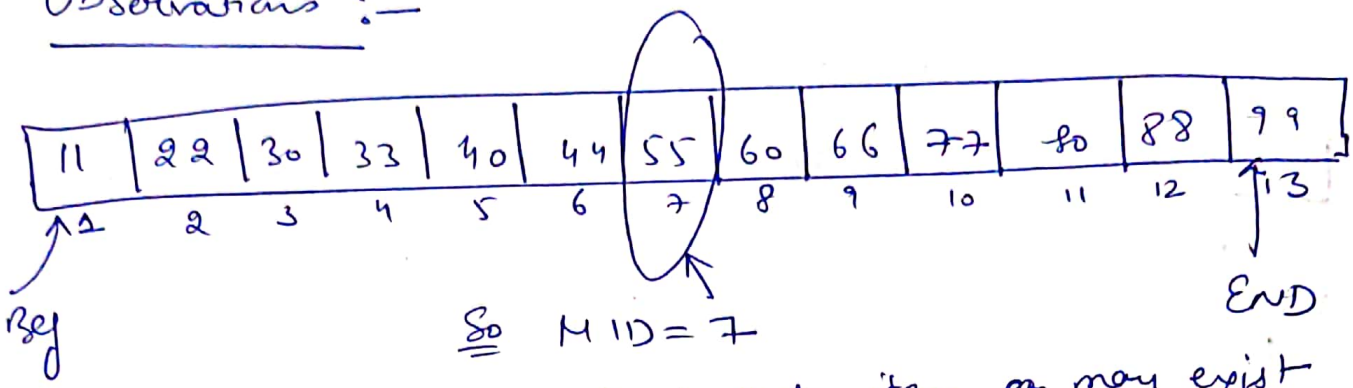
data[MID] = 80.

Compare 85 > 80 ∴ new Beg = mid+1 ⇒ 12

when Beg > END it means item does not exist in the list

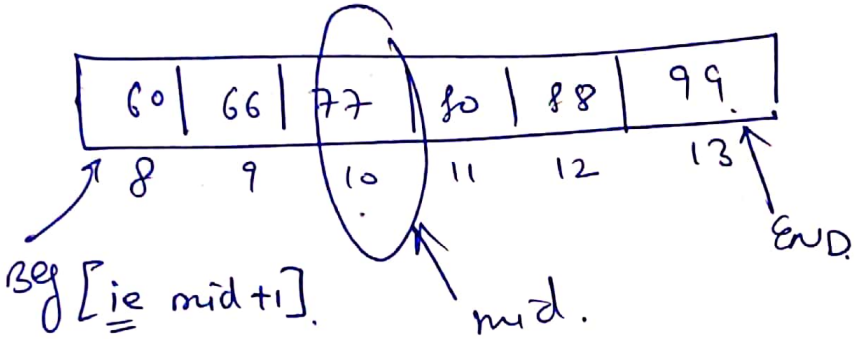
as Beg = 12 & END = 11

Observations :—

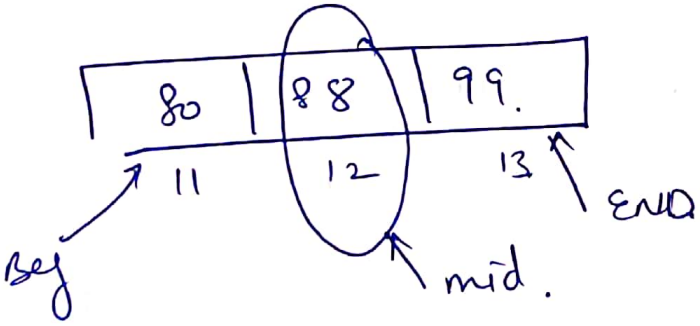


So MID = 7

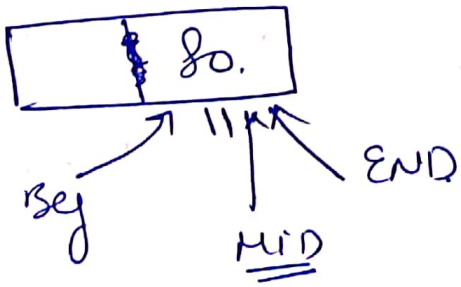
After comparison we found that item may exist in right sublist.



This list is reduced to half of the original one.



Again the list is reduced to half.



ie Beg = END

After comparison we found that Beg = mid+1

ie Beg = 12

& END = 11

so Beg > End it means data doesnot exist.

Time taken by Binary search

As we had observed that if we have n elements, each comparison reduces the sample size (or you can say size of array) in half.

$\therefore \approx O(\log_2 n)$

You can also compare the working of Binary search with the following loop.

~~for(i=0; i<=n; i=i*2)~~

```
for (i = n ; i >= 1 ; i = i/2)
{
≡
}
```

if we take $n = 1024$

$2^{10} = 1024$.

$\log_2 1024 = 10$

or $\log_2 n = 10$; iterations.

Limitations :-

1. List must be sorted.
2. One must have direct access to the middle element in any sublist.
3. Keeping data in sorted array is normally expensive when there are many insertions.

Algorithm.

Binary (Data , LB , UB , Item , LOC)

- Data → name of array
- LB → lower bound of array
- UB → upper Bound
- Item → to be searched.
- LOC → return the location of item if found.

1. Set Beg = LB , END = UB
   MID = INT((Beg + END)/2)

2. Repeat step 3 & 4 while Beg ≤ END && Data[MID] ≠ item
   (Begin of loop Body)

3. if item < Data [MID] // compare
   Set END = MID-1
   Else
   Set Beg = MID+1

4. MID = INT((Beg + END)/2).
   (End of loop Body.)

5. if Data [MID] = item // compare
   Set LOC = MID
   Else
   Set LOC = NULL

6. EXIT