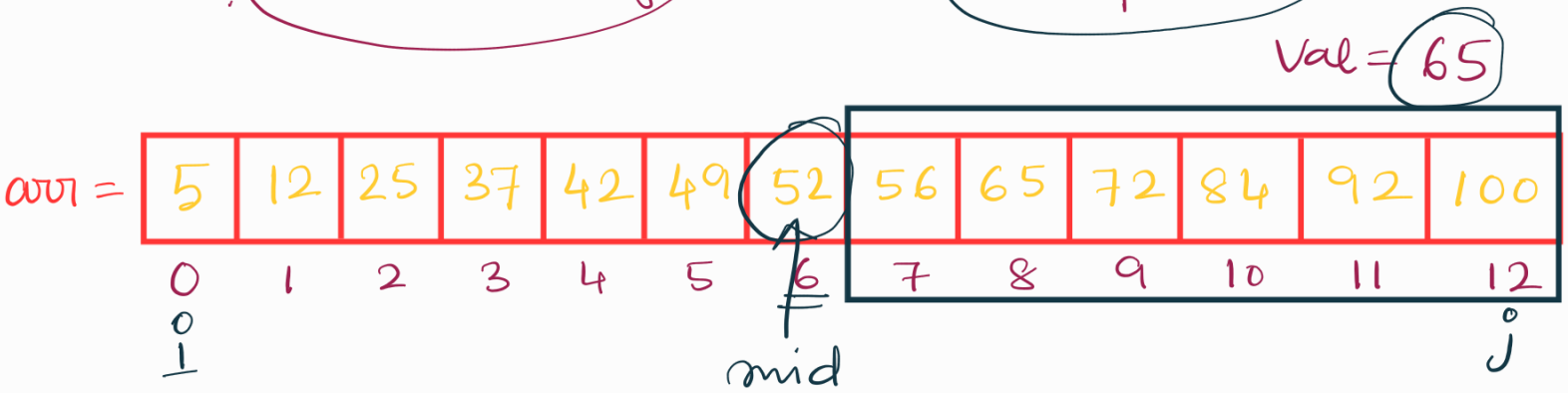
Binary Search

A. Sorted Array

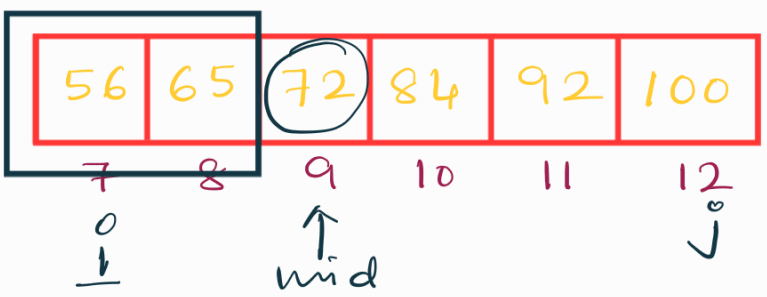
B. Two pointer



$mid = \frac{i+j}{2}$

$= \frac{0+12}{2}$

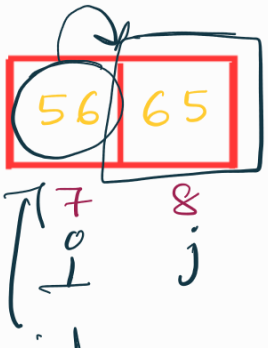
$= 6$



$mid = \frac{i+j}{2} = \frac{7+12}{2} = \frac{19}{2}$

$= 9$

```
if(arr[mid]<val)
  i = mid+1;
else if(arr[mid]>val)
  j = mid-1;
else
  return mid;
```

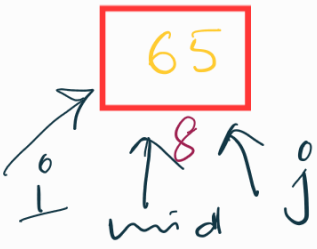


$mid = \frac{7+8}{2}$

$= \frac{15}{2}$

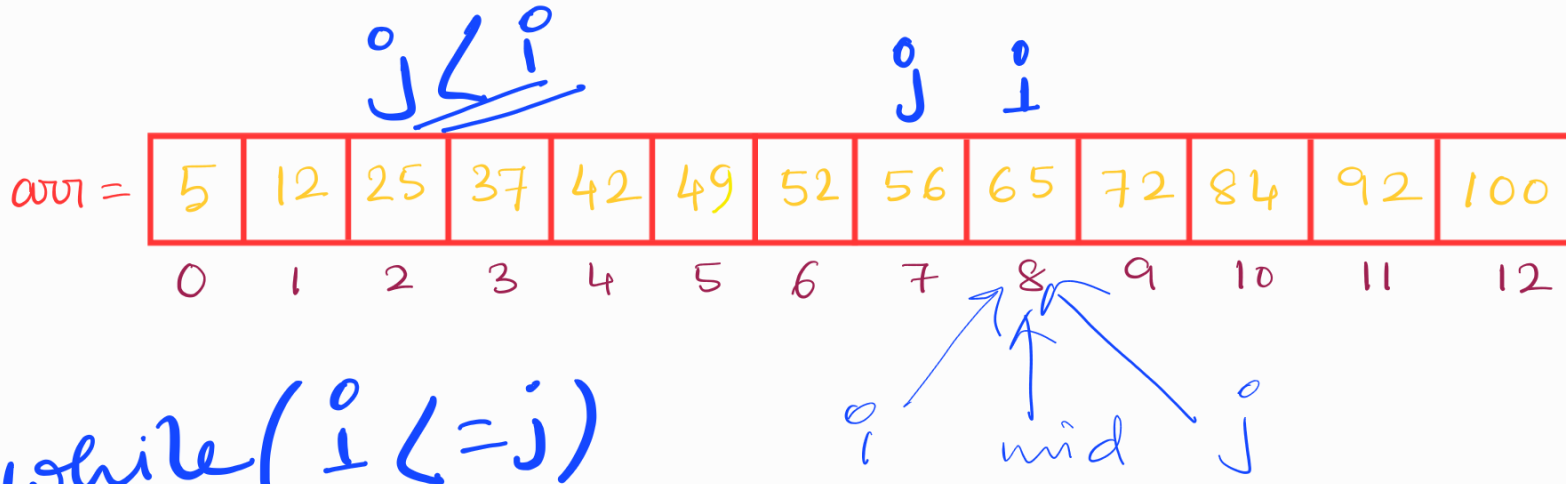
$= 7$

```
int mid=(i+j)/2;
if(arr[mid]==val)
  return mid;
else if(arr[mid]<val)
  i = mid+1;
else
  j = mid-1;
```



$mid = \frac{8+8}{2}$

$= 8$




```
while(i<=j)
{
  int mid=(i+j)/2;
  if(arr[mid]==val)
    return mid;
  else if(arr[mid]<val)
    i=mid+1;
  else
    j=mid-1;
}
return -1;
```

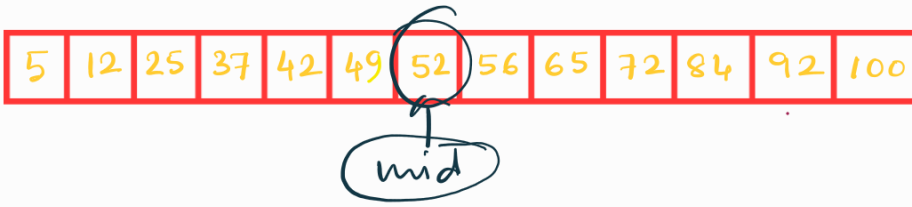
val = 60

65 < 60 ✗

Space and Time Complexity

```
int BiSearch(int a[],int n,int val)
{
   int i=0, j=n-1;
   while(i<=j)
   {
      int mid = (i+j)/2;
      if(a[mid]==val)
      return mid;
      else if(a[mid]<val)
      i=mid+1;
      else
      j=mid-1;
   }
   return -1;
}
```

- Space Complexity : O(1)
- Time Complexity :-



- Best Case : O(1)

: Average and Worst Case

O(log n)

O(k) ⇒

$\frac{n}{2^k} = 1$

$n = 2^k$

$\log n = \log(2^k)^{(k)}$

$\log n = k \log 2$

$\log n = k$

n

$n/2 \rightarrow n/2^1$

$n/4 \rightarrow n/2^2$

$n/8 \rightarrow n/2^3$

$n/16 \rightarrow n/2^4$

$n/\ldots = 1 \Rightarrow n/2^k = 1$

$k = \log n$

O(log n)