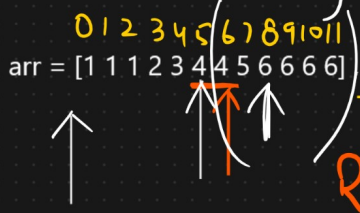
Searching (Assignment Solutions)



Last occurrence

Result = 6

Sorted array

time Complexity → $O(\log_2 n)$

n = 12

low = ~~0~~ 6 7

high = ~~11~~ 7 ~~6~~ 6

mid = ~~5~~ ~~8~~ ~~6~~ 7

result = ~~8~~ 6

```
binarySearch(arr, low, high, target){
    result = -1;
    while(low <= high){
        mid = low + (high-low)/2;
        if(nums[mid] == target){
            result = mid;
            low = mid+1;
        }
        else if(nums[mid] < target){
            low = mid+1;
        }
        else{
            high = mid-1;
```

Last occurrence of target value

Last occurence of target element

Traverse right side

Traverse right side

```
}
```

→ Traverse left side

```
}
return result;
}
```

Binary Search

3

0 1 2 3 4 5 6 7 8 9
arr = [0 0 0 0 1 1 1 1 1 1]
↑

low = 0 → 2 → 3 → 4, high = 9 → 3

mid = 4 → 1 → 2 → 3

n=10

false ← while (low<=high){

```
binarySearch(arr, low, high){
  while (low<=high){
    mid = low+(high-low)/2;
    if (arr(mid)==0)
      Right side traverse
      low= mid+1;
    else{
      Left side traverse
      high=mid-1;
  }
return (n-low);
```

4

10 - 4 = 6

↓

count of number of 1's

4

0 1 2 3 4 5 6 7 8 9

nums[] = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]

n = 10

target = 5

firstOcc = 1

lastOcc = 3

Result = 3

frequency = lastOcc − firstOcc + 1

= 3 − 1 + 1

= 3

5

num = 16

low = 0, high = 8 (num/2);

mid = 4

```
while(low <= high){
    if (mid * mid == num){
        return true;
    }

    else if (mid * mid < num){
        // Right side
        low = mid + 1;
    }

    else {
        // Left side
        high = mid - 1;
    }
}
return false;
}
```