In terms of time complexity Binary search is better approach as compared to Linear search

# Binary Search

Binary search is based on Divide and Conquer approach where we are splitting the search space in 2 equal parts until we find the element that needs to be searched.

↳ Divide & conquer

1) Sorted array

For applying the BS we need a sorted array

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 2 | 4 | 8 | 12 | 20 | 25 | 50 | 70 |

$x = 50$

$i = 0$

$J = 7$

**Small Problem** → Single element

$x = 50 \longleftrightarrow [50]$ — only one condition

```
if arr(i) == x :
    return i
return -1
```

**Big Problem** →

As a step 2 find the mid of the given array. Mid can be obtained using 2 approaches:
1. (i+j)//2 where i and j are lower and upper bounds respectively. Avoid using this approach since in case array size is huge then it will result in overflow condition.

2. i + (j-i)//2 Always use this approach since this will overcome the overflow condition.

overflow

$mid \Rightarrow (i+J)//2 = (0+7)//2$

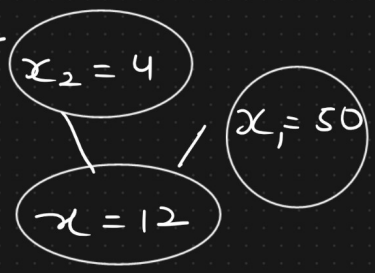
$= 3$

✓

$\Rightarrow i + (J-i)//2$

$(0+7)//2$

$= 3$




| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 2 | 4 | 8 | 12 | 20 | 25 | 50 | 70 |

→ Sorted

BinarySearch(arr, 0, 7, x)

BinarySearch(arr, i, J, x)

1. mid = 3

i <= J

2. arr(mid) == x :

As a third step if middle element is equal to the element to be searched then the element is found else go to right or left half based on the below condition. Also keep on doing this recursively (dividing into 2 half and going to right or left section for searching) until element to be searched is found.

return mid

## Psuedocode

element to search

12 < 50

arr(mid) < x

Recursion

Search the right half where starting index is mid+1 and ending index is j

BinarySearch(arr, mid+1, J, x)

OR

i = mid + 1

12 > 4

Search the left half where starting index is i and end index is mid-1

arr(mid) > x

Recursion

BinarySearch (arr, i, mid-1, x)

OR

J = mid - 1

return -1

In Binary Search we are dividing the search space into 2 halves followed by checking whether mid is equals to the element to be searched. If yes then break out from the loop else then we are deciding whether to go on left side or the right side. This is how we are reducing the search space and that is the reason why it is treated as an better approach when compared with linear search.

For Eg; Searching meaning of a word in Dictionary, Searching contact info of any person in Phonebook.