

Binary Search

→ Search a sorted array by repeatedly dividing the search interval in half

→ Begin with an interval covering the whole array

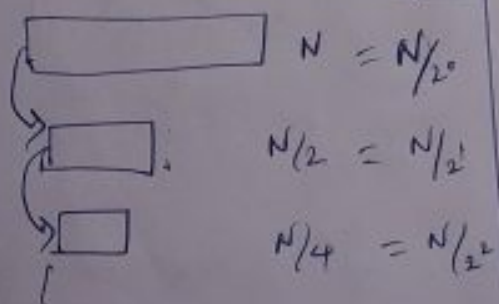
→ If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half

→ Otherwise, narrow it to the upper half

→ Repeatedly check until the value is found or the interval is empty

Worst case:

Time complexity = $O(\log n)$



$$1 = N/2^k = N/2^k$$

$$\frac{N}{2^k} = 1 \Rightarrow N = 2^k$$

Take log

$$\log N = \log 2^k$$

$$\log N = k \log 2$$

$$k = \frac{\log N}{\log 2}$$

$$k = \log_2 N$$

Steps: Best case: $O(1)$

1) Find the middle element

2) If target > middle
search in the right

else if target < middle
search in the left

else
answer // got target

Better way to find middle

$$1) m \neq (\text{start} + \text{end}) / 2$$

$$2) m = \text{start} + \frac{(\text{end} - \text{start})}{2}$$

$$\Rightarrow m = s + \frac{e - s}{2}$$

$$= s + \frac{e - s}{2}$$

$$m = s + e/2$$

Program

```
public class BinarySearch {
    public static void main(String[] args) {
        int[] arr = {-18, -12, -4, 0, 3, 3, 4, 15, 16, 18, 22, 45, 55};
        int target = 22;
        int ans = binarySearch(arr, target);
        System.out.println(ans);
    }
}
```

```
static int binarySearch(int[] arr, int target) {
    int start = 0;
    int end = arr.length - 1;
    while (start <= end) {
        mid = start + (end - start) / 2;
```

if (target < arr[mid])
end = mid;

else if (target > arr[mid])
start = mid;

else
return mid;

3
return -1;

3

3

Order & guess

If the array is descending, we guess

Agnostic BS

Steps

1) Find the middle element

2) If target < arr[mid]

search in the left

else if target > arr[mid]
search in the right

else
answer

eg:

public class Order

public static void main

int[] arr = {99, 88, 77, 66, 55, 44, 33, 22, 11, 0, -1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11, -12, -13, -14, -15, -16, -17, -18, -19, -20, -21, -22, -23, -24, -25, -26, -27, -28, -29, -30, -31, -32, -33, -34, -35, -36, -37, -38, -39, -40, -41, -42, -43, -44, -45, -46, -47, -48, -49, -50, -51, -52, -53, -54, -55, -56, -57, -58, -59, -60, -61, -62, -63, -64, -65, -66, -67, -68, -69, -70, -71, -72, -73, -74, -75, -76, -77, -78, -79, -80, -81, -82, -83, -84, -85, -86, -87, -88, -89, -90, -91, -92, -93, -94, -95, -96, -97, -98, -99, -100};

10, 5, 2, 1

int target = 22

int ans = order

System.out.println

3

case: $O(N)$

middle element

middle

in the right

$t < \text{middle}$

in the left

// got target

find middle

$(start + end) / 2$

$(end - start) / 2$

$2 - 1/2$

$2 - 1/2$

$1/2$

arraySearch{

main(String[] args){

12, 4, 0, 3, 3, 4, 15,

16, 18, 22, 45, 89

2;

arraySearch(arr, target);

println(ans);

arraySearch(int[] arr, int target){

arr.length - 1;

$start \leq end$){

$start + (end - start) / 2$;

if (target < ^{arr}mid[mid])

~~mid~~ = end = mid - 1;

else if (target > arr[mid])

start = mid + 1;

else

return mid;

}

return -1;

}

}

Order Agnostic BS

If the array is sorted in descending, we go for order Agnostic BS

Steps:

1) Find the middle element

2) If target < middle

search in the right

else if target > middle

search in the left

else

answer // got target

eg:

public class OrderAgnosticBS{

public static void main(String[] args){

int[] arr = {99, 80, 25, 22, 11,

10, 5, 2, -3};

int target = 22;

int ans = orderAgnostic(arr, target);

System.out.println(ans);

}

static int OrderAgnostic(int[] arr, int target){

int start = 0;

int end = arr.length - 1;

boolean isAsc = arr[start] < arr[end];

// true → ascending

// false → descending

while (start <= end){

int mid = start + (end - start) / 2;

if (arr[mid] == target){

return mid;

}

if (isAsc){

if (target < arr[mid])

end = mid - 1;

else

start = mid + 1;

}

else{

if (target > arr[mid])

end = mid - 1;

else

start = mid + 1;

}

return -1;

}