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)

$N = N/2^0$

$N/2 = N/2^1$

$N/4 = N/2^2$

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

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

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) $m = (start + end)/2$

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

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

$= 2s + e - s/2$

$m = s + e/2$

Program

```
public class BinarySearch{
  public static void main(String[] args){
    int[] arr = {-18, -12, -4, 0, 3, 3, 4, 15,
                 16, 18, 22, 45};
    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 <
    mid = end =
else if (target >
    start = mid
else
    return
}
return -1;
}
}
```

Order Agnost

If the array is descending, we g

Agnostic BS

Steps:

1) Find the m
2) If target <
   search
   else if target
   search
   else
   answer

eg:

```
public class Order
  public static void
  int[] arr = {99,
               10, 5, 2,
  int target = 22
  int ans = order
  System.out.pri
}
```

case: O(1)

dle element

middle
in the right
t < middle
in the left

// got target

find middle

t+end)/2

(end-start)
2

e-s/2
2

arySearch{
d main(String[] args){
-12, -4, 0, 3, 3, 4, 15,
16, 18, 22, 45, 89
2;
arySearch(arr, target);
println(ans);

aySearch(int[] arr, int
target){
r.length-1;
<=end){
t+(end-start)/2;

if (target < ~~arr~~ arr[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;
}
}
```