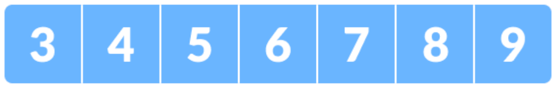
Binary Search

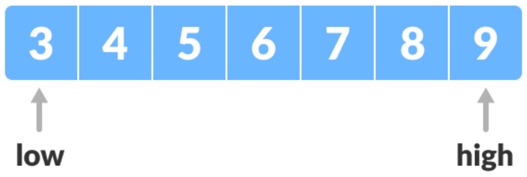
Binary Search is an implementation method for several programming languages in searching for data in array element positions that have been arranged using Iterative and Recursive methods. In Binary Search, always look for 'in the middle' of an existing array position.

In searching for Binary Search, you need to know how Binary Search works.

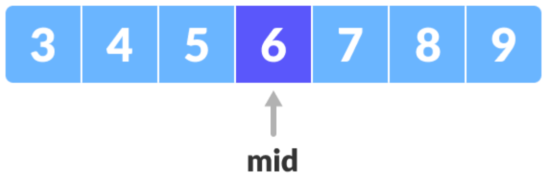
- Can be seen in the following initial array:



- Determine that x (the data we want to find) is the number 4, [x == 4]. Then determine the data in the highest and lowest positions

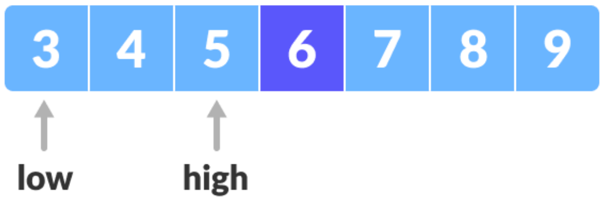


- Then find the 'middle' side in the data array. This can be searched by typing 'arr[(low+high)/2] = 6' (an array where finding the 'n' data from the highest data and the lowest data is divided into two and finding the 'n' data in the middle is 6)

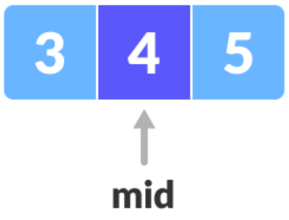


- Then equate the existing data, before we looked for x as 4 [x==4] not 6. Then what do we do next? If 6 cannot be the equation as the x we are looking for, then we need to determine which data we need to look for is where? Left side (lower from6)? Or the right side (higher than 6)?
- Define 'if x > mid', compare x with the right side of the mid data (6). Because the data is sequential, we can set a rule where 'low = mid+1'

- Determine 'else x < mid', compare x with the left side of the mid data (6). Because the data is sequential, we can set a rule where 'high = mid-1'. If else is a true argument, then we need to determine the higher-lowest data again to find the 'mid' data



- - Then it will find data where [x == 4]



For the difference between Interative and Recursive methods can be found in the following code:

Iterative Method

```
do until the pointers low and high meet each other.
    mid = (low + high)/2
    if (x == arr[mid])
        return mid
    else if (x > arr[mid]) // x is on the right side
        low = mid + 1
    else                       // x is on the left side
        high = mid - 1
```

Recursive Method

```
binarySearch(arr, x, low, high)

    if low > high

        return False

    else

        mid = (low + high) / 2

        if x == arr[mid]

            return mid

        else if x > arr[mid]        // x is on the right side

            return binarySearch(arr, x, mid + 1, high)

        else                               // x is on the left side

            return binarySearch(arr, x, low, mid - 1)
```

- The difference lies in the Iterative method of dividing two data to find the data in the middle of the array and comparing which data you want to find with the larger or smaller mid data (low = mid+1 or high = mid-1), this will do a loop and using FOR, WHILE, DO-WHILE logic
- The difference in the Recursive method of dividing two data to find the data in the middle of the array and 'return binarySearch' by entering some conditions (arr, x, mid+1, high) or (arr, x, mid -1, low) and this will repeating the data separately
- Time Complexity:
    - Best Case Complexity O(1)
    - Average Case Complexity O(log n)
    - Worst Case Complexity O(log n)
    - Space Complexity O(1)