Topics: Binary Search
Using Iteration and Recursion

Searching

Searching an unsorted array is difficult and often a linear process. It may require scanning each item of the array, from first to last, to find the desired item. This is called sequential search, and can be done by via a for or while loop. However, if the array or database is large, this can take a lot of time. Since we are usually searching for information most of the time, we would like searching to be a quick and efficient operation.

But if we were to pass the array through one of our Sorting algorithms, we could then use an interesting heuristic for searching it efficiently – Binary Search.

The algorithm for Binary Search is simple:

```
def binarySearch(array, target){

     min=0
     max = len(array)-1

     while(min <= max){

          // find middle index halfway between min and max
          mid = (min + max) / 2

          // check if target found
          if array[mid] == target:
               return mid

          else if target > array[mid]:
               min = mid + 1

          else: //target < array[mid]
               max = mid - 1

     }

}
```

This has a very simple logic.

1. Take a sorted array.

2. Split it in half, into a 'lower' half and 'upper' half.

3. Check if the value at halfway mark is the actual target value. If so, found it! Return the index position of the halfway mark where it was found.

4. If it's not, check if target value could possibly be in 'lower' half or 'upper half.
   a. If target is less than value at halfway point, it must be in lower half,
   b. If target is greater than value at halfway point, it must be in the upper half.
5. If the outcome is 4a, then make the current search area the lower half of the array and disregard the upper half. If the outcome is 4b, then make the current search area the upper half of the array and disregard the lower half

6. Repeat until Step 3 finds the target value, or until the array is too small to split.

7. If the item is not found, return -1.

Let's practice on an example.

For the following array, let's write out the mid, min, and max values per each loop in the binary search algorithm, until the target value of 37 is found.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
11	18	29	37	42	49	51	63	69	72	77	82	88	91

```
min = 0, max = 13

loop 1) mid = (0+13)/2 = 6. target < mid, so max = mid-1 = 5.

loop 2) mid = (0+5)/2 = 2. target > mid, so min = mid+1 = 3.
loop 3) mid = (3+5)/2 = 4. target < mid, so max = mid-1 = 3.

loop 4) mid = (3+3)/2 = 3. target == mid, so return 3.
```

Q1) Repeat, this process of writing out min, max, max and return values this time for the target value of 11, which is the very first item and for an item that is not in the array e.g. 100.

Recursive Binary Search

Now, what if we wanted to convert Binary Search to its recursive version?

Well, the repetitive operation is splitting the array in half, and comparing target to the halfway value. The base case is that either the item is found at the midway mark or the current segment of the array is too small to be further broken into half. The smaller case is focusing the search on either the upper or lower half of the array.

Here is the pseudocode for the recursive version of Binary Search.

```
def binarySearch(array, target, min, max){

     // base case
     if min > max:
          return -1

     // recursive case
     mid = (min + max) / 2

     // check if target found
     if array[mid] == target:
          return mid

     // continue on upper half
     else if target > array[mid]:
          binarySearchRecursive(array,target, min=mid+1, max)

     // continue on lower half
     else: //target < array[mid]
          binarySearchRecursive(array,target,min, max=mid-1)
}
```

Q 2) Convert this pseudocode into Java code