Binary Search

The element half way into the array is first compared with the element to find,

If this element is bigger than what we want to find, we wet a new upper bound for our search and half the difference between the lower and upper bound and subtract that from the index and search with that new index.

If the element was smaller we set a new lower bound for the search and add the same half difference.

The recursiveness of the function is in calling the search function with a lower or higher index.

public int getIndex(int searchFor)

{

this.searchFor = searchFor;

return search(size / 2);

}

private int search(int i)

{

if (i > lowerBoundIndex && i < upperBoundIndex)

{

int element = array.get(i);

if (element == searchFor)

{

return i;

}

if (element > searchFor)

{

upperBoundIndex = i;

return search(i - ((upperBoundIndex-lowerBoundIndex) / 2));

}

lowerBoundIndex = i;

return search(i + ((upperBoundIndex-lowerBoundIndex) / 2));

}

return -1;

}

Since we half the search space at each recursive call, our theoretical efficiency is O(log(n).