Question 8.3 from CTCI: Magic Index

Benjamin De Brasi

February 17, 2018

Question

Magic Index: A magic index in an array A [1. .. n -1] is defined to be an index such that A[i] = i. Given a sorted array of distinct integers, write a method to find a magic index, if one exists, in array A.

FOLLOW UP

What if the values are not distinct?

Explanation and Algorithm

Here are two ways to solve the non distinct values:

The first is brute force. Just go through each index and see if A[i] == i till end of the array.

A better way is to use binary search where the value you are trying to mathc is the mid index itself on each call rather than a target given as an argument. You can totally elminate half of the array each time because if the array value at a given mid is smaller than the index then you know half of the array must contain values smaller than the array and the same is true for bigger values if the mid is bigger than the index.

Things become a bit tricker if there are no uniqueness of numbers are not guaranteed, but the approach is similar to binary search. In this case, you can only eliminate parts of the array each run where there is not a match and that is not out of bounds. For the left side of the array, you can elminate the part of the array from the min value of the middle index and the value of that index to the 0th index. For the right, you can eliminate from the max value of the middle index and the middle index itself to the end of the array. Search through the one side till your each the end and then the other till the end cutting out sections that you can eliminate each time until you find a magic index or run past the 0th and length of array indicies.

Hints

1. Start with a brute force algorithm.
2. Your brute force algorithm probably ran in O(N) time. If you're trying to beat that runtime, what runtime do you think you will get to? What sorts of algorithms have that runtime?
3. Can you solve the problem in O(log N)?
4. Binary search has a runtime of O(log N) . Can you apply a form of binary search to the problem?
5. Given a specific index and value, can you identify if the magic index would be before or after it?

Code

```
/*Answer 1 */

int magicFast(int[] arr){
   return magicFast(arr, 0, arr.length-1);
}

int magicFast(int[] arr, int start, int end){
   if (end < start){
      return -1;
   }

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

   if(arr[mid] == mid){
      return mid;
   }
   else if(arr[mid] > mid){
      return magicFast(arr, start, mid -1);
   } else{
      return magicFast(arr, mid+1, end);
   }
}
```

```
/* Answer 2 */

int magicFast(int[] arr){
  return magicFast(arr, 0, array.length-1);
}

int magicFast(int[] arr, int start, int end){
  if(end<start){
    return -1;
  }
  int midIndex = start (end-start)/2;
  int midValue = arr[midIndex];
  if(midValue == midIndex){
    return midIndex;
  }

  int leftIndex= Math.min(midIndex-1,midValue);
  int left = magicFast(arr, start, leftIndex);
  if (left >= 0){
    return left;
  }

  int rightIndex = Match.max(midIndex+1,midValue);
  int right = magicFast(arr, rightIndex, end);

  return right;
}
```

Big O analysis

The brute force has a O(n) run time and the binary search one has O(logn) run time. The solution for the nondistinct values is almost O(logn).

Sources

Question, answer and other material taken from Cracking the Coding Interview 6th edition by Gayle Laakmann McDowell.