Question 16.17 from CTCI: Contiguous Sequence

Benjamin De Brasi

February 7, 2018

Question

Contiguous Sequence: You are given an array of integers (both positive and negative). Find the contiguous sequence with the largest sum. Return the sum.

Explanation and Algorithm

The keys to solving this problem is figuring out the right way to look at the array and how to keep running sum while keeping track of the running sum and the max sums.

First, let's cover the how to look at the array. We know the array consists of ints. That means the numbers can be positive, negative or 0. This is obvious, but important information. This divides the problem into 3 cases. Let's imagine for a moment we were guaranteed all the numbers in the array were positive. That would mean the max sum would always be the sum of the entire array. We can easily see that if the array was all 0s or mixtures of positive numbers and 0, the same would be true. So the only possible complications occur when we encounter a negative number.

When we encounter a negative number we then need to decide whether it should be including in the current sum or not. It should be included in the sum if there exists a positive segment before and after the negative number which when summed together have a value greater than either segement alone after subtracting the value of the negative segment. Otherwise, the bigger of the positive segments will be the max sum. The only other case to think about when encountering a negative number is if the entire array consists of negative values in which case we want the smallest negative number. Using all of this we can code the logic with 2 if statements within a loop traversing each value in the array keeping a running sum variable and max sum variable.

The first if statement should just check if the current sum you are adding up is

greater than the current max. If that is the case make max = sum. The second if statement should check if the current sum is less than 0. If that is the case then you should set the sum to 0. Doing this covers the case where all numbers are negative and allows you to get the max as the smallest negative number or the case where a negative number breaks two positive segments and you have to discern which part of the array contains the biggest sum and output it.

Hints

1. Picture the array as alternating sequences of positive and negative numbers. Observe that we would never include just part of a positive sequence or part of a negative sequence.
2. Observe that if you have a sequence of values which have a negative sum, those will never start or end a sequence. (They could be present in a sequence if they connected two other sequences.)
3. Start from the beginning of the array. As that subsequence gets larger, it stays as the best subsequence. Once it becomes negative, though, it's useless.
4. If we tracked the running sum, we should reset it as soon as the subsequence becomes negative. We would never add a negative sequence to the beginning or end of another subsequence.
5. You can solve this in O(N) time and 0(1) space.

Code

```
/*Answer 1 */

int getMaxSum(int[] a){
   int maxsum = 0;
   int sum = 0;
   for(int i = 0; i < a.length; i++){
      sum += a[i];
      if(maxsum < sum)
         maxsum = sum;
      else if (sum < 0)
            sum = 0;
   }
   return maxsum;
}
```

Big O analysis

The Big 0 is O(n) because you only need to run through the array once and check a few if statements each time.

Sources

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