CS325 Homework 2

Kabir Kang, Paul Ely, Jason Dorweiler

May 4, 2014

Recursive Function

Describe the solution to the maximum subarray problem recursively and mathematically based on the above idea.

The given idea was: *The maximum subarray either uses the last element in the input array, or it doesn't.* For an array A with n elements, a recursive function to solve this problem would find the maximum value of all subarrays up to n. For each subarray, it would test whether the next element should be included in the maximum sum or whether it should not.

So, let A be an array with n elements. Then let MaxSum(k) be the maximum sum of all subarrays up to and ending at j. Then the maximum subarray value can be found by taking max(MaxSum(k-1) + A[k], A[k]).

Pseudocode - DP Solution

```
MaxSubArray(array):                                          1
    maxSum = -∞                                              2
    tempSum = 0                                              3
    for i in array:                                         4
     if tempSum > 0:                                         5
         tempSum = tempSum + i                               6
     else:                                                   7
         tempSum = i                                         8
     maxSum = np.maximum(maxSum, tempSum)                    9
    return maxSum                                           10
```

Listing 1: pseudo code for DP algorithm

Running Time

The running time for this algorithm is $\theta(n)$. The algorithm only loops from 0 to n once, performing a number of constant-time operations per iteration. It will always run from 0 to n, thus giving a $\theta(n)$ running time. The calculated running time for the dynamic algorithm based on the slope of our log-log plot is 0.98 which shows our assumptions are correct. The run time for the $n \log(n)$ algorithm came out out to 1.10 which we know from the previous assignment is correct.

Theoretical Correctness

MaxSubArray will return the maximum value contiguous subarray for some array A with n elements, where $n > 0$. It does so by saving the maximum subarray value and building up from 0 to n.

Base case: The base case is when $n = 1$. Then the value of the maximum subarray is $A[n]$.

Proof: Assume that the algorithm is correct for $1 \leq n \leq k$. Now consider adding another element to the array. There are two cases:

- Case 1: running sum is > 0. In this case, the next element is added to the running sum.

- Case 2: the running sum is $<= 0$. In this case, the running sum is set equal to the next element.

The maximum value is then set to the maximum of either the previous maximum or the running sum. By the inductive hypothesis, we know that the previous maximum sum was the maximum value for the subarray up to k. Therefore the algorithm correctly returns the maximum subarray.
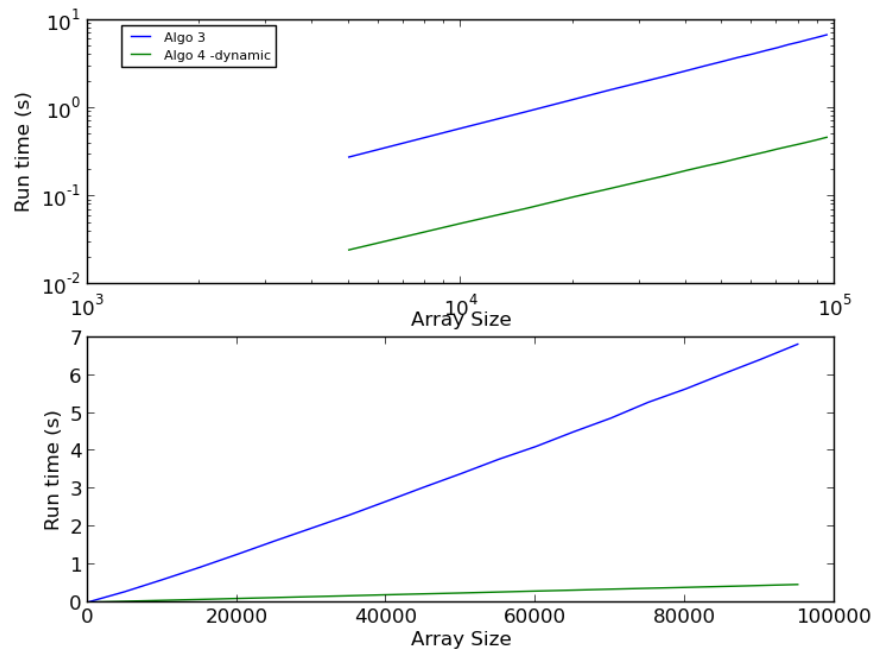
Figure 1: Plot of the dynamic algorithm (green) and the $n\log(n)$ algorithm from home-work1