

Assignment 2: Dynamic Programming project

Francis Vo, Soo-Hyun Yoo

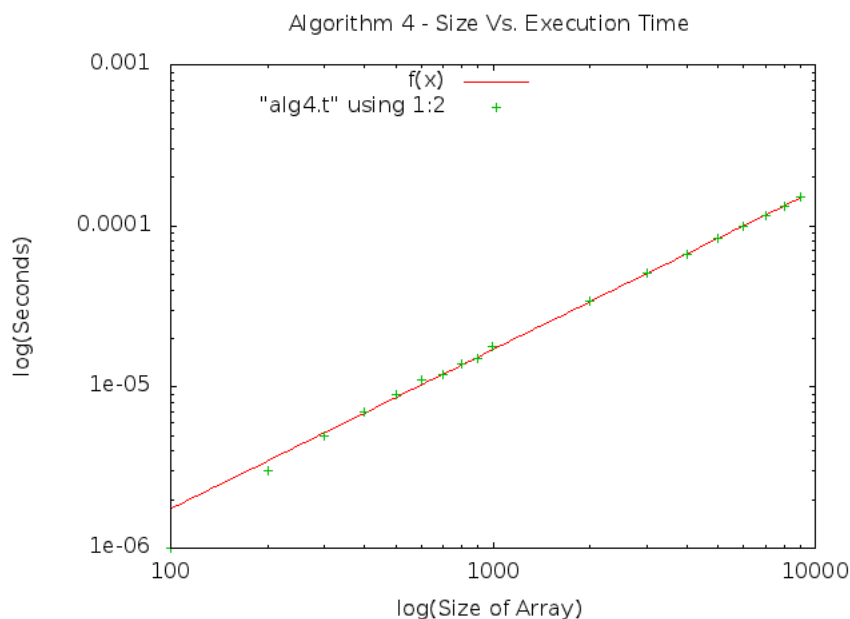
October 27, 2012

1 Recursive function

2 Pseudocode

3 Running time

Using the Log-Log plot will give us a good hint for the asymptotic run times. The slope for the graph is 0.987956, meaning that the asymptotic run times is around $\Omega(n)$. Then looking at the code, it only looks at each element once, making it $\Omega(n)$



4 Theoretical correctness

Induction Proof. $MS(k)$ will return the maximum subarray sum for the array $A[0 : k]$

Base case: If $n = -1$ then $max = current = 0$

Inductive Step: $maxSubarray(n - 1).current + A[n]$ or 0 is the current largest sum starting from the left

Proof:

Case if $A[n] > 0$ then $current = MS(n - 1).current + A[n] > MS(n - 1).current$.

This number might also be the max value. So $max = Greater(max, current)$

Case if $A[n] > -maxSubarray(n - 1)$ then $maxSubarray(n - 1) + A[n] < 0$ making the Null set greater.

$max = MS(n - 1).max$ and $current = 0$

Case else making $A[n]$ negative but $maxSubarray(n - 1) + A[n] \geq 0$

so it is still good to use for the next current: $current + A[n + 1] > A[n + 1]$

$max = MS(n - 1).max$ and $current = maxSubarray(n - 1) + A[n]$

$MS(n).max = max$ and $MS(n).current = current$

□

5 Implement

5.1 Algorithm 4

```
1  /*
2  * Enumeration
3  * Loop over each pair of indices i; j and compute the sum from k=i to j of a[k].
4  * Keep the best sum you have found so far.
5  */
6
7  using namespace std;
8
9
10 int MaxSubarray(int a[], int n){
11     int current = 0;
12     int max = 0;
13     int i;
14     for(i = 0; i < n; i++){
15         current += a[i];
16         if(current <= 0){
17             current = 0;
18         } else if(current > max){
19             max = current;
20         }
21     }
22     return max;
23 }
```

alg4.cpp

6 Test

7 Compare

Well, there is a huge difference as seen on the Compare Plot

