CS325 Homework 1

Kabir Kang, Paul Ely, Jason Dorweiler

April 27, 2014

Mathematical Analysis

```
if len(array) == 1:
     maxSum = array[0]
else:
     for e in range(len(array)):
          for j in range(e,len(array)):
               maxSum = np.maximum(maxSum, sum(array[e:j]))
```

Listing 1: pseudo code for n^3 algorithm

```
for e in range(len(array)):
     testSum = 0
     for j in range(e,len(array)):
          testSum += array[j]
          maxSum = np.maximum(maxSum, testSum)
```

Listing 2: pseudo code for n^2 algorithm

```
def algo3(array):
if(len(array) == 0):
     return 0
if(len(array) == 1):
     return array[0]

mid = len(array)/2
tempL = tempR = 0
maxLeft = maxRight = -99999

#left side crossing -- mid backwards
for i in range(mid,0,-1):
     tempL = tempL + array[i]
     maxLeft = np.maximum(maxLeft, tempL)

#right side crossing -- mid forwards
for j in range(mid+1, len(array)):
     tempR = tempR + array[j]
     maxRight = np.maximum(maxRight, tempR)
maxCrossing = maxLeft + maxRight

MaxA = algo3(array[:mid])
MaxB = algo3(array[mid+1:])

return np.maximum(np.maximum(MaxA, MaxB),maxCrossing))
```

Listing 3: pseudo code for $n \log(n)$ algorithm

Theoretical Correctness

Experimental Analysis

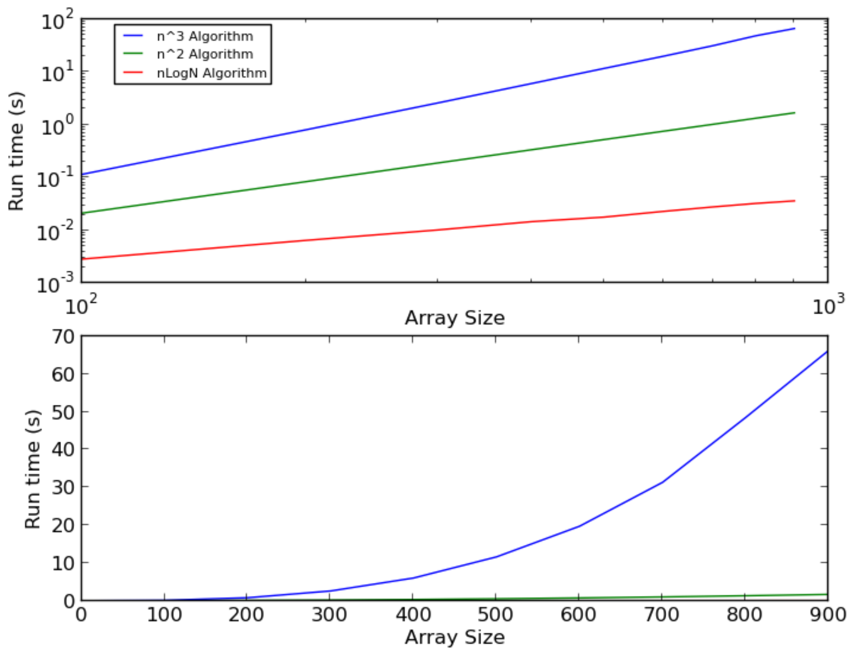

Figure 1: Plot of the three algorithms up to array size of 900, top: log/log, bottom: normal axis

A plot of the run times for the three algorithms are shown in figure 1. The results of the slope calculation for each algorithm are shown in the table below. The slope calculation fits with what we would expect. The n^3 algorithm has a slope 3, the n^2 algorithem has a slope 2, and the $n \log(n)$ algorithm has a slope a bit higher than 1.

Algorithm	Slope of log-log plot
n^3	2.89
n^2	1.99
$n \log(n)$	1.16

In the lower plot on figure 1 we have also plotted the three algorithms with a normal axis. This shows the n^3 run time of the first algorithm. The third algorithm runs so quickly that it doesn't even show up at this scale.

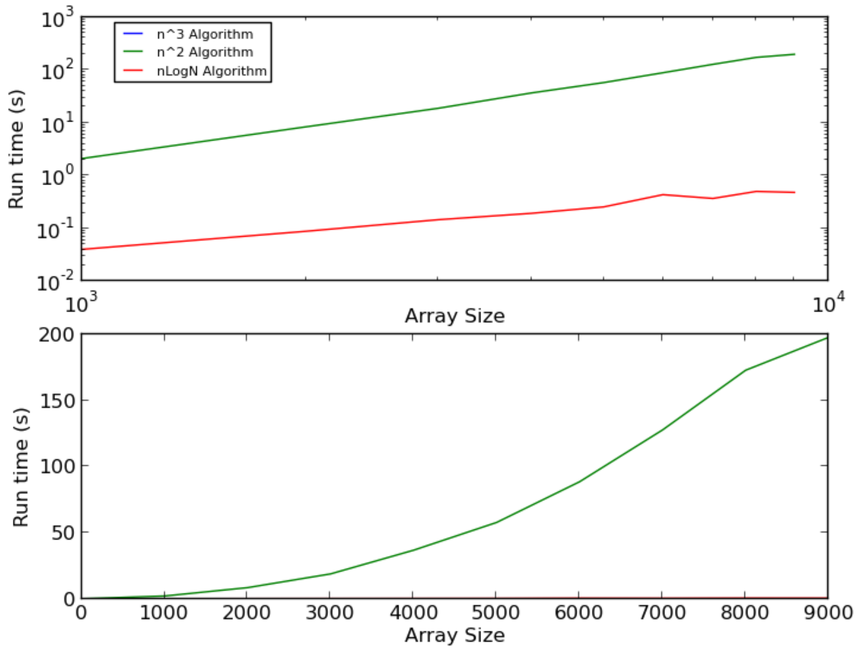

Figure 2: Plot of the three algorithms up to array size of 9000, top: log/log, bottom: normal axis

Extrapolation and Interpretation

What is the biggest instance that you could solve with your algorithm in one hour

From our experimental data and using the slope of our log-log plot we determined that the relationship between run time and array size for the $n\log(n)$ algorithm is approximately:

$$\log(time) = 1.16\log(elements) - 11.15$$

Using this equation and a run time of 1 hr (3600s) we get an approximate array size of 17.4 million elements

Determine the slope of the line in your log-log plot and from these slopes infer the experimental running time for each algorithm. Discuss any discrepancies between the experimental and theoretical running times