FractionalKnapsack Growth Rate Investigation
CS4310 - Design & Analysis of Algorithms
Spring 2017
John Harvey
04/09/2017

Hypothesis:
The FractionalKnapsack algorithm has a time complexity of O(n logn)

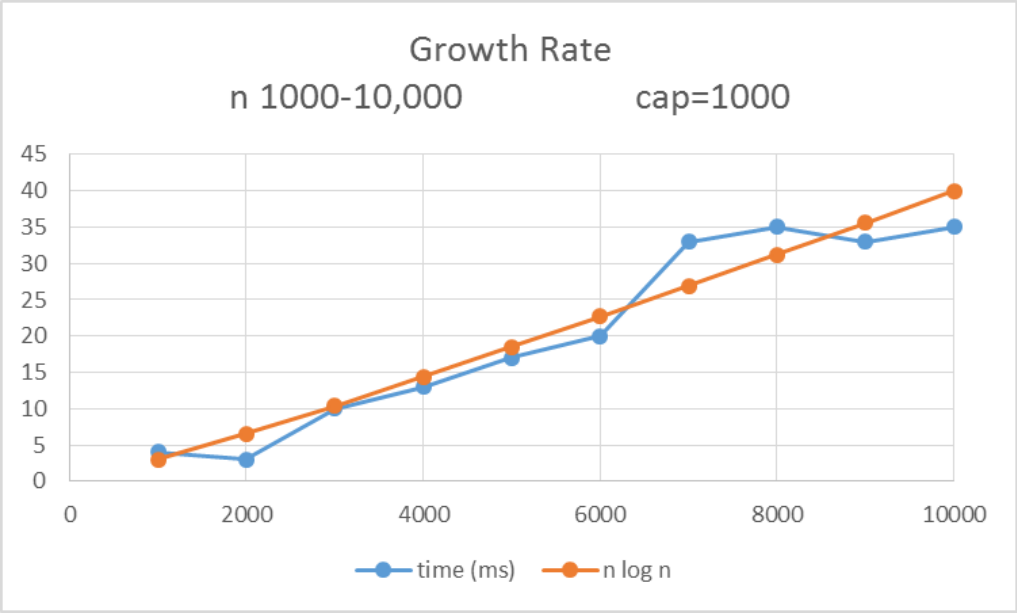
Test Design:
In order to test the hypothesis involved, I have taken the following steps:

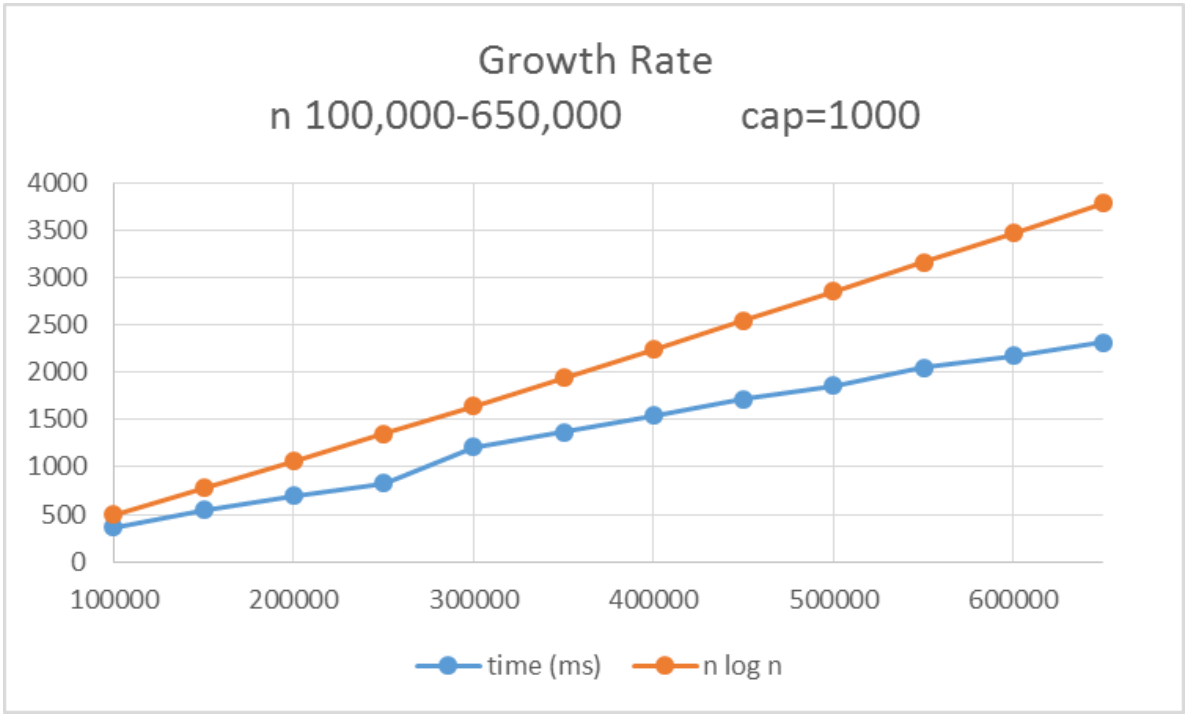
1) Implement the FractionalKnapsack algorithm in Ruby
2) Run the code for multiple amounts of n items
   a) Test with various maximum capacities
   b) Analyze all data on individual graphs
   c) Analyze all data on a single graph
3) Inspect the graphs and compare the growth to an n log n growth rate
4) Examine code and make changes if results do not seem right

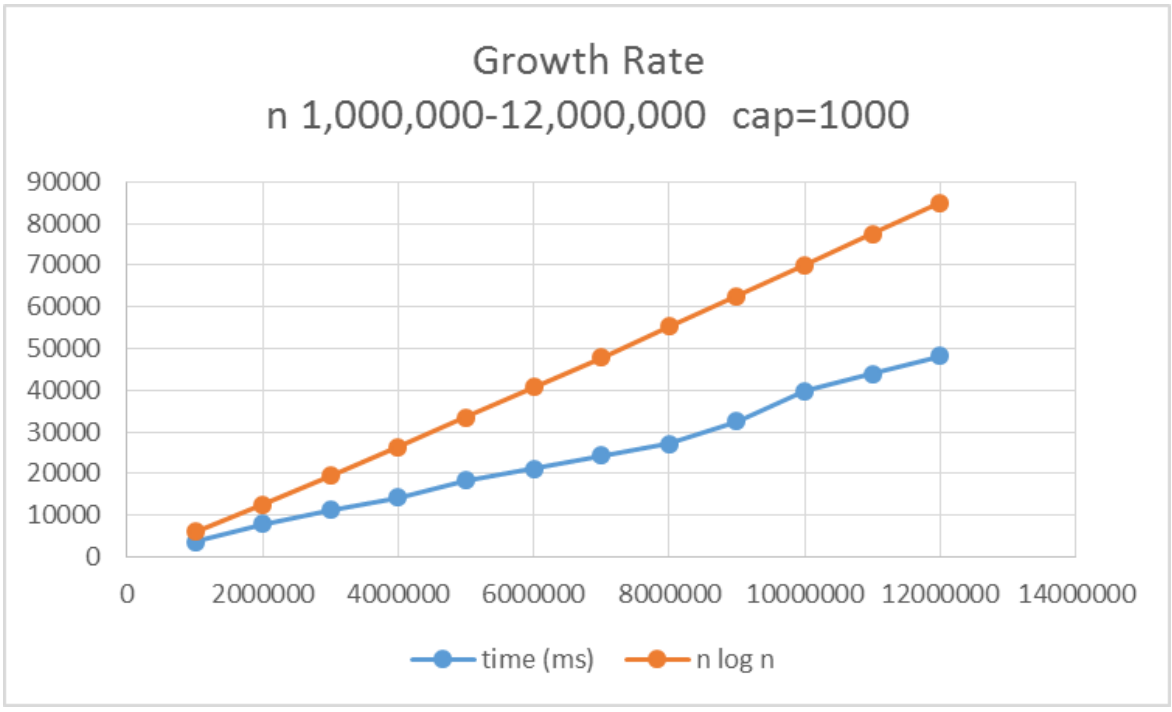
Evaluation of Data:

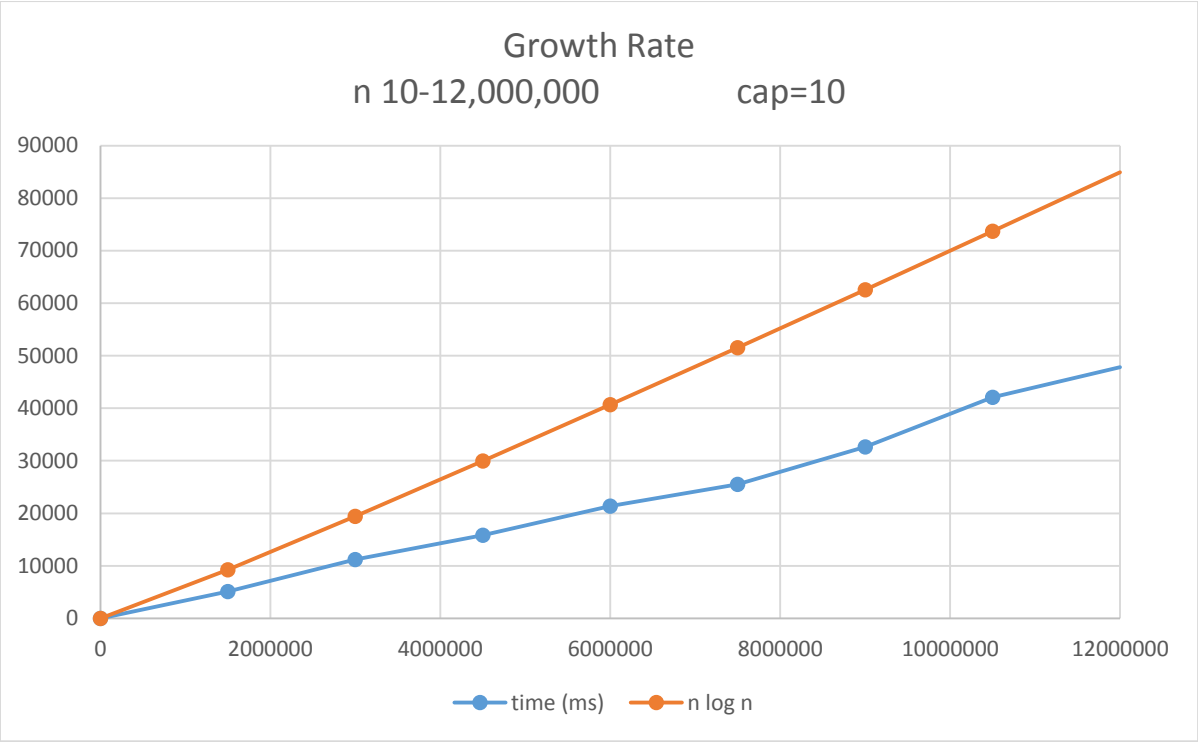
All files will be included in the submission .zip.

Visual Inspection:




Growth Rate
n 100,000-650,000 cap=1000
4000
3500
3000
2500
2000
1500
1000
500
0
100000
200000
300000
400000
500000
600000
time (ms)
n log n


Growth Rate
n 1,000,000-12,000,000 cap=1000
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
0
2000000
4000000
6000000
8000000
10000000
12000000
14000000
time (ms)
n log n



The graphs above show each simulation given different size inputs of n. Each run is plotted against the n log n line. By looking at each one, we can easily reject the exponential possibility for all. We can see that each simulation comes fairly close to the n log n trend line. Actually, our simulation line comes up below the n log n trend line in each one except for the first. The time for each one is in milliseconds.

Conclusion:

The growth of the fractional knapsack algorithm is very close to n log n. As I pushed the boundaries, my computer would fail to allocate memory.