Fractional Knapsack Growth Rate Investigation

CS5310 - Algorithms

Fall 2016

Shashank Kasinadhuni

12/08/2016

Hypothesis:

The fractional greedy knapsack algorithm has a time complexity of O(n log(n))

Test Design:

To test the hypothesis involved, I have taken the following steps:

1) Implemented the Fractional Greedy Knapsack algorithm in Python.
2) Run the code Fractional Knapsack functions for multiple array sizes:
    a) I have taken the first data range and tested it for 30 data points, and the second range for 25 data points.
    b) Tested for two separate ranges:
        i) Test one: 100 - 500000
        ii) Test two: 1,000,000 -2,000,000
    c) Analyzed all data ranges on a single graph/trendline to ensure that experimental growth rate stays consistent for a large range of input sizes.
3) In the first range each trial is run for 50 times and then its average time is calculated to minimize outlying time.Similarly,In the second range,each trial is taken for 40 times and then its average time is calculated.
4) Visually inspect charted data. If the algorithm clearly does not grow linearly, re-evaluate implementation and hypothesis.
5) If the data seems to grow linearly upon visual inspection use mathematical analysis to verify linear growth.

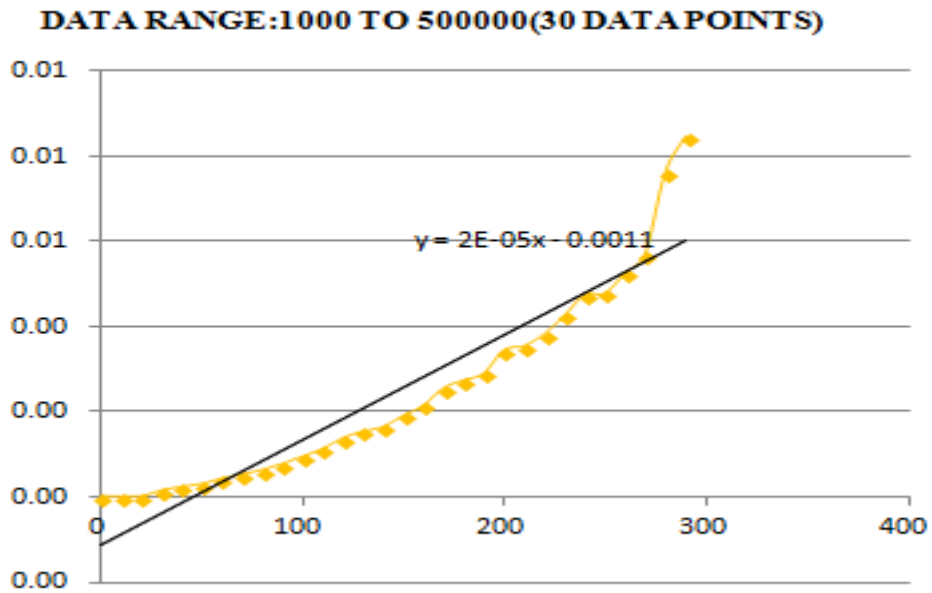
Evaluation of Data:

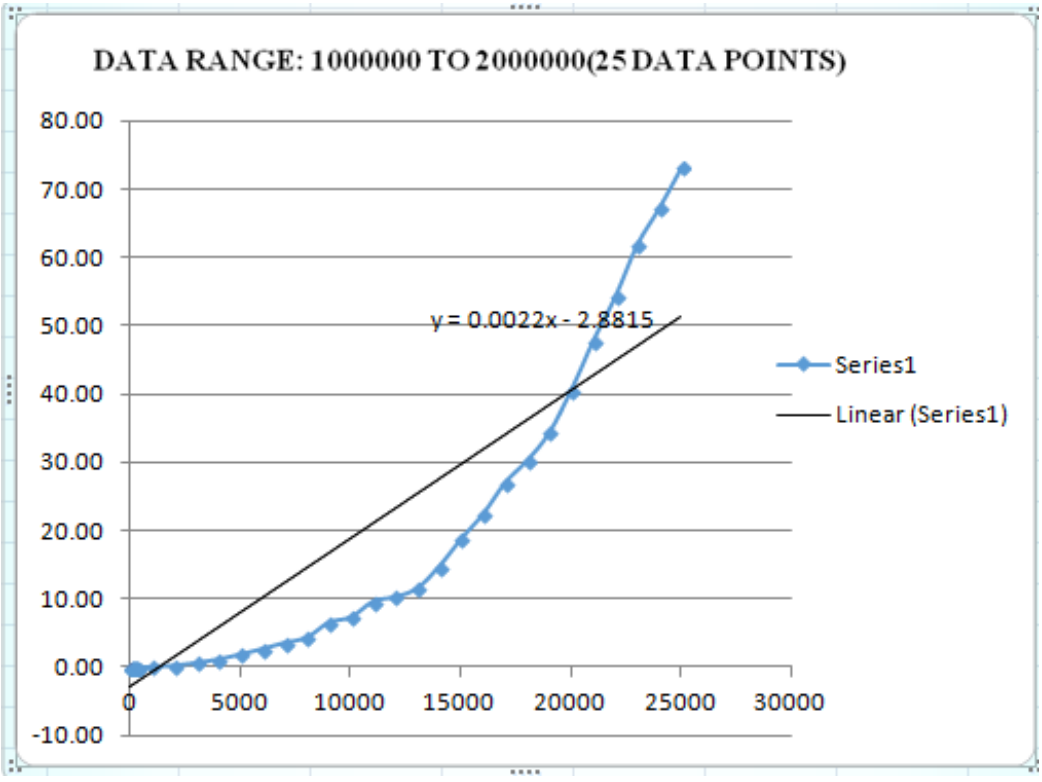
All files referenced can be found in the assignment submission CS5310-ShashankKasinadhuni-Project.zip file.

The implementation can be found in the file “project.py”. The raw output of the program can be found in the files “output100-500000.xlsx” , “output100000-200,000.xlsx” The final aggregated data is found in the file “data.xlsx”. The various “Screen Shot” files show the various trend lines applied to the data.

Visual inspection

Linear Trendlines

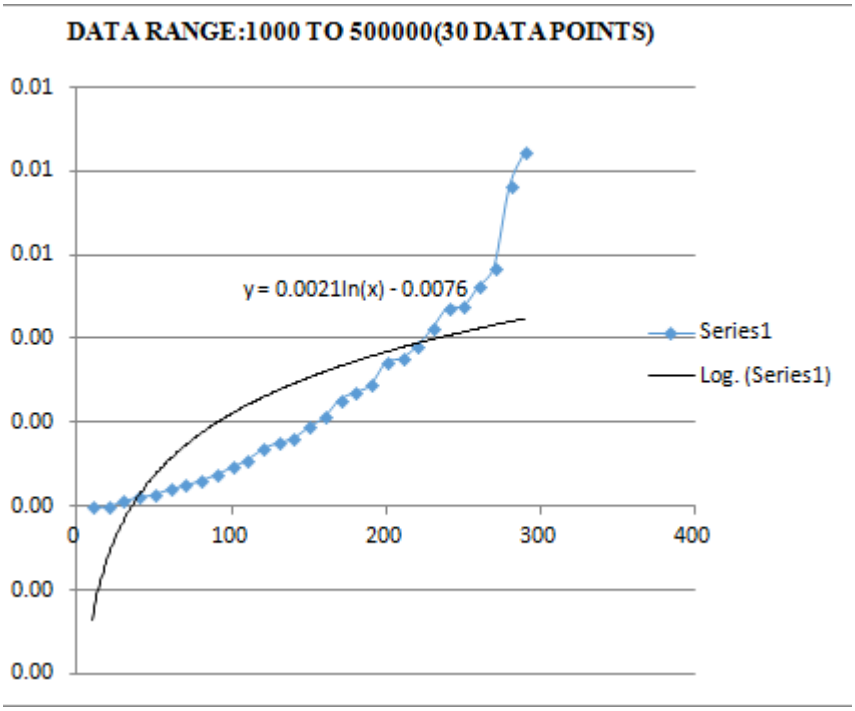


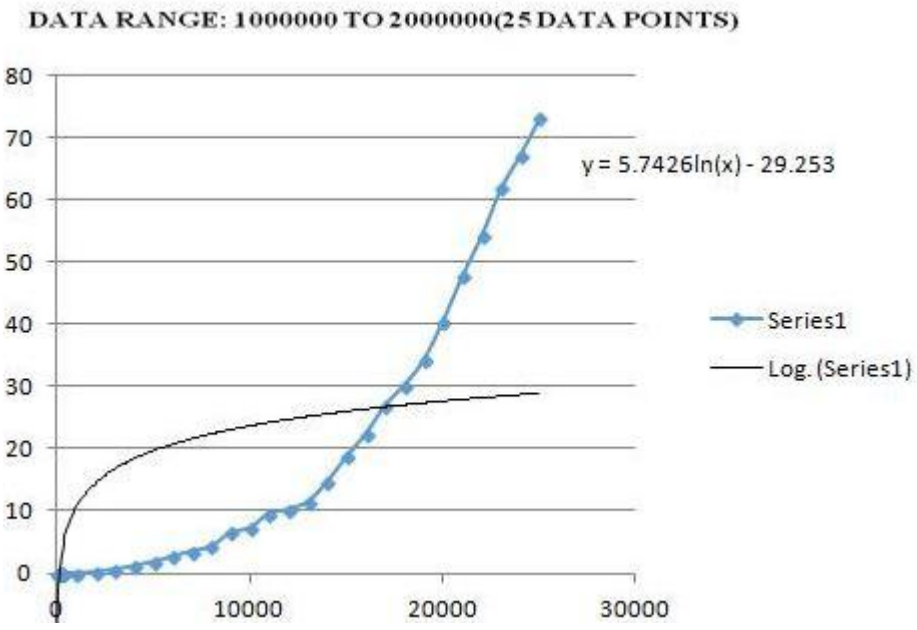


The graphs above show the charted results. The top graph shows the first range of data, and the second shows all data together. Upon visual inspection, the data does not seem to grow in a linear way. Since that is the case, Further Mathematical analysis is done.
I used simple trendline analysis in Microsoft Excel to complete this step. For both the Data ranges the set of array sizes and the full data set I tried exponential, linear, logarithmic, power, and polynomial trend lines. The equations of the trendline are shown on the graphs. I will look at in turn, and give the reason that I accept or reject each as the best fit to the data.

Logarithmic Trendlines:

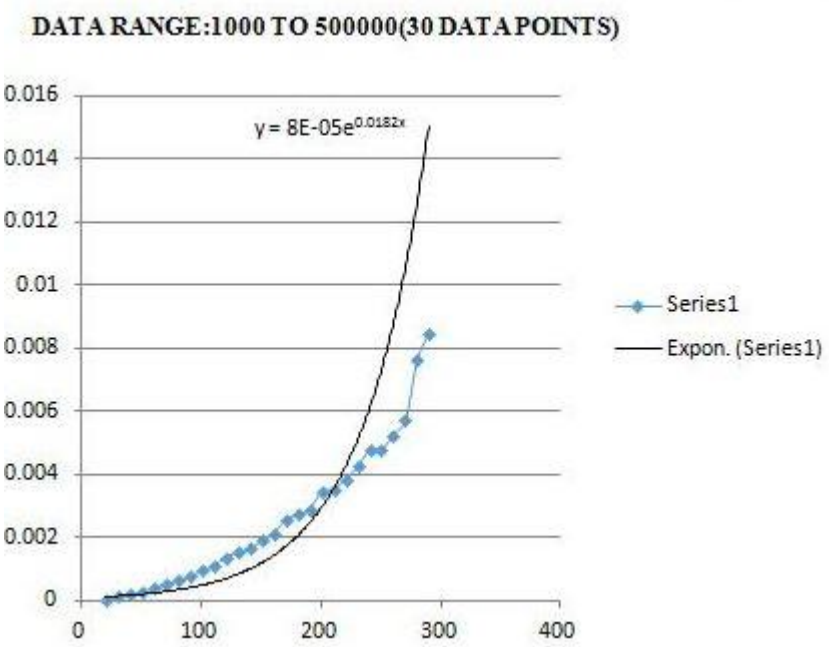
As with the logarithmic trendline, this can be rejected upon simple visual inspection. The trendline hardly any resemblance to the data.

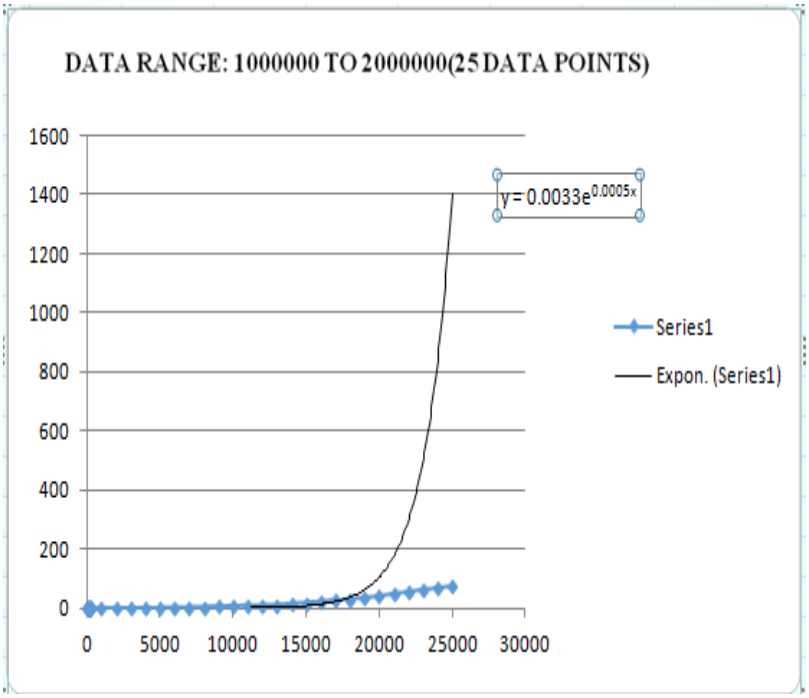




Exponential Trendlines

This can be rejected upon simple visual inspection. The first data set has very little resemblance with data but when the range along with input size is increased we can see that, we can simply reject this case as there is no resemblance with the data

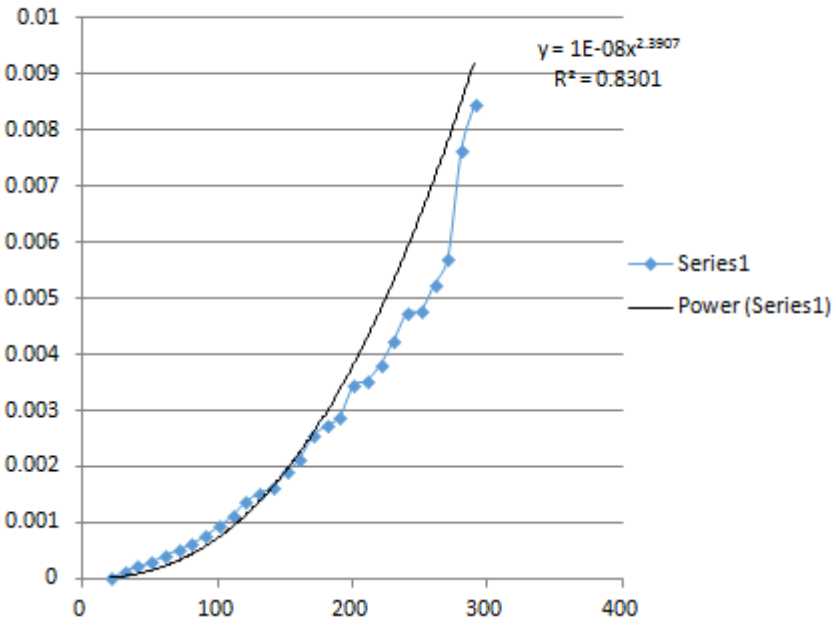




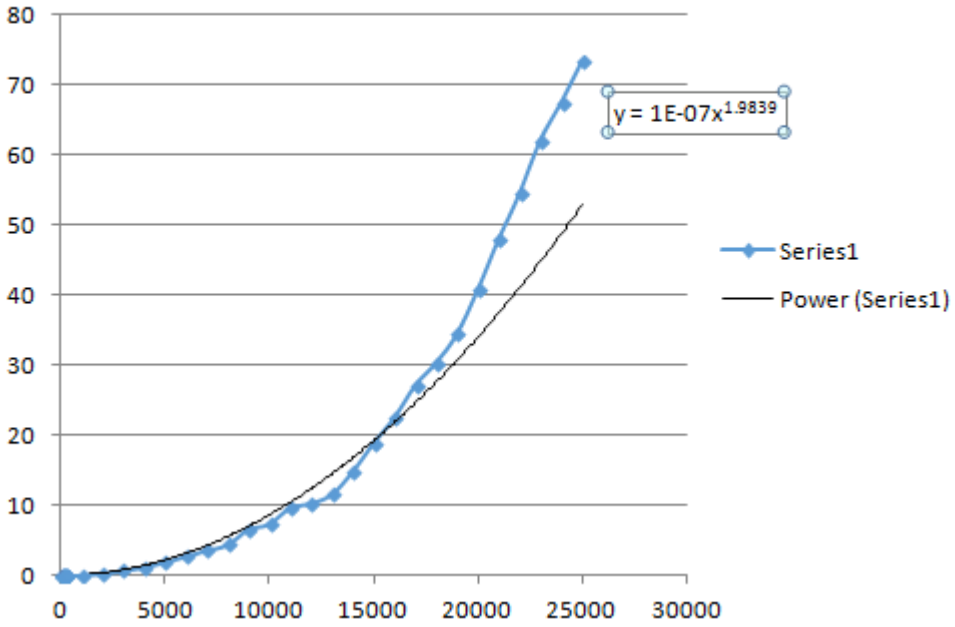
Power Trendline

For this trendline, half the points of data sets for both ranges seem to be accepted but later there is a variation with the data points. For both the graphs of the data sets taken, the power of the exponent is approximately closer to 2, therefore it can be presumed to be taken as a quadratic equation. So I cannot reject this case completely but will further deal with this case along with any other case in the conclusion.

DATA RANGE:1000 TO 500000(30 DATA POINTS)



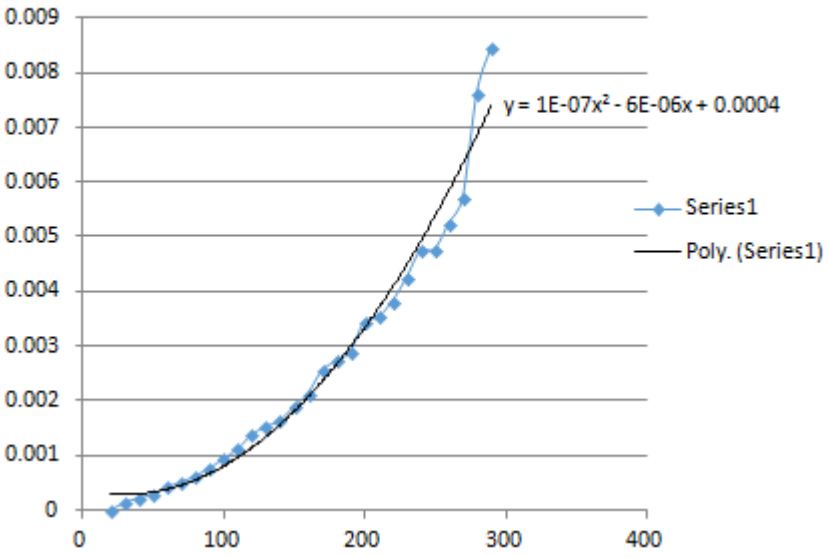
DATA RANGE: 1000000 TO 2000000(25 DATA POINTS)



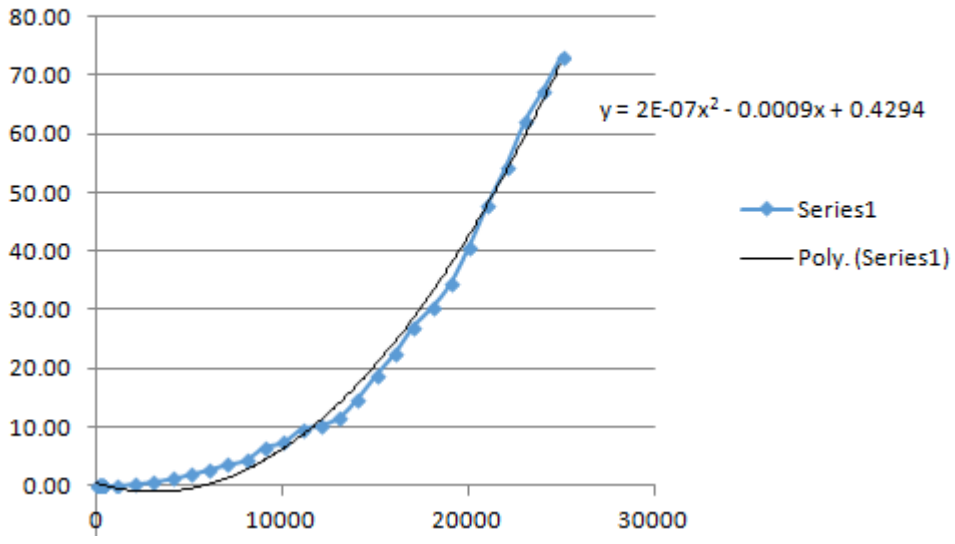
Polynomial Trendline

This trendline seems to be more acceptable on visual inspection as the points are very close along with matching datasets with this trendlines.Therefore I cannot reject this trendline.

DATA RANGE:1000 TO 500000(30 DATA POINTS)



DATA RANGE: 1000000 TO 2000000(25 DATA POINTS)



Conclusion

My data analysis shows that I can accept two possible cases, that is Power trendline and Polynomial trendline. The big Oh notation has a very well defined meaning. The coefficient of x^2 is very small in the polynomial graph.In the knapsack algorithm we say that it has to process n elements at a time and that means that the running time should not be less than or equal to n. Also when we say that f(n)=cg(n) where c is a constant, means that f(N) is bounded above by Cg(N) for all N. The constant 0.4294 is c obtained in the graph. Since Big O(nlog n) indicates that number of elements times the logarithm of the number of elements and the Knapsack algorithm accepts n elements for processing, My graph for both data ranges that is 100 to 5million and 10million to 20 million is neither quadratic nor polynomial, it is somewhere in between.Therefore I would conclude that my algorithm runs under O(nlogn).