Knapsack Growth Rate Investigation

CS5310 – Algorithms

Fall 2015

Simranjit Singh

11/02/2015

# Hypothesis:

The Knapsack algorithm has a growth rate of O (n log n).

# Test Design:

1. Implement the Knapsack algorithm in Ruby
2. Run the code for multiple data sizes
   1. Ensure that there are at least 10 data points for analysis
   2. Test the algorithm on multiple datasets.
      1. The program reads these item from different files
      2. The data set are divides into 10, 100, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 11000, 12000, 13000, 14000, and 15000 items.
   3. Run each individual trial ten times and average the timings to minimize the effects of outlying timings
3. Visually inspect data charted in excel. If the algorithm clearly doesn’t grow logarithmically, re-evaluate implementation and hypothesis
4. If the data seems to grow logarithmically upon visual inspection use mathematical analysis to verify linear growth

# Evaluation of Data:

All files reference can be found in the assignment submission .zip file.

The implementation of Knapsack can be found in the file “Knapsack.rb” and KnapsackTest2.rb”. The timing of running the different dataset are output to file “timelog.txt”. The final aggregated data is found in the file “KnapsackTimePlot.xlsx”.

Following are a total of 17 data points which were collected by running the Knapsack algorithm.

|  |  |
| --- | --- |
| Items | Time (s) |
| 10 | 0 |
| 100 | 0 |
| 1000 | 0.00312 |
| 2000 | 0.00312 |
| 3000 | 0.00468 |
| 4000 | 0.00624 |
| 5000 | 0.0078 |
| 6000 | 0.00936 |
| 7000 | 0.01248 |
| 8000 | 0.01404 |
| 9000 | 0.0156001 |
| 10000 | 0.0156 |
| 11000 | 0.01716 |
| 12000 | 0.0202801 |
| 13000 | 0.02184 |
| 14000 | 0.0218401 |
| 15000 | 0.02496 |

I used excel to build a scatter plot and add the different trend lines to the plot to evaluate the algorithm. The data size was 10 to 15000. Items of size 10 and 100 had 0 second time. Trend line for exponential and power trend line were not possible to plot because of the 0 second time. So these number were removed from the scatter plots. Following are the snapshots of the trend lines, so I will evaluate these one by one.

## Exponential Trendline

By just visually looking at the plot, this can be rejected. There is very little resemblance to the data.

## Power Trendline

This trend line is progress to exponential trend line but also it is very little resemblance to the data.

## Logarithm Trendline

We can clearly see that this tread line does not resemble the data. So we can reject this.

## Polynomial Trendline

The polynomial trend line looks good on visual inspection. But the x2 coefficient is very small. Let’s check the linear trendline before stating our conclusion.

## Linear Trendline

On visual inspection this seems good. By looking at this graph, we know that this algorithm is at learn linear.

# Conclusion

My data analysis show that polynomial and linear are the two possibilities for this programs. Polynomial x2 coefficient is very small and basically in linear function, just in another form. From this we can conclude that our algorithm is O (n).

My hypothesis for the algorithm was O (n log n) but the result is O (n). I think the way I implement my algorithm has lost some performance. Each time I get the item with max benefit, I index into the value array and set its value to -1. This could be one place where the algorithm is losing its performance.