**Fractional Greedy Kruskal Growth Rate Investigation**

**CS 4310 – Design & Analysis of Algorithms**

**Spring 2017**

**Andrew Burcroff**

**April 18th, 2017**

**Hypothesis:**

The Fractional Greedy Kruskal has a time complexity of O (n log (n))

**Test Design:**

1) Implement the Fractional Greedy Kruskal algorithm in ruby

2) Run code for different total Kruskal sizes

a) Collected 40 data points for analysis

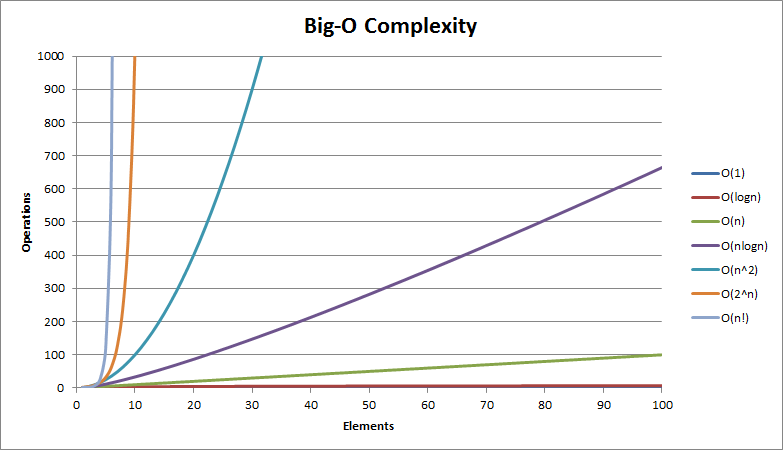
b) Analyze the data points to make sure that growth rate will match that of our hypothesis

3) At this point we can just look at the graph to see if our data is consisted if not visually then we must re work out experiment

4) After determining that the rate of change is similar use mathematical analysis to verify hypothesis

First, let’s look at the at the possibilities of time and complexity.

Here is an example of the Big O Complexity, I will be comparing too.



**Data Evaluation:**

**Exponential Trend Line**

**Power Trend Line:**

**Logarithmic Trend Line:**

**Polynomial Trend Line:**

**Linear Trend Line:**

**How does the Kruskal work?**

Kruskal's algorithm is a [minimum-spanning-tree algorithm](https://en.wikipedia.org/wiki/Minimum_spanning_tree" \l "Algorithms) which finds an edge of the least possible weight that connects any two trees in the forest.[[1]](https://en.wikipedia.org/wiki/Kruskal's_algorithm" \l "cite_note-:0-1) It is a [greedy algorithm](https://en.wikipedia.org/wiki/Greedy_algorithm) in [graph theory](https://en.wikipedia.org/wiki/Graph_theory) as it finds a [minimum spanning tree](https://en.wikipedia.org/wiki/Minimum_spanning_tree) for a [connected](https://en.wikipedia.org/wiki/Connectivity_(graph_theory)) [weighted graph](https://en.wikipedia.org/wiki/Glossary_of_graph_theory" \l "Weighted_graphs_and_networks) adding increasing cost arcs at each step.[[1]](https://en.wikipedia.org/wiki/Kruskal's_algorithm" \l "cite_note-:0-1) This means it finds a subset of the [edges](https://en.wikipedia.org/wiki/Edge_(graph_theory)) that forms a tree that includes every [vertex](https://en.wikipedia.org/wiki/Vertex_(graph_theory)), where the total weight of all the edges in the tree is minimized. If the graph is not connected, then it finds a *minimum spanning forest* (a minimum spanning tree for each [connected component](https://en.wikipedia.org/wiki/Connected_component_(graph_theory))).

**Analysis**

**Conclusion:**