Kruskal's Algorithm Growth Rate Investigation
CS4310 - Design & Analysis of Algorithms
Spring 2017
John Harvey
04/25/2017

Hypothesis:
Kruskal's algorithm has a time complexity of O(m logn)

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

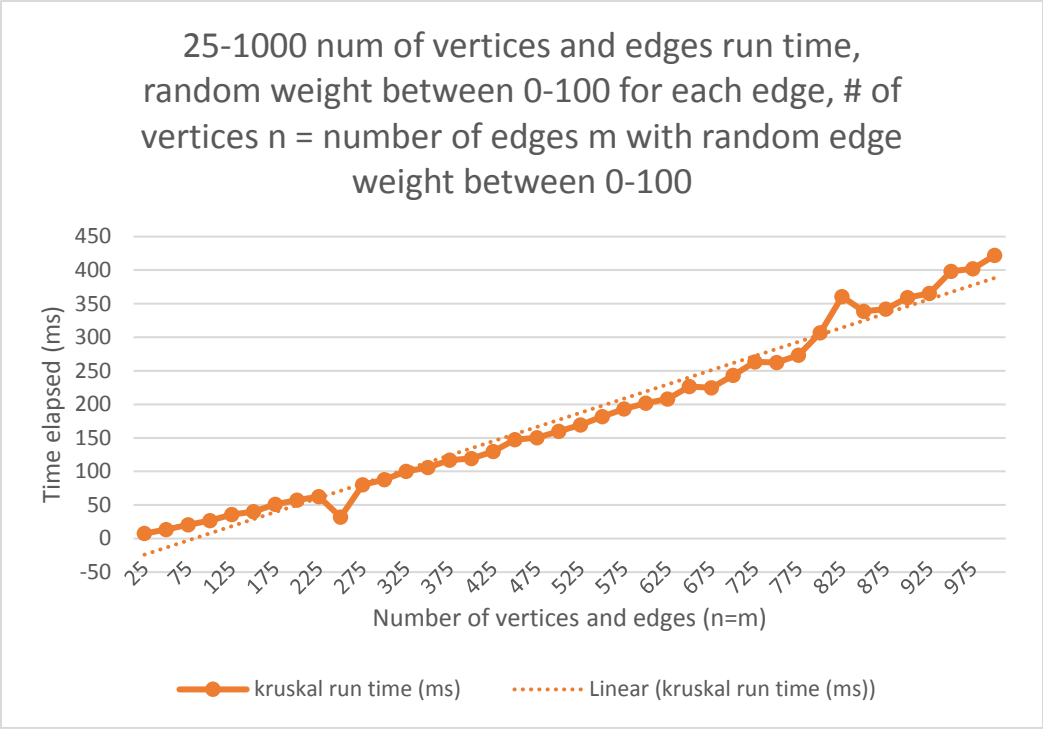
1) Implement Kruskal's algorithm in Ruby
2) Run the code for multiple amounts of n and m items, also with various random weights
    a) Test and gather data for when number of vertices = number of edges
        i) Test again with higher range of random edge weight
    b) Test and gather data for when there are more edges than vertices
        i) Test again with higher range of random edge weight
3) Inspect the graphs and compare the growth to an m log n growth rate
4) Examine code and make changes if results do not seem right

Evaluation of Data:

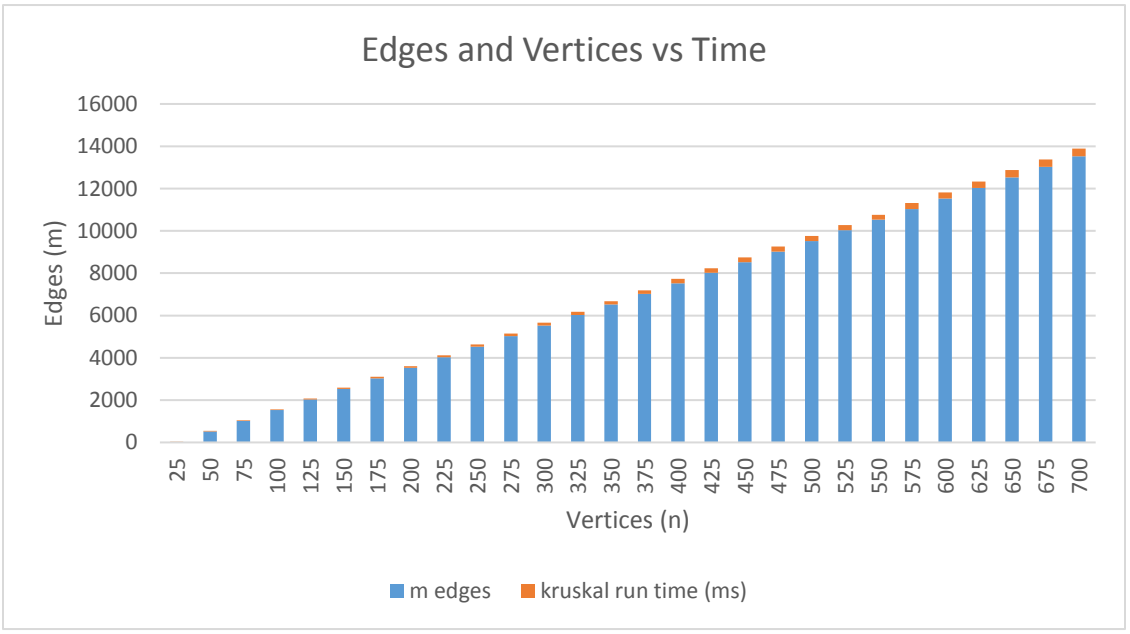
How the data was generated:

The user inputs a number for the amount of vertices, and the amount of edges. The edge weight is a randomly generated integer between 0 and any number you would like. I hard coded different edge weight ranges to reduce time. Multiple ranges of vertices, edges, and edge weights were done to achieve more accurate result. All data was logged in Excel.

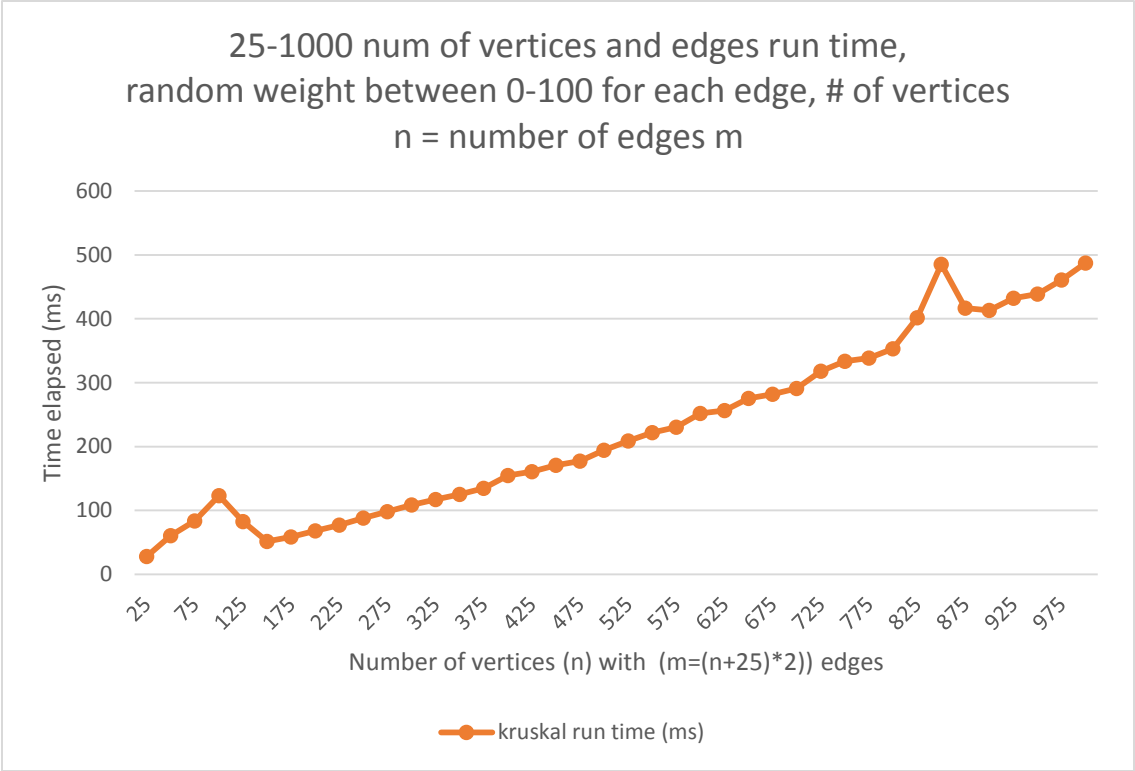
Visual Inspection:

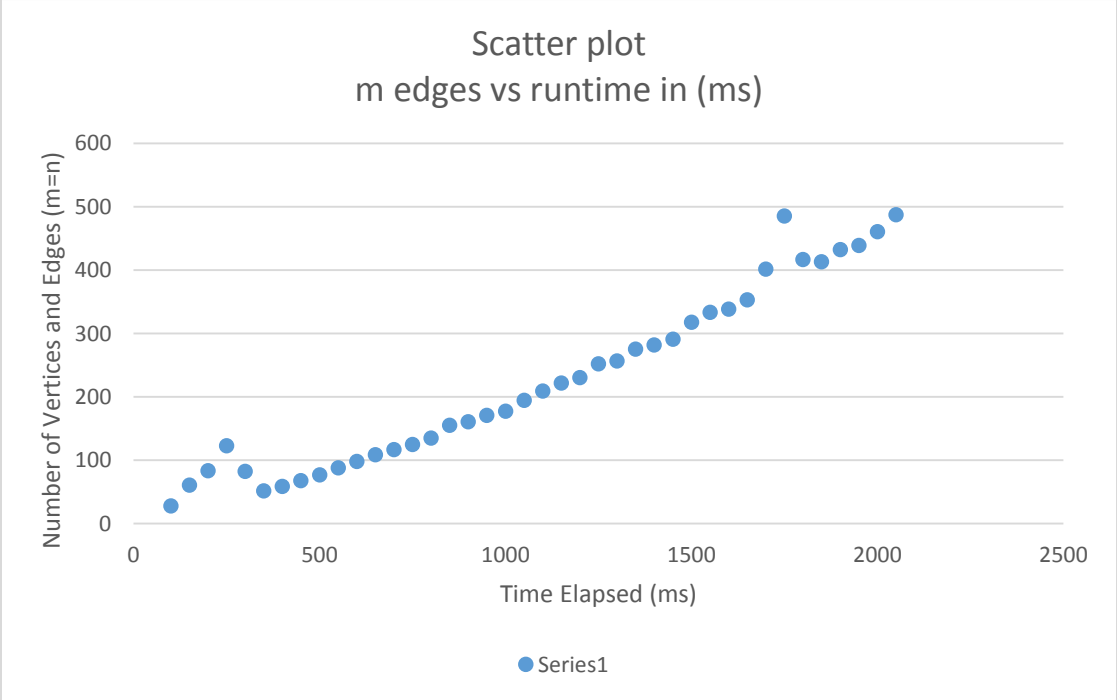


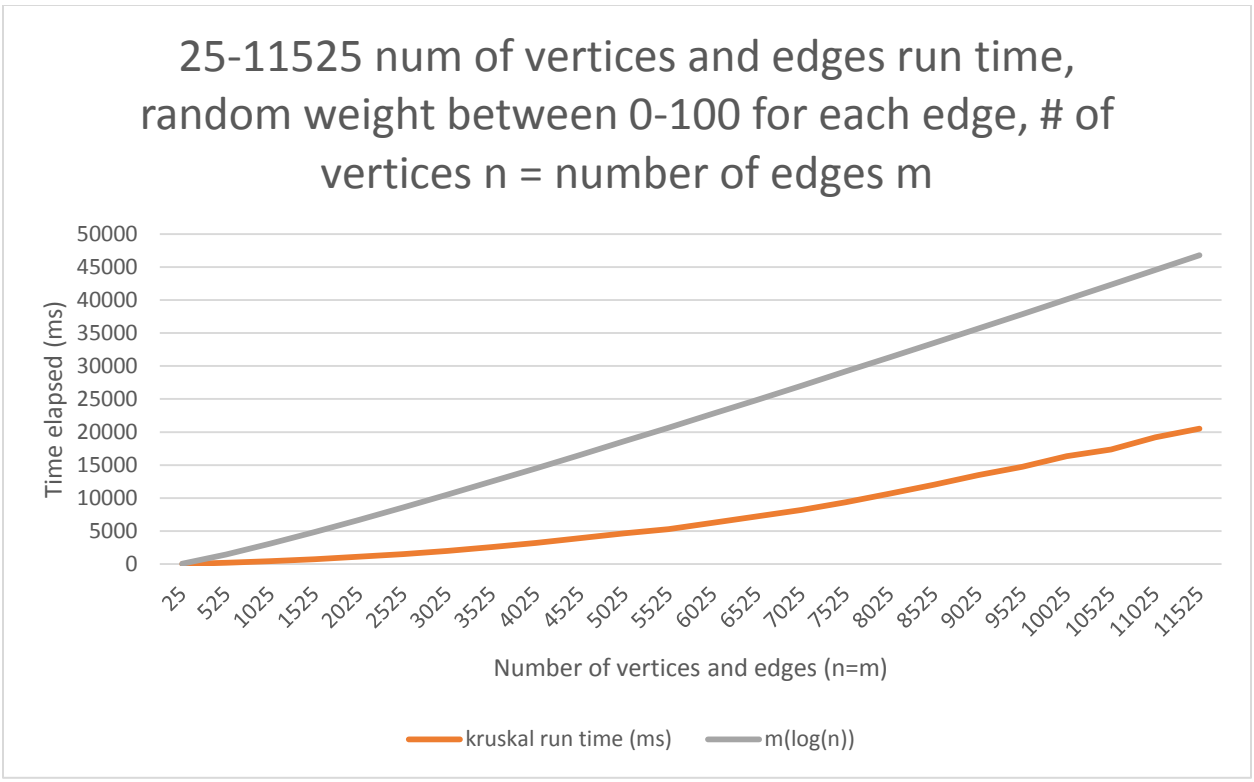
When there are more edges than vertices:

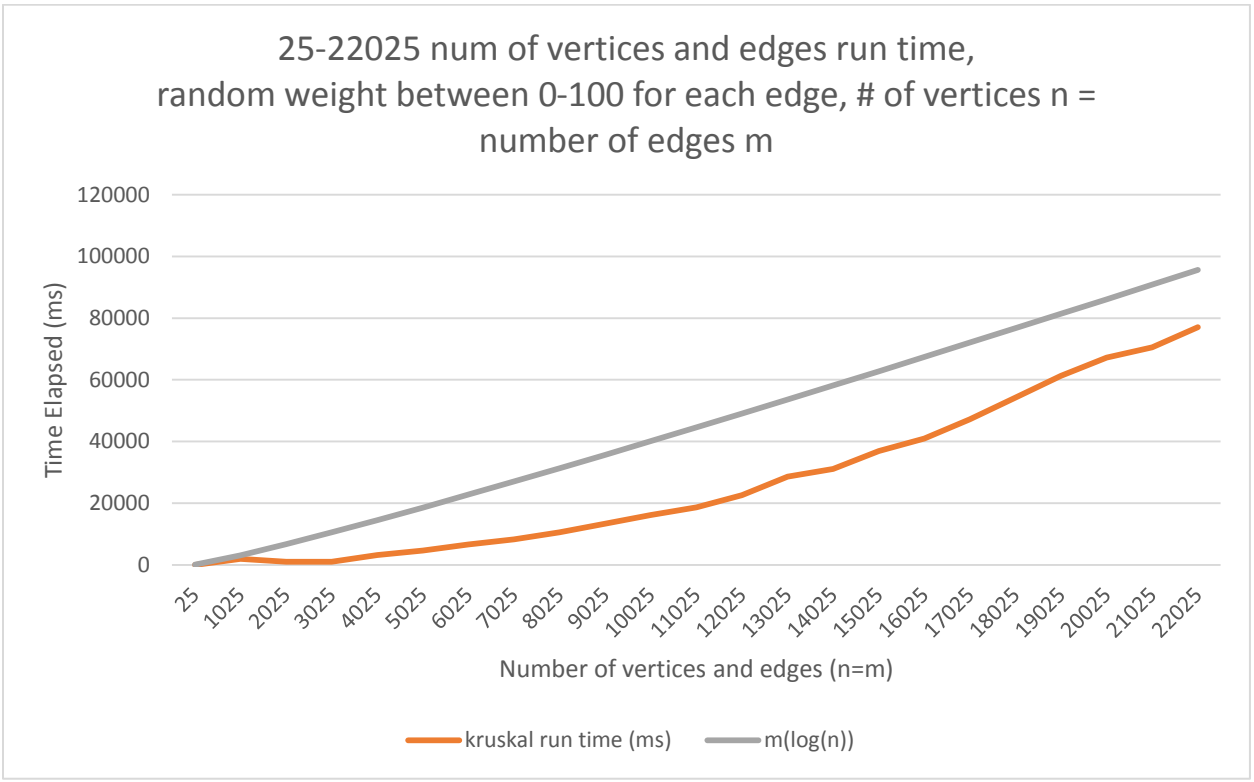


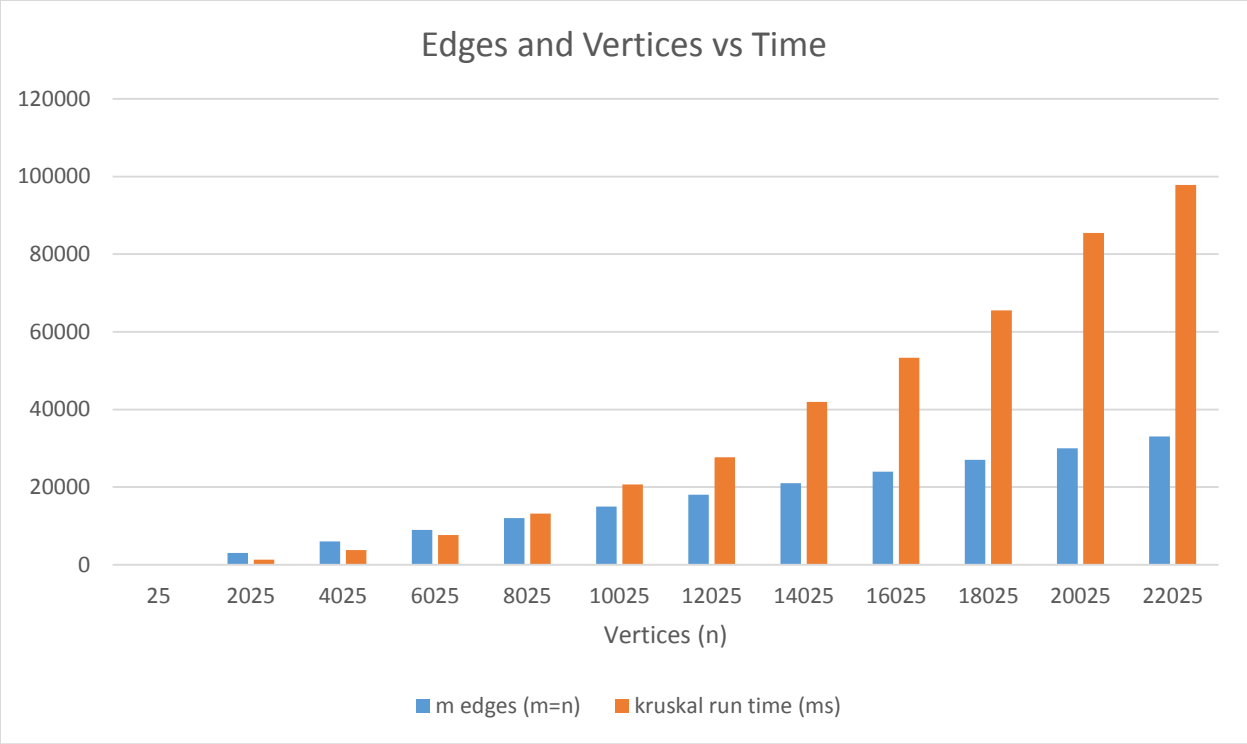
This shows that as the amount of vertices increase, the time increases, but when the edge amount increases more than the vertices, the time difference is minute.


25-1000 num of vertices and edges run time,
random weight between 0-100 for each edge, # of vertices
n = number of edges m
Time elapsed (ms)
600
500
400
300
200
100
0
25
75
125
175
225
275
325
375
425
475
525
575
625
675
725
775
825
875
925
975
Number of vertices (n) with (m=(n+25)*2)) edges
kruskal run time (ms)


Scatter plot
m edges vs runtime in (ms)
Number of Vertices and Edges (m=n)
600
500
400
300
200
100
0
0
500
1000
1500
2000
2500
Time Elapsed (ms)
Series1


25-11525 num of vertices and edges run time, random weight between 0-100 for each edge, # of vertices n = number of edges m
Time elapsed (ms)
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
25
525
1025
1525
2025
2525
3025
3525
4025
4525
5025
5525
6025
6525
7025
7525
8025
8525
9025
9525
10025
10525
11025
11525
Number of vertices and edges (n=m)
kruskal run time (ms)
m(log(n))


25-22025 num of vertices and edges run time, random weight between 0-100 for each edge, # of vertices n = number of edges m
Time Elapsed (ms)
120000
100000
80000
60000
40000
20000
0
25
1025
2025
3025
4025
5025
6025
7025
8025
9025
10025
11025
12025
13025
14025
15025
16025
17025
18025
19025
20025
21025
22025
Number of vertices and edges (n=m)
kruskal run time (ms)
m(log(n))



Conclusion:

I discovered that Kruskals algorithm does in fact run in Big O(m log n). Every set of tested data ran in under m log n time. The time it takes grows as the vertices grow more than as the edges grow. Some tests even ran faster with more edges. I also discovered that increasing the range of the possible edge weight did not drastically change the time elapsed.