Kruskal vs. Prim Algorithms

JJ McCauley

Methods

In order to evaluate these algorithms, a timing test was written in C++ using the Chronos library. Each iteration asks for a maximum number of vertices to add to the graph and a maximum number of vertices to add to the graph. These will be the independent variables, attempting to draw as many conclusions as possible through the following three samples:

1) Vertexes and Edges added at a 1:5 ratio (5 times as many edges as vertices)
   a) 200-1600 vertices & 1000-8000 edges
2) Constant Vertices and increasing Edges
   a) 500 vertices, 1000-50000 edges
3) Increasing Vertices and constant edges
   a) 100-2000 vertices, 20000 edges

These various sample sizes are aimed to draw various conclusions about the algorithms and determine the impacts that vertices and edges may have on them. For each table, a r^2 statistic will be extracted from four different equations to determine the optimal line of best fit, which will be stated and outlined in green.

Potential Limitations

The provided Kruskal's algorithm and the implemented Jarnik-Prim algorithm have inconsistent total weights, which may lead to incorrect data. Additionally, there is a certain degree of randomness due to the implementation of the tests, which could potentially sway results. Lastly, it is difficult to find the line of best fit when two independent variables are present, so a one-variable line of best fit is used for each sample, which may not be completely representative.

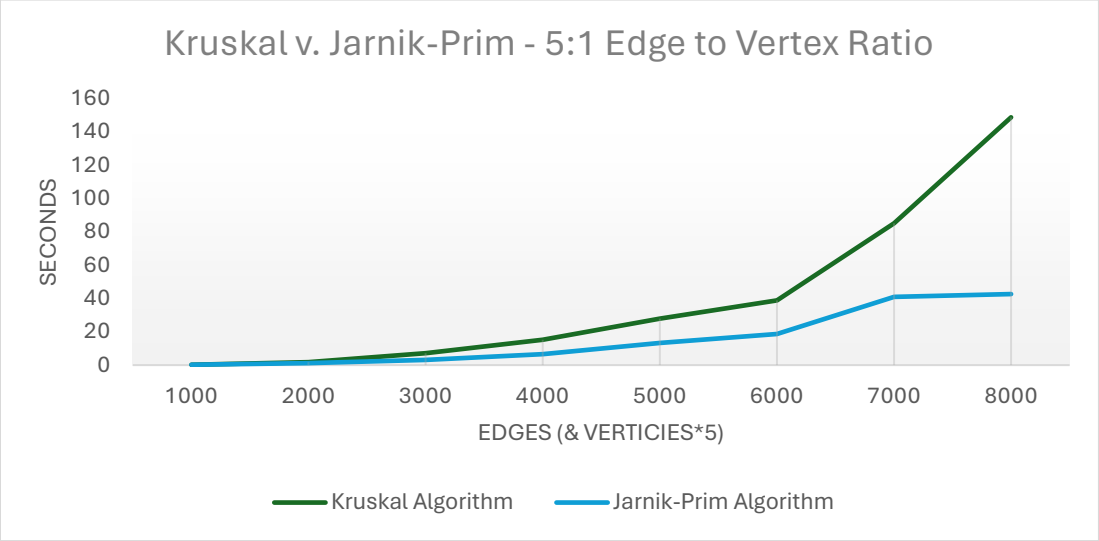
Results

The following pages show the results of the three ran tests.

Sample 1 – 1:5 Ratio

In this test, vertices and edges were limited to a 1:5 ratio, meaning that there were 5 times more edges than vertices in every test. This was to prevent the possibilities of unconnected graphs.

Vertices	Edges	Kruskal Algorithm	Jarnik-Prim Algorithm
200	1000	0.207002	0.229426
400	2000	1.86824	1.20404
600	3000	7.10201	3.20749
800	4000	15.2544	6.52319
1000	5000	27.97	13.3088
1200	6000	38.8004	18.6777
1400	7000	85.1494	40.9438
1600	8000	148.704	42.6447



Kruskal: O(|V²|)

$y_2 \sim x_2 \cdot \left(\log\left(\frac{x_2}{5}\right)\right)$

STATISTICS RESIDUALS

$R^2 = -1.03 \times 10^5$ e_1 plot

$y_2 \sim x_2 \cdot \left(\log\left(x_2\right)\right)$

STATISTICS RESIDUALS

$R^2 = -1.55 \times 10^5$ e_4 plot

$y_2 \sim ax_2 + b$

STATISTICS RESIDUALS

$r^2 = 0.7771$ e_1 plot

$r = 0.8815$

PARAMETERS

$a = 0.0186154$

$b = -43.1375$

$y_2 \sim ax_2^2 + bx_2 + c$

STATISTICS RESIDUALS

$R^2 = 0.9682$ e_3 plot

PARAMETERS

$a = 0.00000461646$

$b = -0.0229327$

$c = 26.1094$

Jarnik-Prim: O(|V²|)

$z_2 \sim x_2 \cdot \left(\log\left(\frac{x_2}{5}\right)\right)$

STATISTICS RESIDUALS

$R^2 = -9.38 \times 10^5$ e_1 plot

$z_2 \sim x_2 \cdot \left(\log\left(x_2\right)\right)$

STATISTICS RESIDUALS

$R^2 = -1.41 \times 10^6$ e_4 plot

$z_2 \sim ax_2 + b$

STATISTICS RESIDUALS

$r^2 = 0.8671$ e_2 plot

$r = 0.9312$

PARAMETERS

$a = 0.00653336$

$b = -13.5577$

$z_2 \sim ax_2^2 + bx_2 + c$

STATISTICS RESIDUALS

$R^2 = 0.9578$ e_3 plot

PARAMETERS

$a = 0.00000105626$

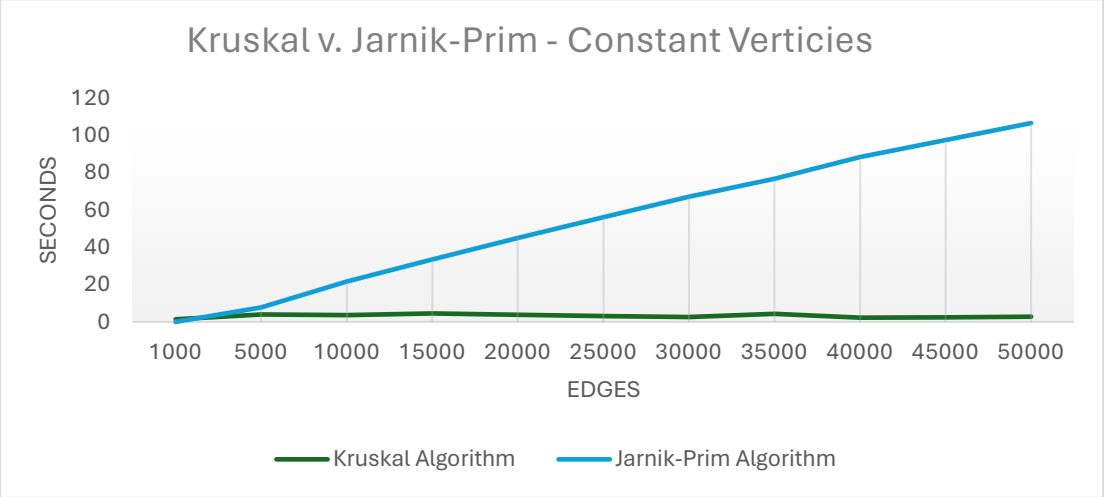
$b = -0.00297296$

$c = 2.28615$

Sample 2 – Constant Max Vertices

In this test, the maximum number of vertices stayed constant while the maximum number of edges increased.

Vertices	Edges	Kruskal Algorithm	Jarnik-Prim Algorithm
500	1000	1.38368	0.018443
500	5000	3.94717	7.82721
500	10000	3.65603	21.5854
500	15000	4.534	33.5567
500	20000	3.8639	44.9752
500	25000	3.06732	56.1374
500	30000	2.62408	67.1349
500	35000	4.36651	76.7804
500	40000	2.19134	88.4304
500	45000	2.50296	97.5039
500	50000	2.71495	106.649



Kruskal: O(|V²|)

$y_3 \sim x_3 \cdot \left(\log\left(\frac{x_3}{5}\right)\right)$

STATISTICS RESIDUALS

$R^2 = -1.49 \times 10^{10}$ e_1 plot

$y_3 \sim x_3 \cdot \left(\log\left(x_3\right)\right)$

STATISTICS RESIDUALS

$R^2 = -2.07 \times 10^{10}$ e_4 plot

$y_3 \sim ax_3 + b$

STATISTICS RESIDUALS

$r^2 = 0.02824$ e_2 plot

$r = -0.168$

PARAMETERS

$a = -0.0000100823$

$b = 3.42133$

$y_3 \sim ax_3^2 + bx_3 + c$

STATISTICS RESIDUALS

$R^2 = 0.2769$ e_3 plot

PARAMETERS

$a = -2.197 \times 10^{-9}$

$b = 0.000101035$

$c = 2.55591$

Jarnik-Prim: O(|V²|)

$z_3 \sim x_3 \cdot \left(\log\left(\frac{x_3}{5}\right)\right)$

STATISTICS RESIDUALS

$R^2 = -1.1 \times 10^{7}$ e_1 plot

$z_3 \sim x_3 \cdot \left(\log\left(x_3\right)\right)$

STATISTICS RESIDUALS

$R^2 = -1.53 \times 10^{7}$ e_4 plot

$z_3 \sim ax_3 + b$

STATISTICS RESIDUALS

$r^2 = 0.9978$ e_2 plot

$r = 0.9989$

PARAMETERS

$a = 0.00220312$

$b = -0.678378$

$z_3 \sim ax_3^2 + bx_3 + c$

STATISTICS RESIDUALS

$R^2 = 0.9997$ e_3 plot

PARAMETERS

$a = -7.0139 \times 10^{-9}$

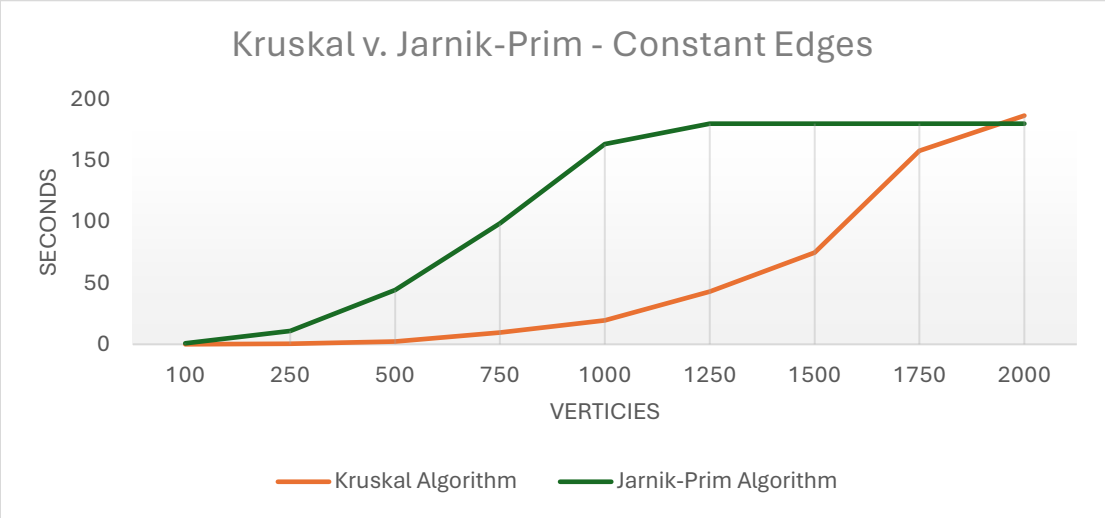
$b = 0.00255786$

$c = -3.4412$

Sample 3 – Constant Max Edges

In this test, the maximum number of edges stayed constant while the maximum number of vertices increased. (*Note, time for Jarnik-Prim was not recorded past 1250 vertices, as the time became excessively large)

Vertices	Edges	Kruskal Algorithm	Jarnik-Prim Algorithm
100	20000	0.034467	0.825877
250	20000	0.350903	10.7774
500	20000	2.26623	44.3235
750	20000	9.62595	98.5201
1000	20000	19.4833	163.277
1250	20000	43.005	180+
1500	20000	74.8397	180+
1750	20000	157.833	180+
2000	20000	186.579	180+



Kruskal: O(|V²|)

$y_4 \sim x_4 \cdot \left(\log\left(\frac{x_4}{5}\right)\right)$

STATISTICS $R^2 = -1840$ RESIDUALS e_1 plot

$y_4 \sim x_4 \cdot \left(\log(x_4)\right)$

STATISTICS $R^2 = -3057$ RESIDUALS e_4

$y_4 \sim ax_4 + b$

STATISTICS $r^2 = 0.8194$ $r = 0.9052$ RESIDUALS e_2 plot

PARAMETERS $a = 0.0966037$ $b = -42.7863$

$y_4 \sim ax_4^2 + bx_4 + c$

STATISTICS $R^2 = 0.979$ RESIDUALS e_3 plot

PARAMETERS $a = 0.0000805026$ $b = -0.0696368$ $c = 11.1665$

Jarnik-Prim: O(|V²|)

$z_4 \sim x_4 \cdot \left(\log\left(\frac{x_4}{5}\right)\right)$

STATISTICS $R^2 = -438.4$ RESIDUALS e_1 plot

$z_4 \sim x_4 \cdot \left(\log(x_4)\right)$

STATISTICS $R^2 = -786.1$ RESIDUALS e_4

$z_4 \sim ax_4 + b$

STATISTICS $r^2 = 0.9647$ $r = 0.9822$ RESIDUALS e_2 plot

PARAMETERS $a = 0.181781$ $b = -30.9814$

$z_4 \sim ax_4^2 + bx_4 + c$

STATISTICS $R^2 = 0.9995$ RESIDUALS e_3 plot

PARAMETERS $a = 0.000136862$ $b = 0.0320034$ $c = -4.69393$

Analysis

After observing the data, the following observations can be made:

General Observation

After testing all of the data and plugging them into Desmos to find the line of best fit, it can be determined that all of the data follows the time complexity of

$$O(|V^2|),$$

which misaligns with the theoretical complexity of $O(Elg(V))$, as discussed in the instructions. This could be due to the testing environment (randomness and samples), the method to find the line of best fit, or the implementation of the algorithms.

Sample 1

When both the edges and vertices are increased proportionally (5:1, in this case), the time appears to increase (roughly) exponentially for both algorithms. However, the implementation of Prim's algorithm is faster in this sample, finishing in roughly have the time for most data points. Additionally, Kruskal's algorithm appears to exponentially break away as the edges and vertexes become larger, suggesting that it could potentially not be the best in large data sets. However, the limitations mentioned could have a significant impact on this.

Sample 2

When the number of vertices is fixed and the number of edges increases, Prim's algorithm appears to increase linearly as more edges are added. This is likely due to the algorithm having the check for incidence (search) during every iteration, which takes a consistent amount of time throughout the program. Kruskal's algorithm, however, takes roughly the same amount of time throughout all the iterations, being significantly faster than Prim's after the first test. This is likely due to the conditional statement in Kruskal's algorithm that moves to the next iteration if the edge is already in the graph, which would be hit every time after so many edges have been inserted. Overall, Kruskal's algorithm is significantly superior over Prim's when there will be significantly more edges than vertices (as seen in this sample).

Sample 3

When the number of edges is fixed and the number of vertices increases, both algorithms appear to increase exponentially. However, Prim's algorithm is significantly slower, taking around ten times as much time as Kruskal's algorithm for the smaller samples. Kruskal's algorithm likely performs much better because it does not check for incidence, which does not require accessing or searching all of the vertices. Overall, Kruskal's algorithm is superior to Prim's when there is expected to be an increasing number of vertices while the edges stay the same.

Conclusion

According to this testing, it is apparent that Prim's algorithm and Kruskal's algorithm can be superior in their own use cases. If there will be a fixed number of edges or vertices, if the number of edges will increase significantly faster than the number of vertices, or if the number of vertices will increase significantly faster than the number of edges (assuming that the graph stays connected), then Kruskal's algorithm appears to be the better choice. On the other hand, if the number of edges and vertices is expected to increase proportionally, then the Jarnik-Prim algorithm is the better option. However, as always, it is important to acknowledge the limitations, and more tests may need to be run in the future to determine the most optimal algorithm in each specific case.