Thomas Pearson

05/06/2024

Don Spickler

COSC – 320

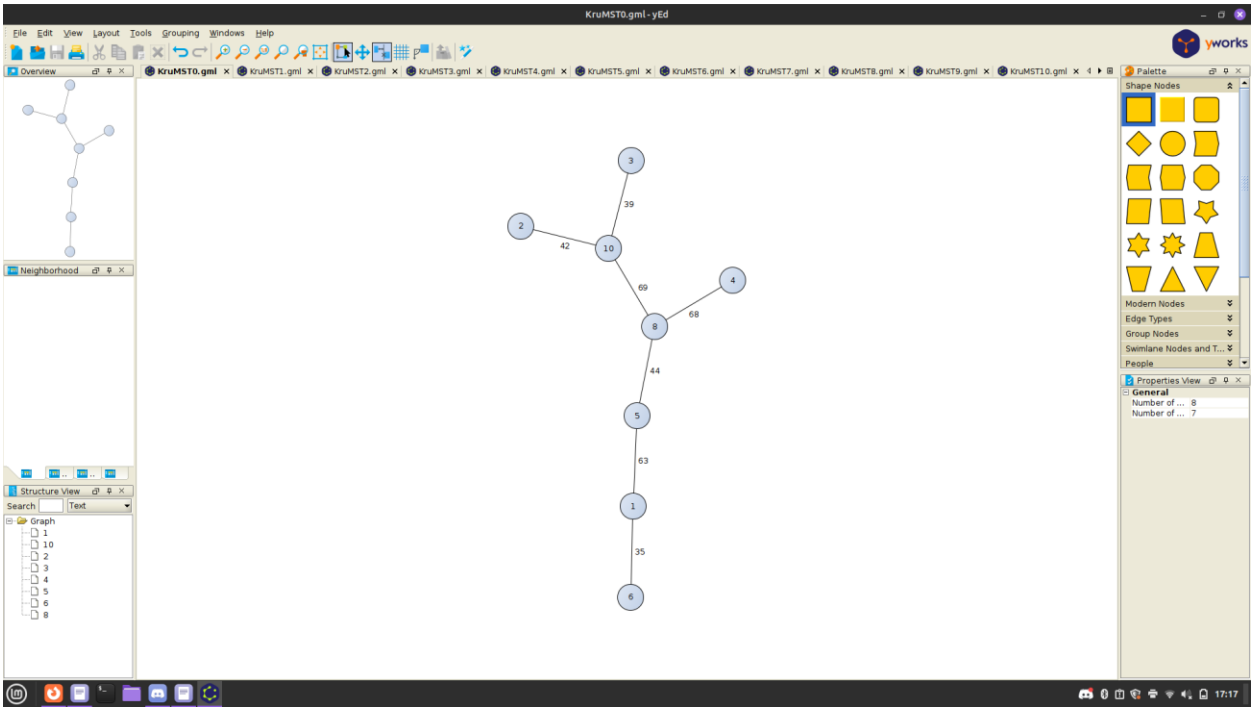
Analysis of Prim and Kruskal's Shortest Path Algorithms

Max Vertices	Number of Edges	Kurskal (ms)	Prim (ms)
10	10	143	729
10	20	351	1337
10	30	425	1338
10	40	464	1334
10	50	449	1317
10	60	219	1483
10	70	294	1263
10	10	163	810
20	10	607	5755
30	10	1746	37238
40	10	4335	76032
50	10	9407	187556
60	10	12342	314183
70	10	17224	497667

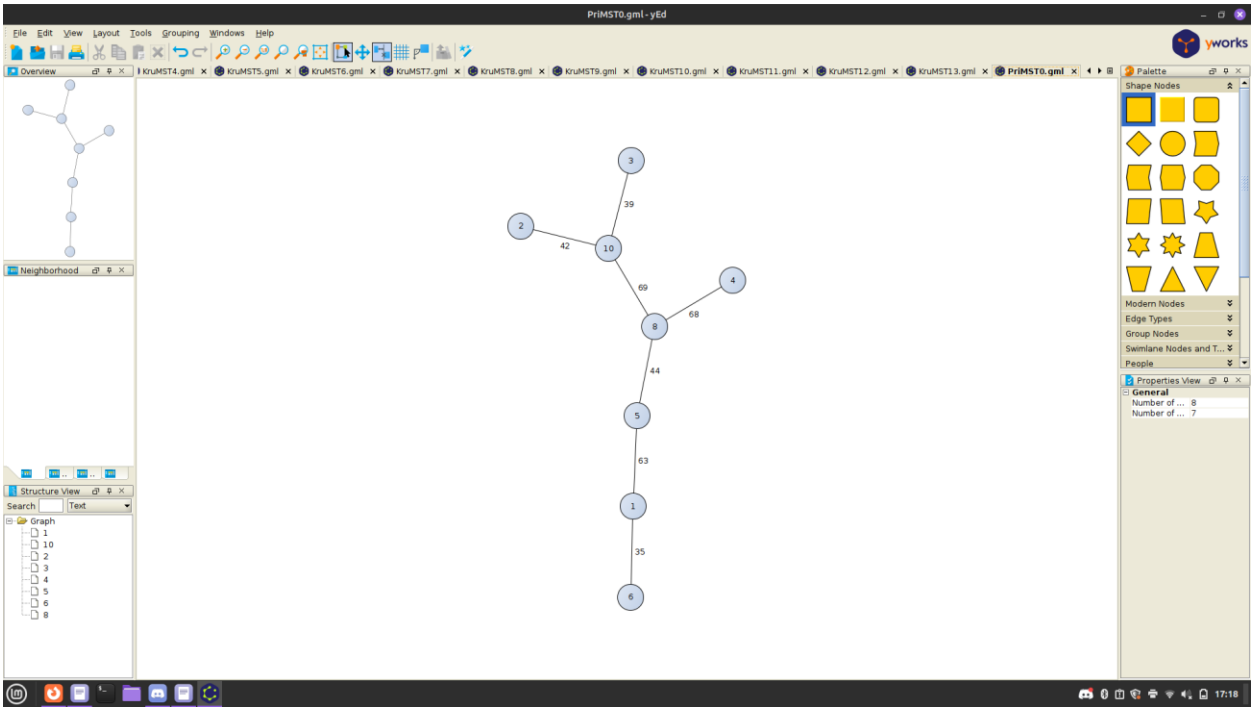
Upon first glance at the chart, it appears as though when we have the same number for max vertices and number of edges it's the lowest run time. As we increase edges and keep Vertices constant it increases the time before stabilizing. When we raise the max vertices but keep the number of edges constant the time begins to increase exponentially.

Round 1

Kruskal

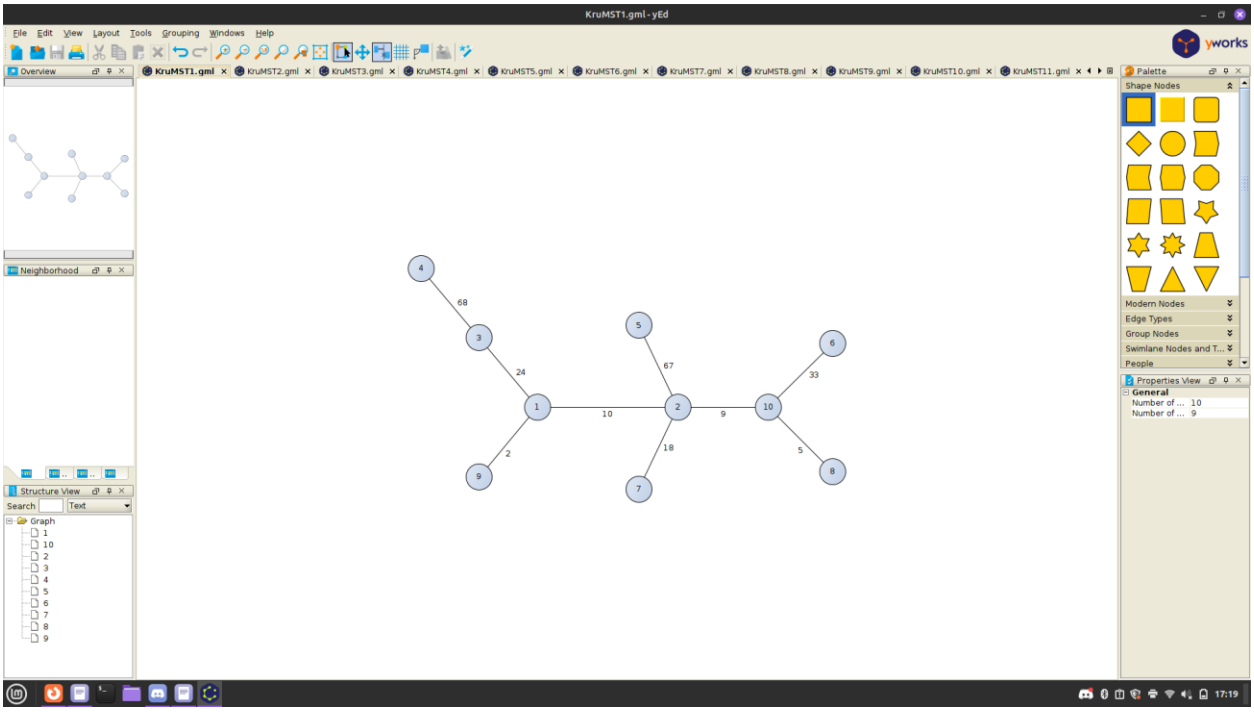


Prim

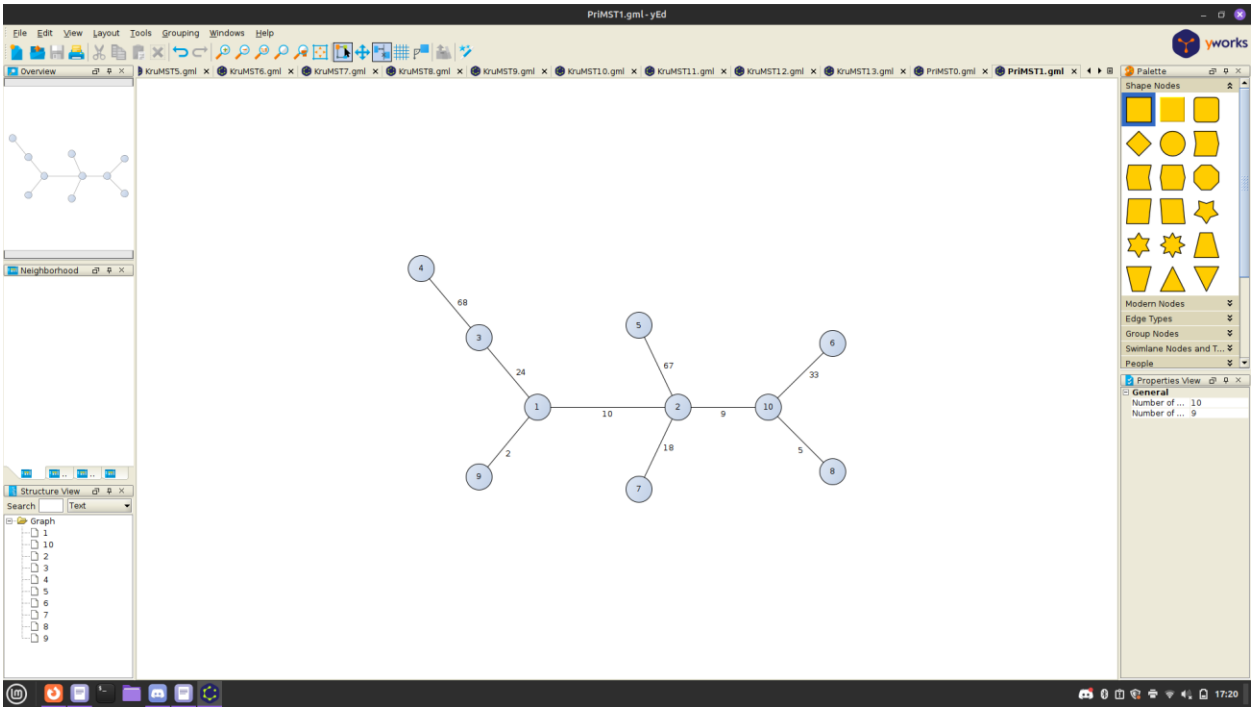


Round 2

Kruskal



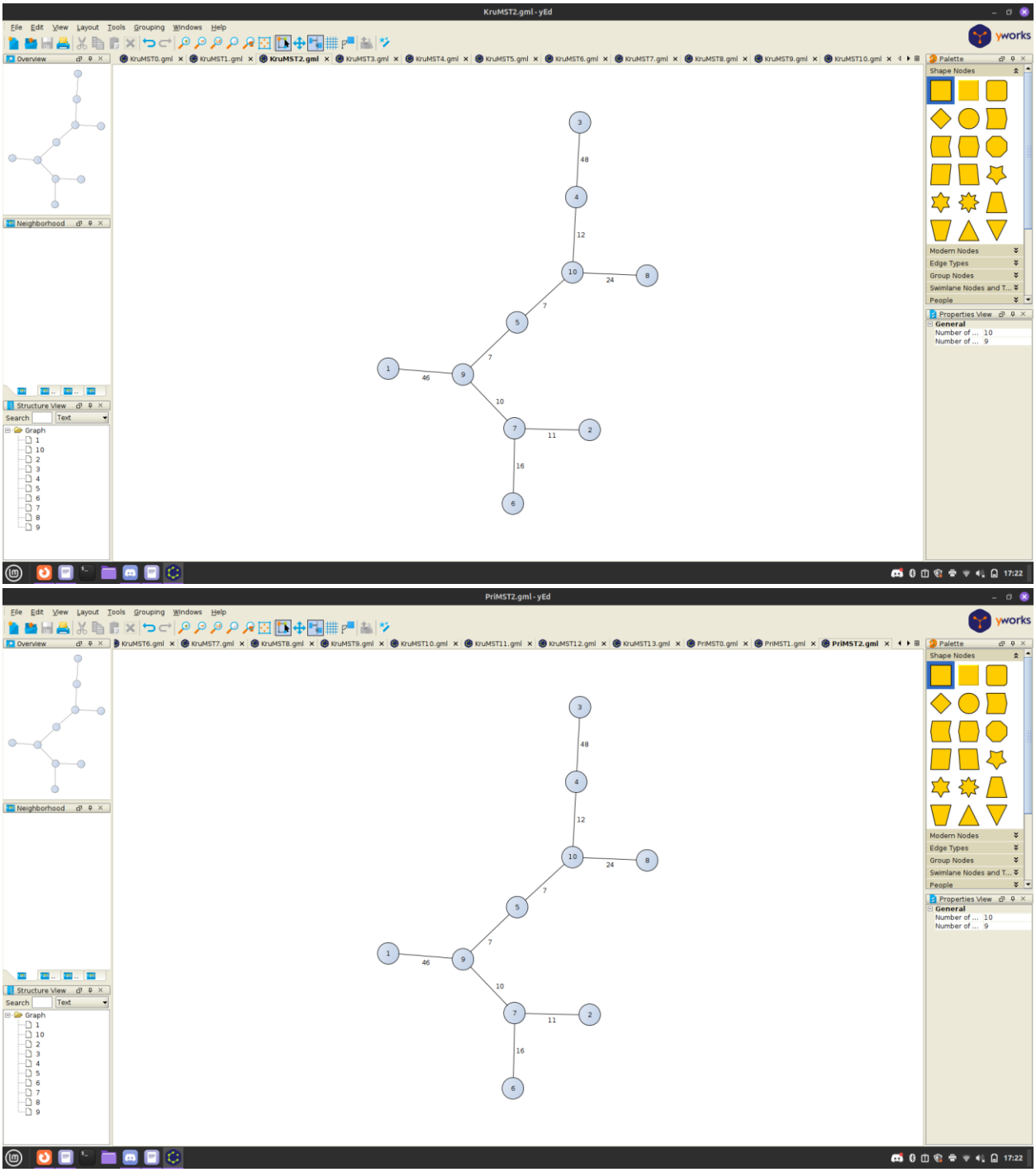
Prim



After two rounds it looks like the graphs produced are identical to each other. This could be due to the graph used being unique at this size.

Round 3

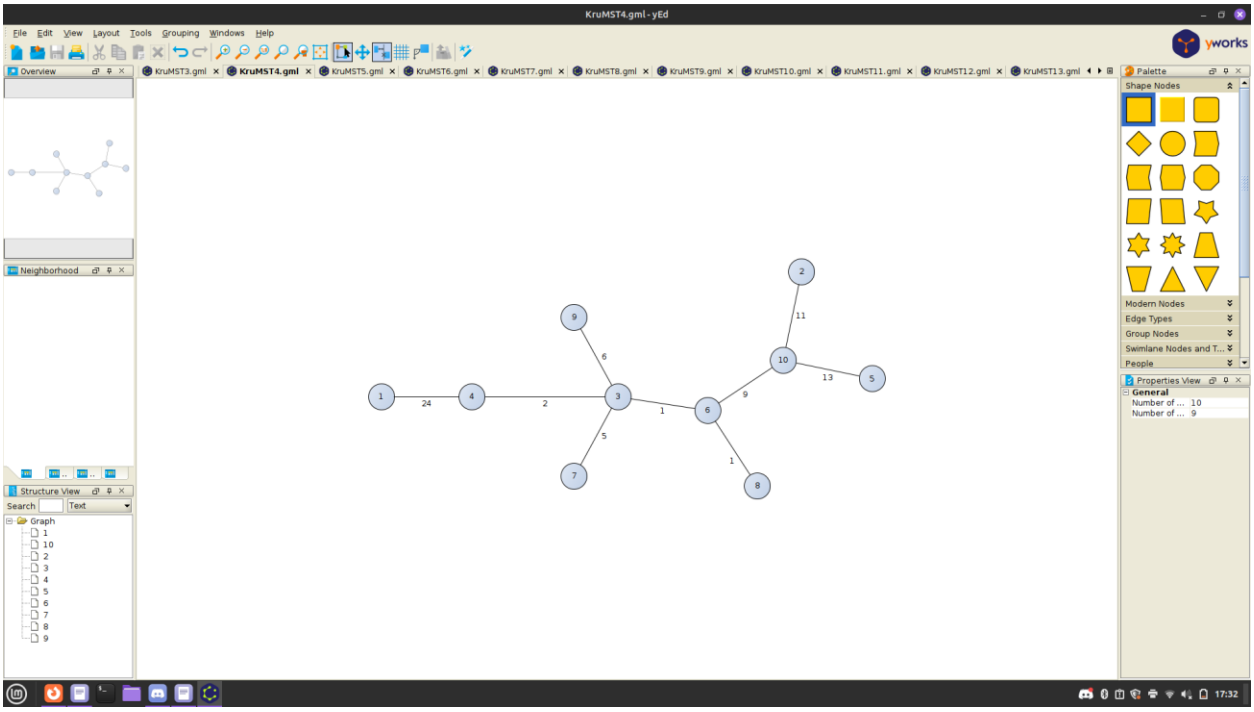
Kruskal



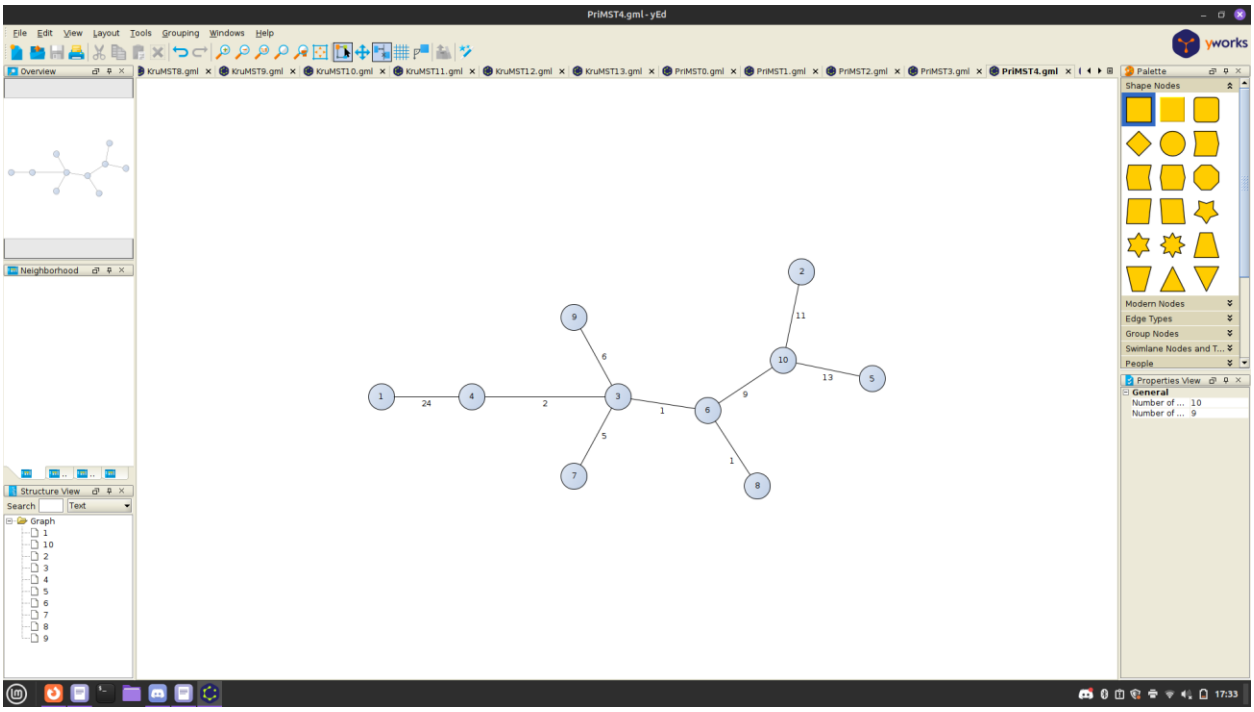
After 3 rounds the graphs are still identical

Round 4

Kruskal



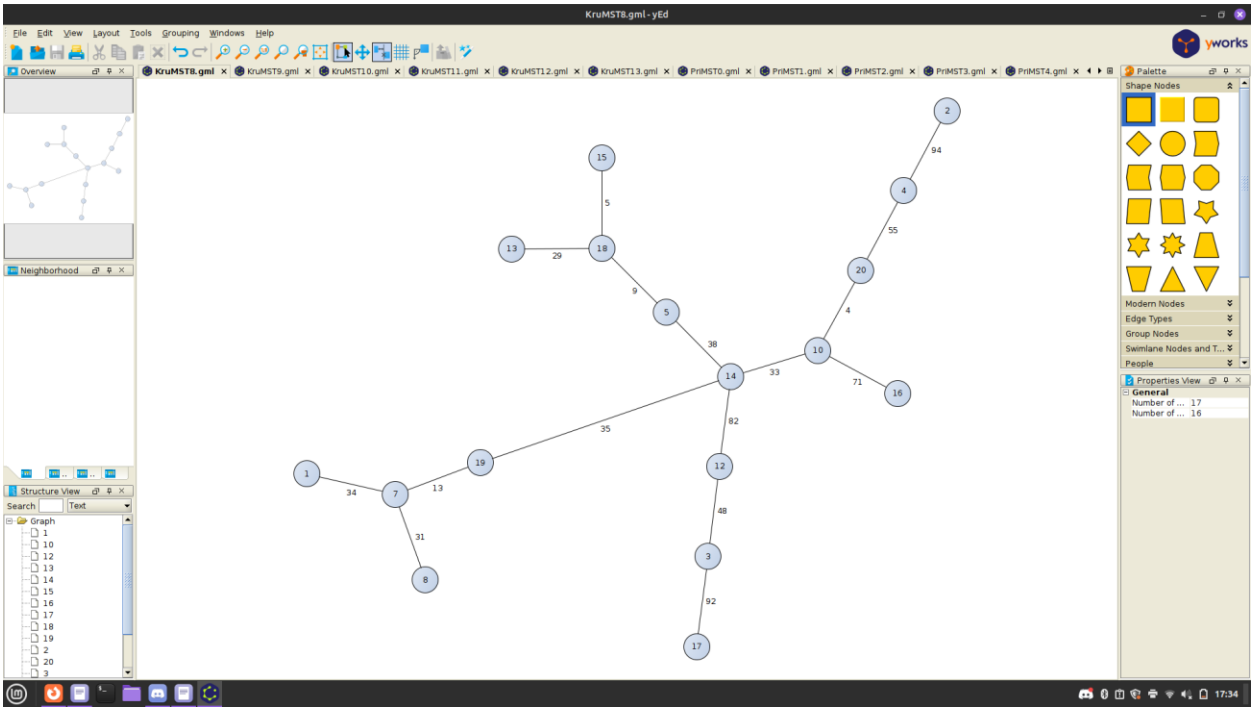
Prim



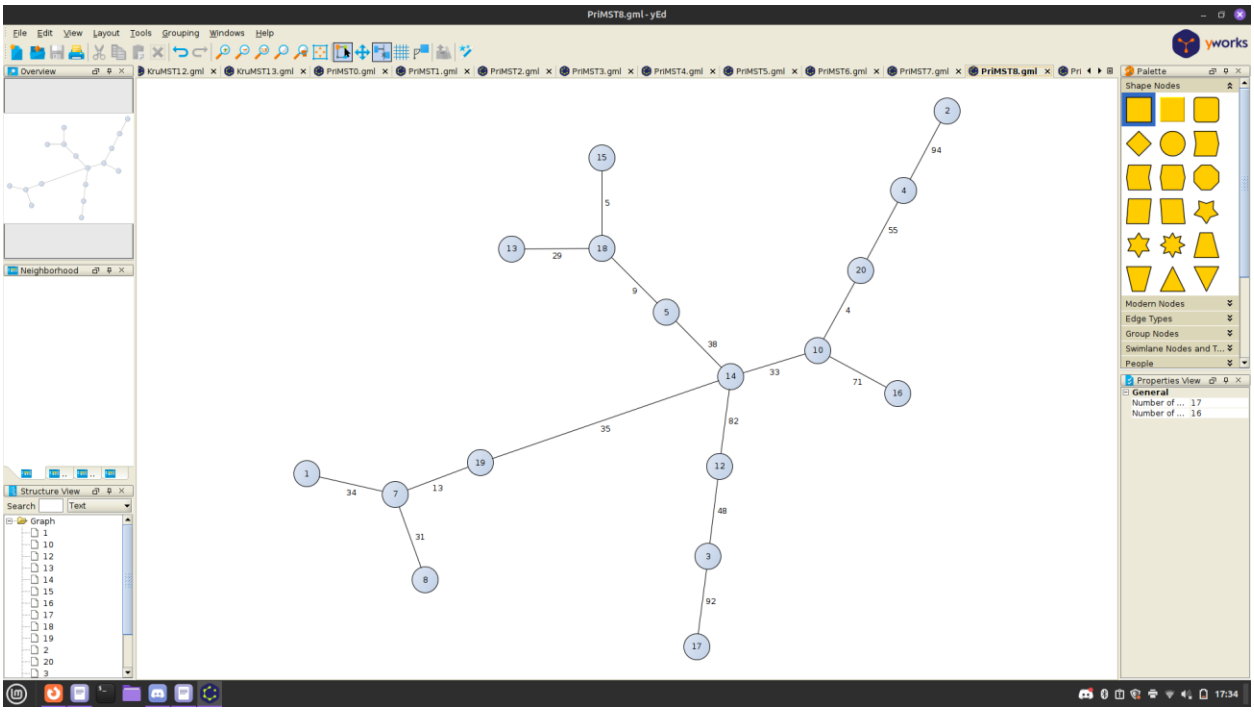
Still identical

Round 8

Kruskal



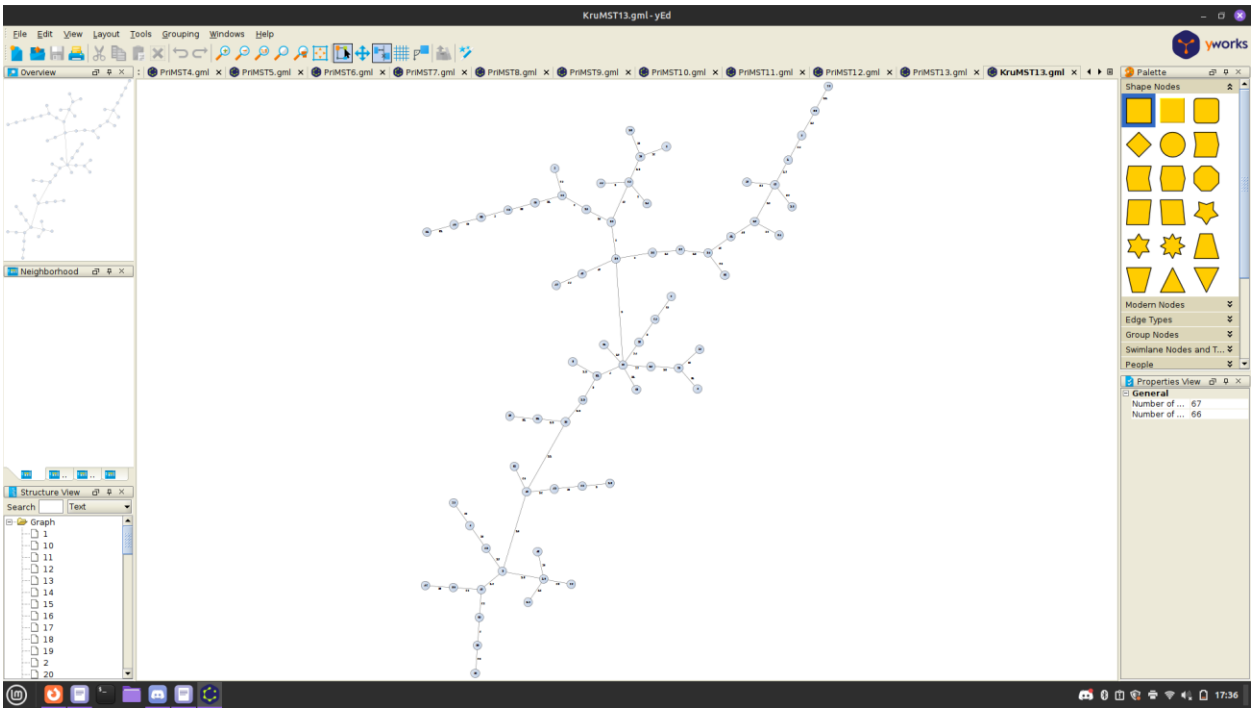
Prim



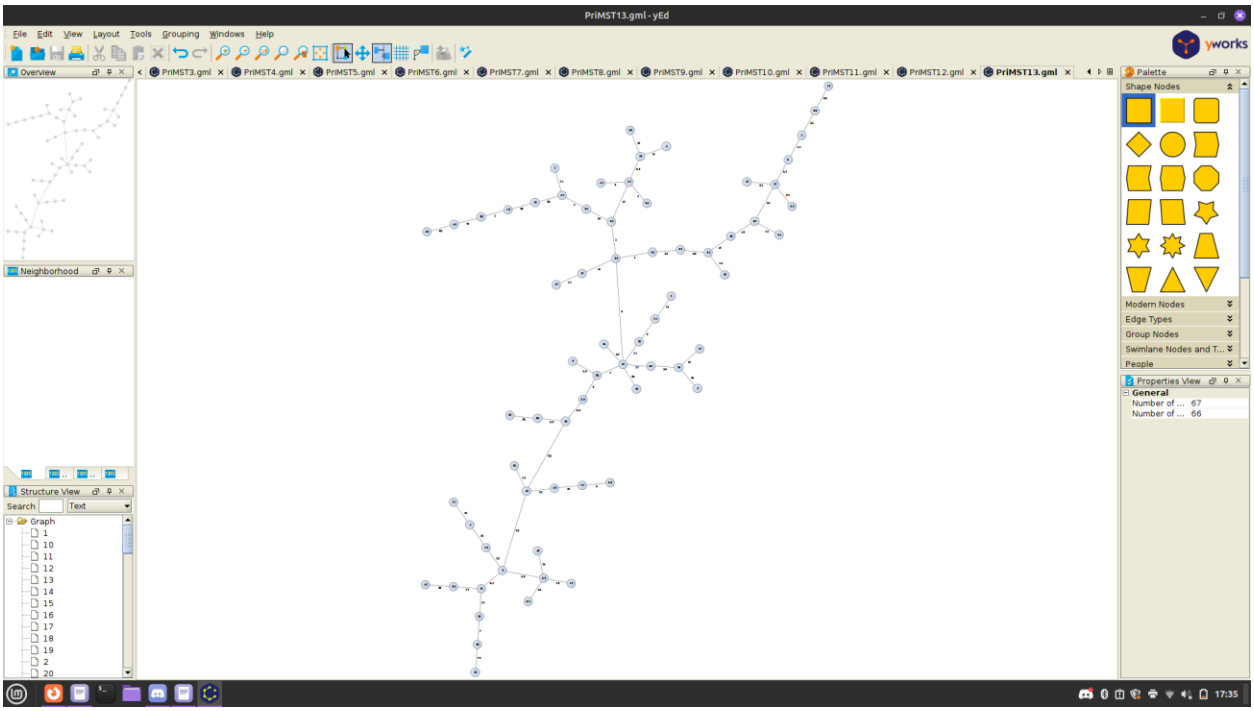
Round 8 this is the reverse test case and although runtime suffered drastically, we are still identical

Round 14 final

Kruskal



Prim



This was the final round, and the tree still looks the same

Conclusion:

In conclusion for my data (I probably did not use big enough graphs) the weights may be different due to the random numbers however the graphs themselves look the same throughout all the test cases. Constant vertices with growing edges ran smoothly with a stable time on average. When we switched to constant edges with growing vertices the run time began to grow exponentially. Overall, it looks like Kruskal's algorithm ran faster than my implementation of prims algorithm.