Project Report

Harman Kumar
2013CS10224

September 17, 2015

1 Testing

The testing of the algorithms was done on the following three kinds of graphs:

1. Very sparse: These graphs had the number of edges very close to the number of verices.

$$E = V + \epsilon$$

$$\epsilon \leq 100$$

2. Sparse graphs: The number of edges were comparable to the number of vertices.

$$E = V * \alpha$$

$$\alpha \leq 10$$

3. Dense graphs: The number of edges was quadratic in the number of vertices.

$$E = V^2/\alpha$$

$$E \leq V * (V - 1)/2$$

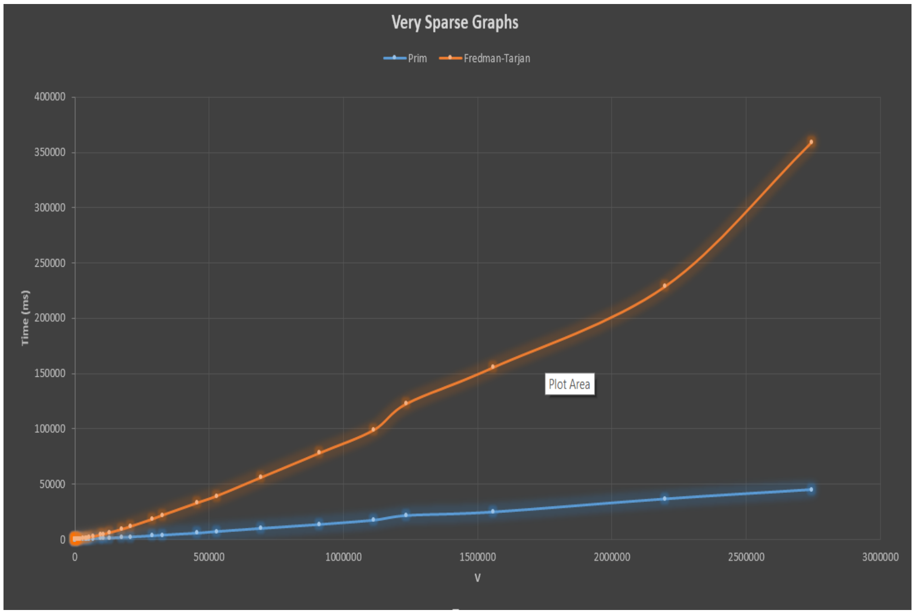
2 Results And Plots:

2.1 Very Sparse Graphs:

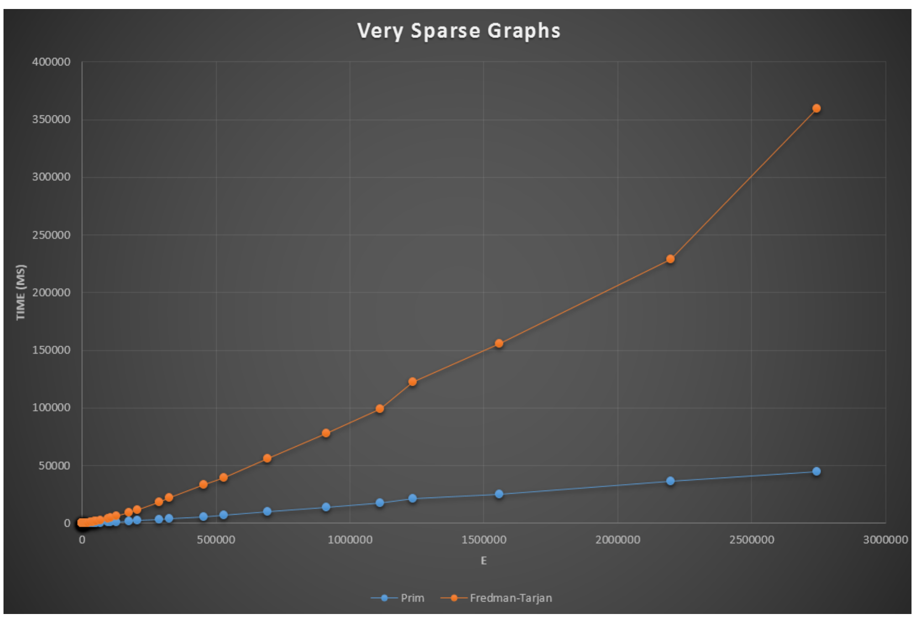
For very sparse graphs,

$$E \approx V$$

Following are the time curves:



Time Vs. V



Time Vs. E

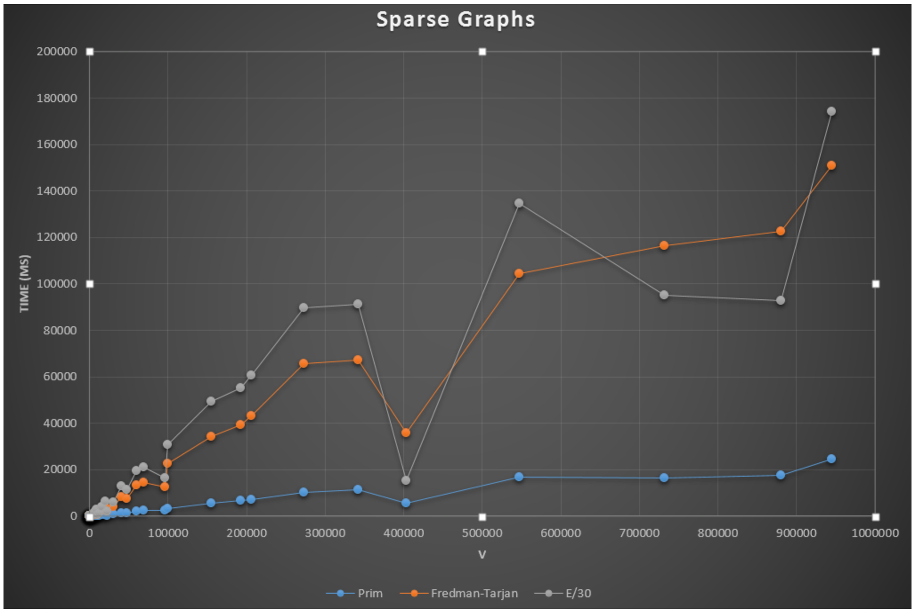
Since $E \approx V$, t is correlated with both V and E.

2.2 Sparse Graphs:

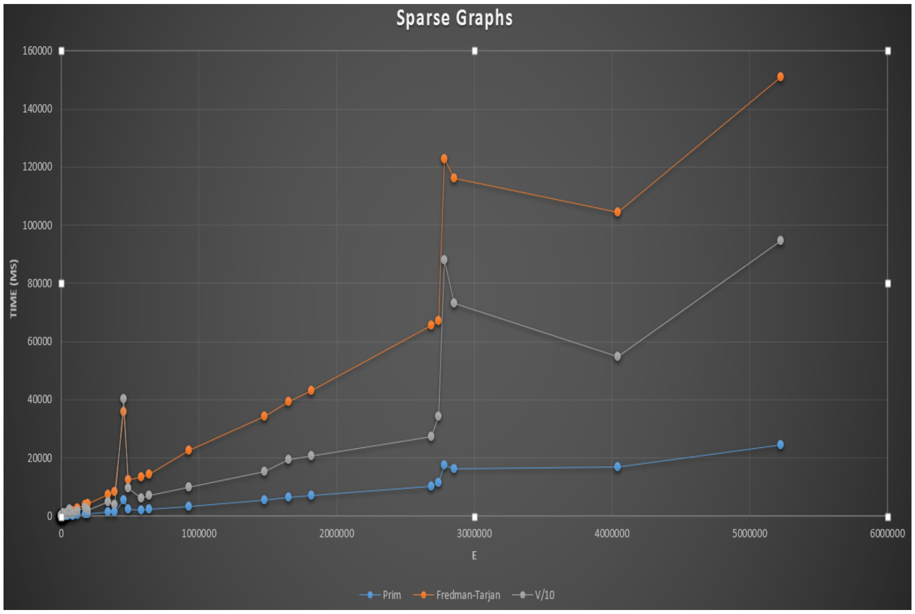
For sparse graphs,

$$E = \theta(\alpha * V)$$

Following are the time curves:



Time Vs. V



Time Vs. E

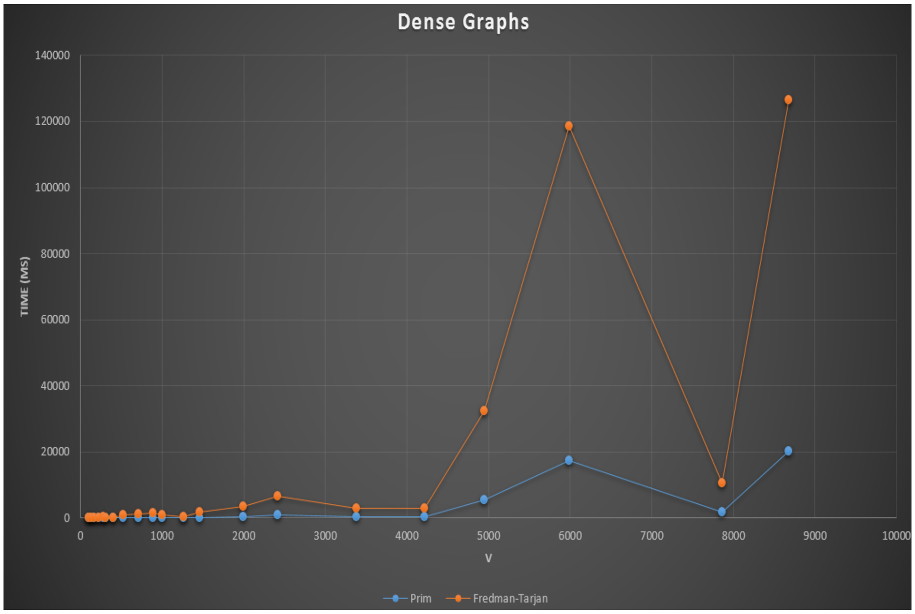
It could be seen that t is neither fully correlated with V, nor with E.

2.3 Dense Graphs:

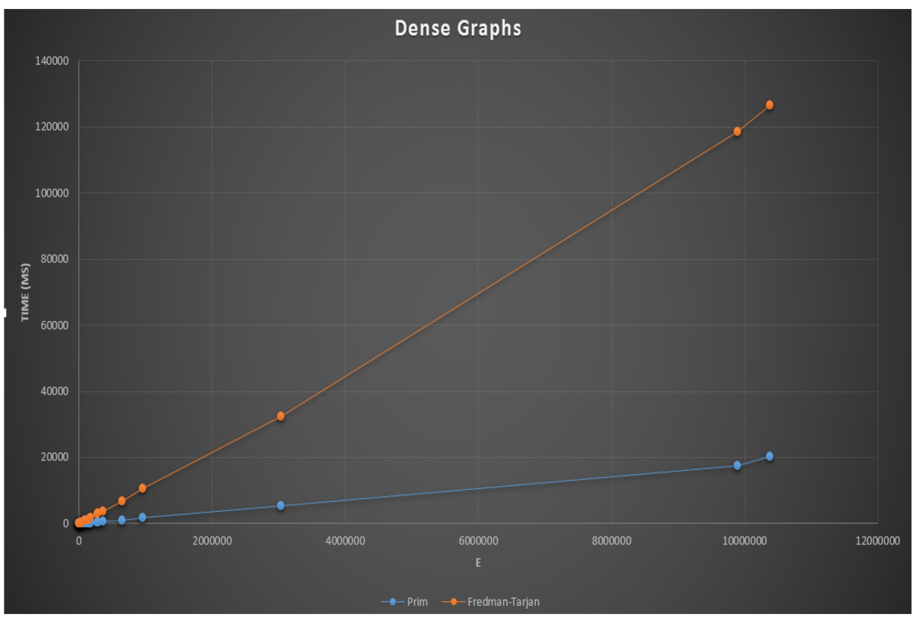
For dense graphs,

$$E = \theta(V^2)$$

Following are the time curves:



Time Vs. V



Time Vs. E

It could be seen that there is no correlation between t and V, whereas t increases linearly with E.

Time complexity for Prims Algorithm:

$$\theta(E + V \log(V))$$

Time complexity for Fredman-Tarjan Algorithm:

$$\theta(E \log^*(V))$$

Since, $E = \theta(V^2)$, E becomes the dominating factor in the time complexities and hence t starts to depend purely on E.

3 Conclusion:

1. As the graph becomes dense, the correlation between V and time to compute MST decreases while the time complexity starts to depend more on E.

2. It was seen that even for huge graphs, the Fredman-Tarjan algorithm using the fibonacci heap data structure took more time as compared to Prims algorithm, using the binary heap.

3. The reason for the poor inferior performance of fredman-tarjan algorithm using the fibonacci heap is that Fibonacci heaps are slow and have significant storage overheads (4 pointers per node, plus a data field and a bool for housekeeping).