Date: 17/Sep/2015

COL702: Advanced Data Structures
Programming Assignment – 1

Submitted By: Rajesh Kedia (2014CSZ8383)

Results and Analysis

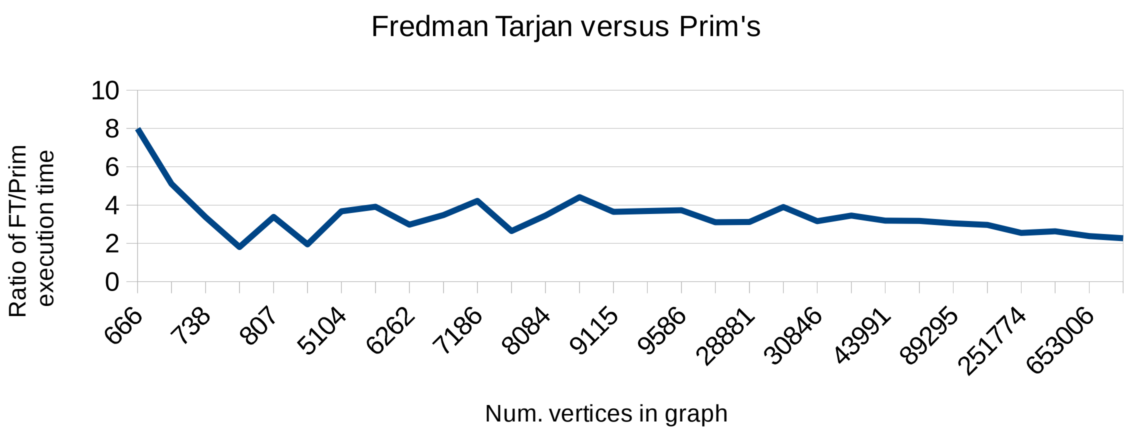
Results:
Current state: The Prim's algorithm is working for all graphs.

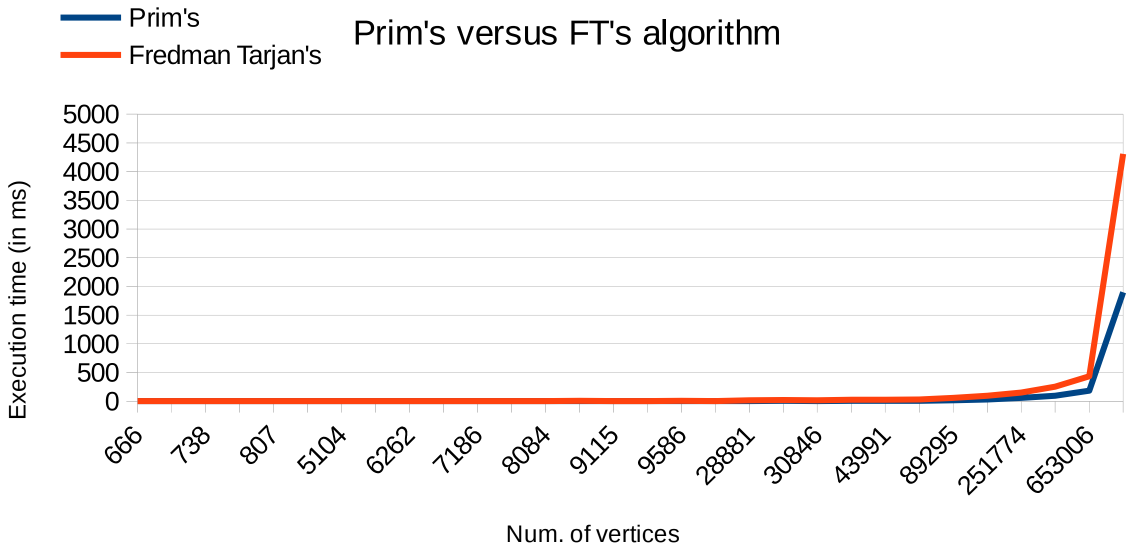
FT's algorithm is working for many of the graphs having low fanout, but the code crashes with Segmentation Fault for cases with high connectivity of graphs. This is possibly due to a bug in the program which couldn't be debugged.

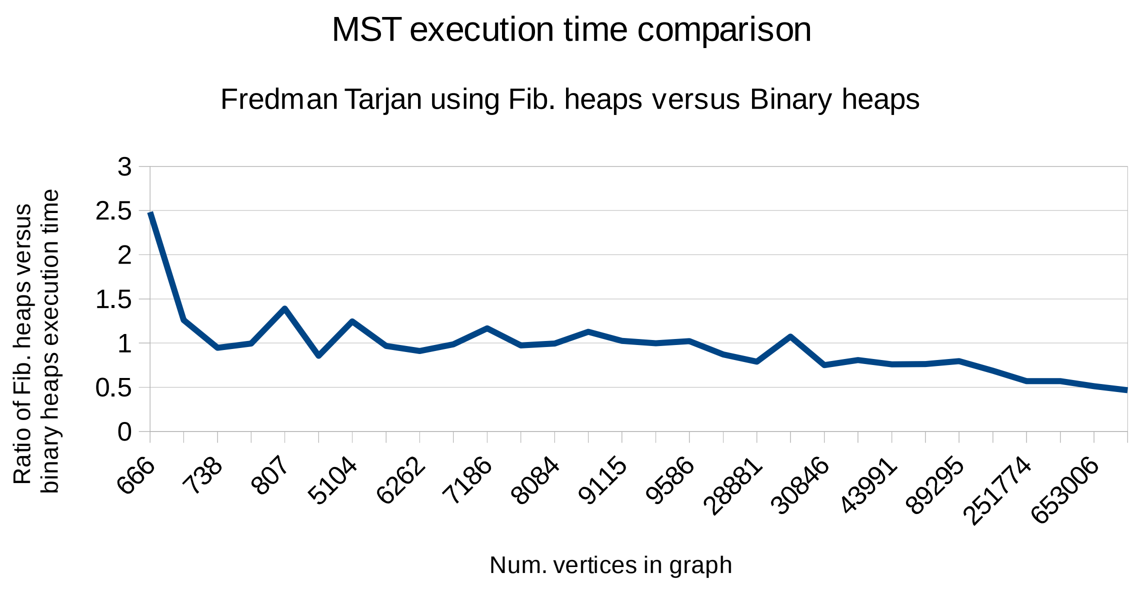
Runtime analysis between Prim's and FT's algorithm:

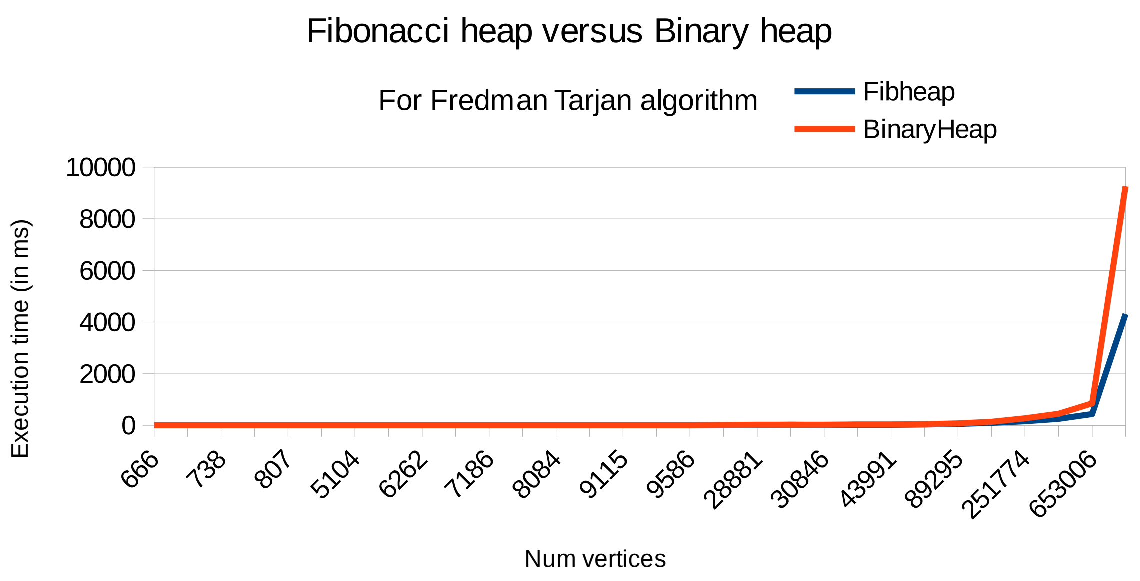
1. As per the below graphs, we clearly see that for smaller input set, the FT's algorithm takes much larger time compared to Prim's algorithm. But with increasing the number of vertices, the ratio is decreasing. In the set of experiments tried out, the ratio hasn't reached 1 yet and remains above 2. This needs further debug and analysis for verifying the implementation. Some of the possible cases are discussed at the end of the document.
2. To compare the efficiency of Fibonacci heaps over Binary heaps, the Fredman-Tarjan algorithm was implemented using both Fibonacci and Binary heaps. The graph clearly indicates that the Fibonacci heaps are efficient compared to Binary heaps for larger graphs and the execution time ratio drops much below 1.

MST execution time comparison









<u>Code bugs discovered as a result of the runtime analysis:</u>

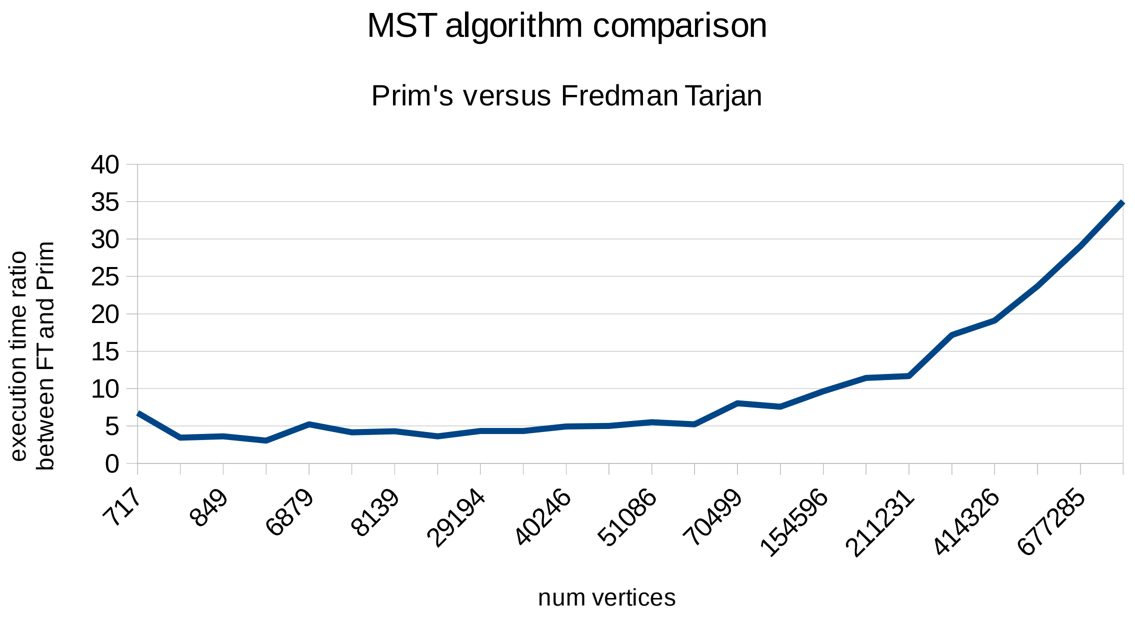
1. At certain places, the loop index was iterated over for larger count than the actual expected counts. By doing analysis using gprof, it was discovered.
2. Careful analysis of source code and paper reveals that certain implementation are not constant time unlike as expected in paper. e.g. delete operation requires adding the child of the node to root list. Though the two lists can be concatenated in constant time, setting the parent of all such nodes as NULL will require to iterate over all the children.

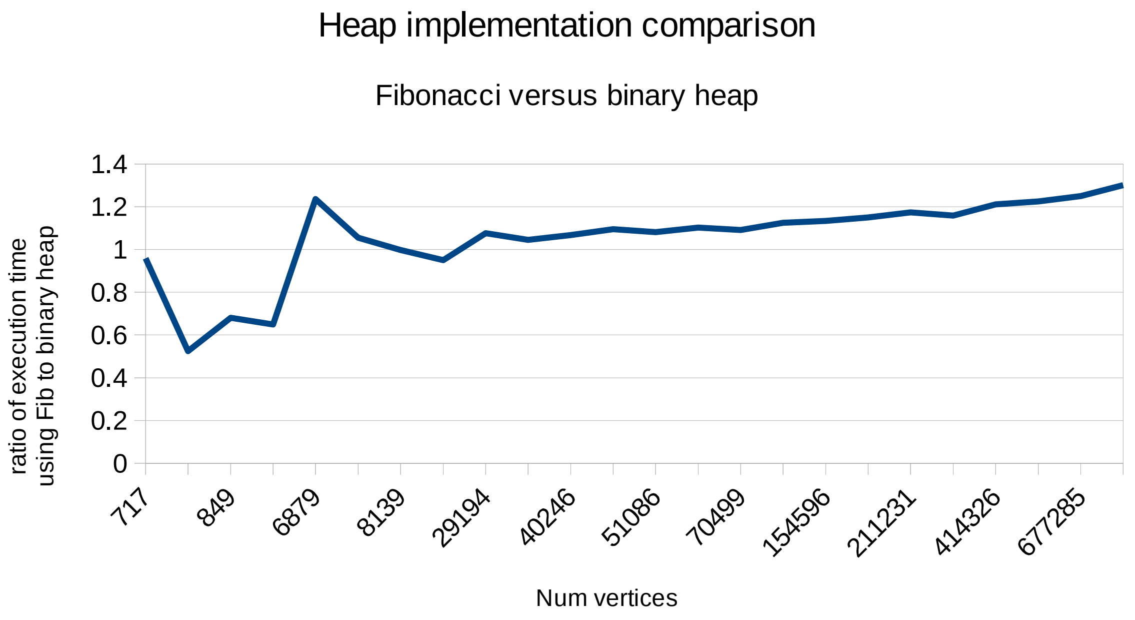
<u>Other factors which may be causing the performance difference:</u>

1. Use of dynamic memory allocation for heap nodes: Since the memory allocation from heap is not within the control of the program and goes through a system call to the OS, it may be introducing additional delays.
2. Use of STLs at certain places for ease of implementation: Again, the exact behavior of such constructs is not known and may be causing this behavior.
3. At certain places, the exact implementation of the algorithm from the paper couldn't be done. Such cases may be causing the additional runtime.

4. The FT algorithm itself has a lots of additional steps to be performed which may be adding significantly to the constant time compared to standard Prim's algorithm.
5. For graphs where multiple iterations of Tarjan algorithm is carried out, the time taken is much larger for FT's algorithm. This clearly indicates some inefficiency in implementing the FT's algorithm.

Below are the sets for which the performance behavior is seen to be completely opposite of the expectation. The only difference between this set and the above set is that these graphs require more than one iteration of FT algorithm to converge.


MST algorithm comparison
Prim's versus Fredman Tarjan
execution time ratio between FT and Prim
num vertices


Heap implementation comparison
Fibonacci versus binary heap
ratio of execution time using Fib to binary heap
Num vertices