CS/SE 2XC3 Lab 8 Report

Glotov, Oleg
L03, 400174037
glotovo@mcmaster.ca

Willson, Emma
L02, 400309856
willsone@mcmaster.ca

March 20, 2022

This report includes the main observations that we found in this week's lab, along with the analysis of our results.

1 Prim's Algorithm

In this section, we discuss Prim's algorithm for finding the minimum spanning tree.

1.1 Prim's Algorithm Version 1

Explanation of how algorithm works and its time complexity

1.2 List vs. Min Heap

The most expensive functions in the implementation of Prim's algorithm are finding and updating the weight of the minimum edge. Our first implementation uses a list of edges that are sorted by weight. The algorithm re-sorts this list for every edge visited. The Python `sort()` function has a time complexity in $O(nlogn)$. Our second implementation uses a heap of nodes that are sorted by edge weight. The heap property is maintained using `extract_min` and `decrease_key`, so there is no need to sort. Both of these functions are in $O(logn)$ The difference between these implementations is that the first one visits each edge, while the second visits each node. As the graph becomes more densely connected, visiting nodes (v2) becomes less efficient than visiting edges (v1). However in a sparse graph with a large number of nodes, using a min heap of nodes is faster. That's why we expect v2 to perform significantly better on large, sparse graphs.

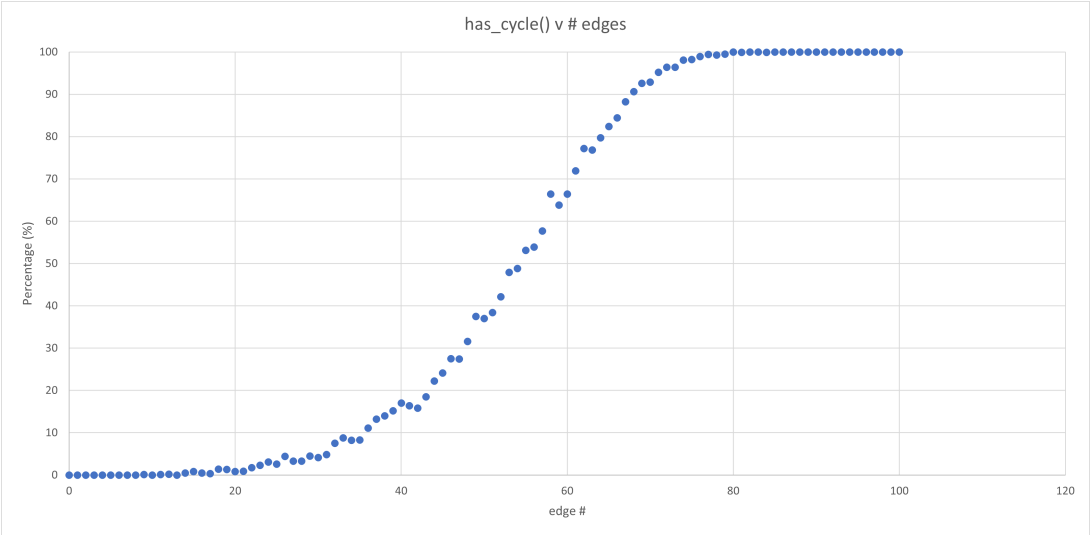

Figure 1: time complexity of prim v1 vs. prim v2

As shown in the graph above, ...