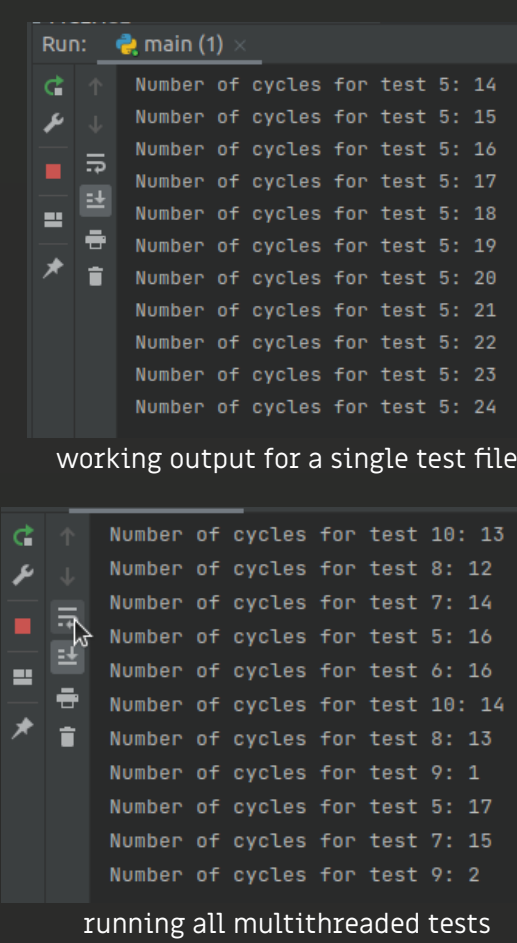


Created by: Darren Ellsworth, Omar Guerra (2021)

We also divide a traditionally larger network into **self-regulating clusters**, which will alleviate memory constraints.

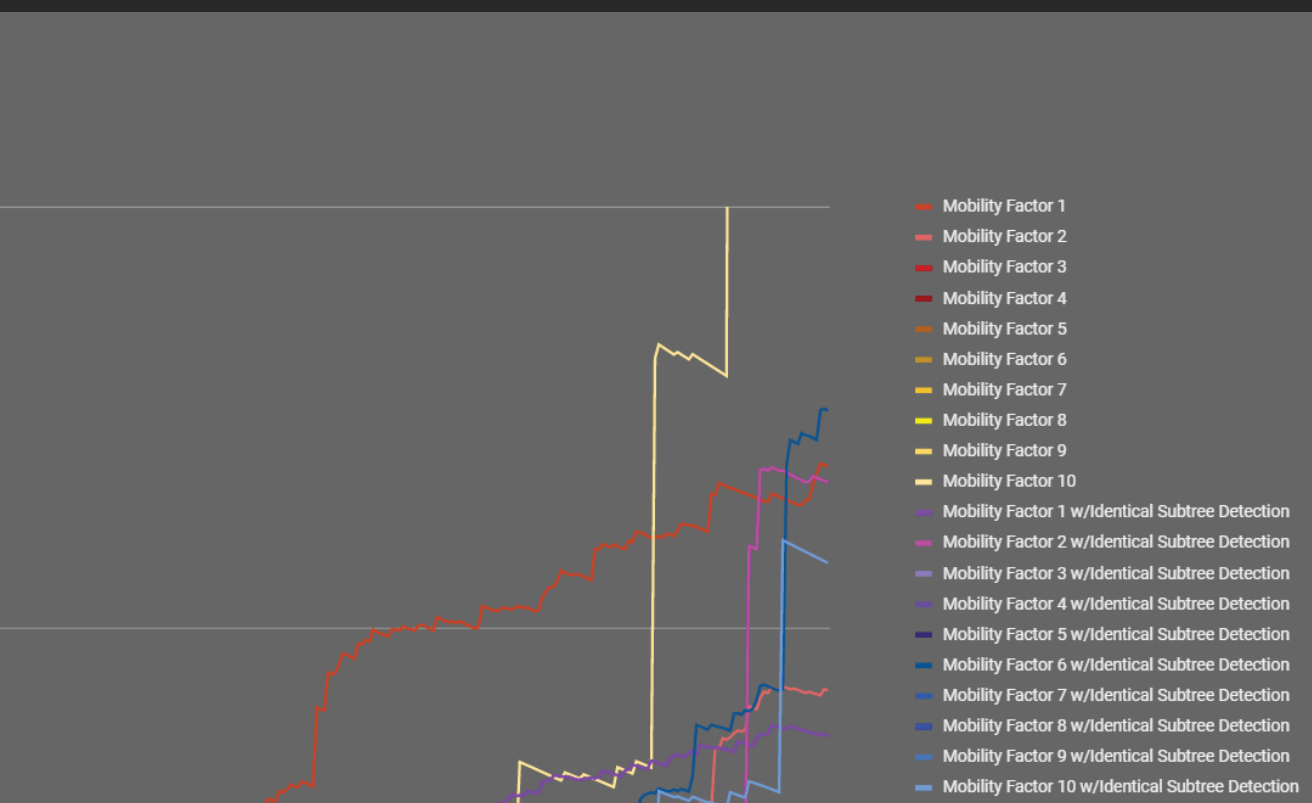
The code will run on Linux and Windows. Once the necessary modules are installed, run the main.py file and the tests will start. **Take care!** The datasets are very large; with enough time the computer running the code will slow down and begin stalling. This is to be expected and is not an issue with the code. For a brief run use test 10. To choose the tests, uncomment them in main.py.



Our NoMerge

As seen in the `rec_merge` code the binary structure of our data is traversed in a traditional way. When an identical subtree is spotted by virtue of hash the merge can return and blocks will not be needlessly counted, thus providing an efficiency gain.

```
def csv_bfprint(root, test_num):  
    """  
    Print an MDG-tree into a .csv file.  
  
    A breadth-first search algorithm modified in order to  
    write the node, its parent, and its subtree hash to a .csv file.  
  
    Parameters:  
        root (kallisto): node representing the root of the tree  
        test_num (int): number of the test that is running the function  
    """  
    temp = [root, "root"]  
    cw = csv.writer(open("{}blockchains_{}.csv".format(test_num, "a"))  
                    , mode="a") as branch_size,  
                  delimiter=";", quotechar="'",  
                  quoting=QUOTE_MINIMAL)  
    quote_id, node_node_id, node_blockchain_blocks_size,  
    blocks_left_size, node_blockchain_blocks_right_size,  
    blocks_height()  
    while temp:  
        i = 1  
        for x in temp:  
            if i % 2 == 1:  
                temp1.append(str(x.data))  
                temp1.append(x.subtree_hash)  
                if x.left:  
                    temp2.append(x.left)  
                    temp2.append(x.data)  
                if x.right:  
                    temp2.append(x.right)  
                    temp2.append(x.data)  
            else:  
                temp1.append(str(i))  
                cw.writerow(list(temp1))  
                temp1.clear()  
                i += 1  
            temp.clear()  
            temp = copy.deepcopy(temp2)  
            temp2.clear()  
            i += 1
```



[2] W. Hou, D. Li, C. Xu, H. Zhang and T. Li, "An Advanced k-Nearest Neighbor Classification Algorithm Based on KD-tree," 2018 IEEE International Conference of Safety and Production Informatization (IICSPI), 2018, pp. 902-905, doi: 10.1109/IICSPI.2018.8690508.

