# Project Report

Ju Cai(UFID: 96691796) [jucai1@ufl.edu](mailto:jucai1@ufl.edu)

In this Java program, I write three classes performing functions.

The first one is treeNode class which defines the node structure discussed in class. It has one constructor to initialize a Node:

treeNode(String theNode, int amount)

And a series of functions to get and modify its properties;

public int getAmount() { return data.getAmount();}  
public String getHashtag() { return data.getHashtag();}  
public int getDegree() { return degree;}  
public void addAmount(int amount) { data.addAmount(amount);}  
public void minusDegree() { degree--;}  
public void addDegree(int amount) { degree += amount;}

The second class is datafield. The main purpose of this class is to package the data field, String hashtag and int theAmount. Every treeNode object has a datafield object as one of its properties. Functions in this class is similar to previous class.

The most important class is MaxFibonacciHeap. Objects in this class contains two private field. One is MaxElement, which is a treeNode type pointer. The other one is hashtable, which is a HashMap used to track arbitrary nodes in a tree.

It provides all functions of a Fibonacci Heap discussed in class. In this scenario, firstly we need to add data we read from input file. When we call addData(String theNode, int theAmount), it will check that if theNode comes in first time or not, and call Insert() or IncreaseKey() for us. Secondly, we need to do query. However, Fibonacci Heap do not offer this function directly. In my program, it will do n times RemoveMax() along with printing hashtag and then Insert() them back to heap.

public void Insert(String theNode, int theAmout)

public void IncreaseKey(String theNode, int theAmount)

public datafield RemoveMax()

public datafield remove(treeNode theNode)

Beside those important functions mentioned above. There are many helper functions in this class either. Among them the pairwideCombine() and cacadingCut() are the most important. The former is needed when we do RemoveMax, which is the only way that one tree/node could become a child of the other one. The latter shall be necessary in doing IncreaseKey(). This is the key operation to make the amortized complexity of IncreaseKey() and other functions get better.

public void pairwiseCombine(treeNode starter)

public void cascadingCut(treeNode starter)

For the whole structure of my program, the logic is easy in the main() function. It is responsible for processing where to read in and where to print out, judging from the parameters it got. The most important job was to initialize a MaxFibonacciHeao object named maxFiheap. And then do what we need with the help of maxFiheap.