COP5536: Advanced Data Structures, Fall 2016

Project Report

Aman Yadav amannagarro@ufl.edu **1157-1388**

A **Fibonacci heap** is a data structure for priority queue operations, consisting of a collection of heap-ordered trees. It has a better amortized running time than many other priority queue data structures including the binary heap and binomial heap. (Ref. Wikipedia)

Various Complexity of Fibonacci heap

	Actual Complexity	Amortized Complexity
Insert	O(1)	O(1)
Remove min (or max)	O(n)	O(log n)
Meld	O(1)	O(1)
Remove	O(n)	O(log n)
Decrease or (Increase Key)	O(n)	O(1)

Compiling Instructions:

The Project has been compiled and tested under the following platforms:

Sr. No.	Environment	Compiler	Pass
1.	Windows	Eclipse	Test passed
2.	Linux	JDK	Test passed

To compile the program in JDK compiler:

1. Open a new project

2. Copy the files into the project.

3. Give the file path in the argument of the project and then run the following code.

To compile the following project on thunder(linux environment) follow the following steps:

1. Open putty and enter thunder.cise.ufl.edu as the server name

2. Copy the files on the server using any filetransfer software like Filezilla or Winscp.

3. run using the following command javac hashtagcounter or use command Make all.

Program Structure:

The project has three classes, hashtagcounter.java FibonacciHeap.java FibonacciNode.java where hashtagcounter.java is the Main function.

This class describes the structure of a node used in the Fibonacci Heap. It contains the following class variables:

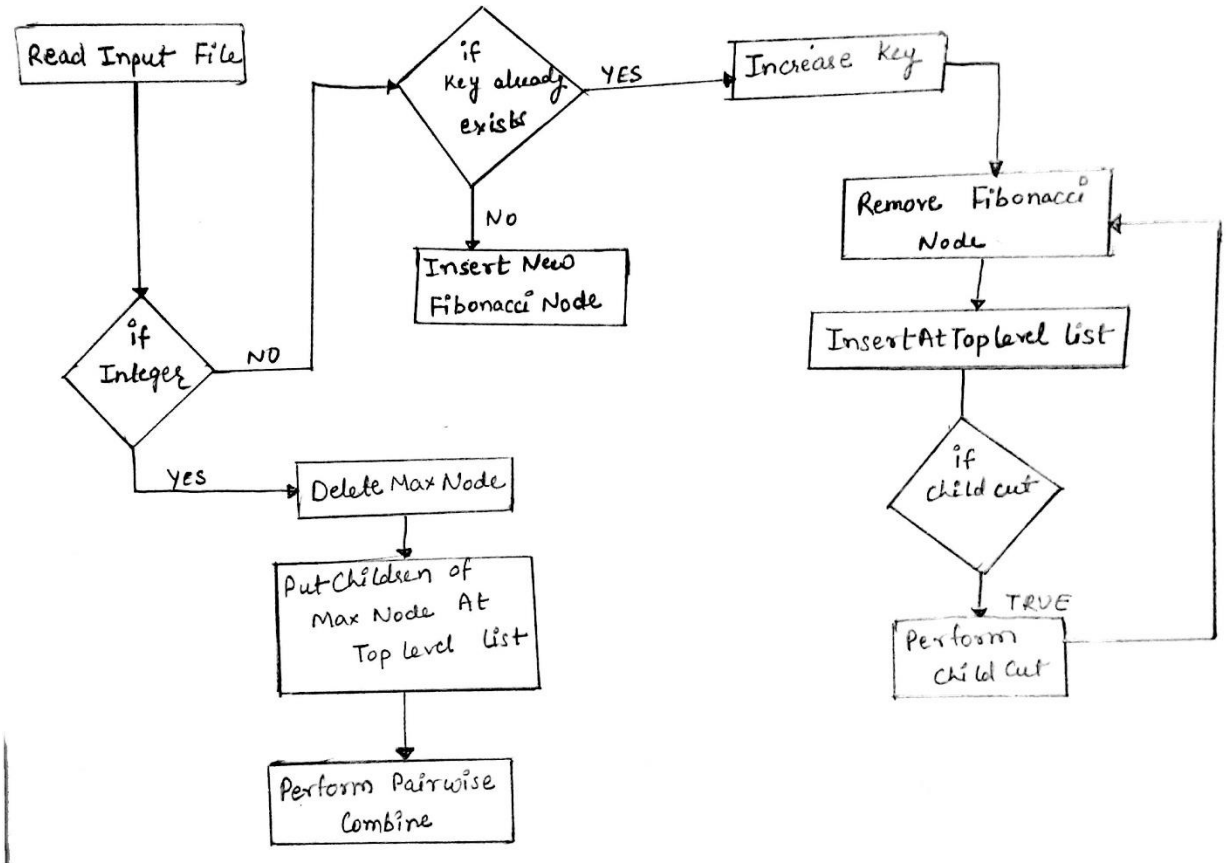
FibonacciNode.java

int count;	-total frequency of Hashtags.
FibonacciNode leftSibling;	- links to the left child of the node.
FibonacciNode rightSibling;	- links to the left child of the node.
FibonacciNode parent;	- links to the parent of the node.
FibonacciNode child;	-links to the child of the node.
String hashtagValue;	-stores hashtag string.
boolean childCut;	-indicates whether parent has lost a child or not.
int degree;	-total number of children of a node.

FibonacciHeap.java

```
static FibonacciNode maxNode;

static HashMap<String,FibonacciNode> hmap = new HashMap<String,FibonacciNode>();

static public void InsertOrIncreaseKey(String hashTagVal, int count)

static public void InsertNewFibonacciNode(String hashTagVal, int count)

static public void IncreaseKey(FibonacciNode node,int count)

static public void InsertAtTopLevelList(FibonacciNode newNode);

static public void RemoveFibonacciNode(FibonacciNode node, FibonacciNode parent)

static public void PerformChildCut(FibonacciNode node)

static public ArrayList<FibonacciNode> GetTopNHashTag(int n)

static public FibonacciNode DeleteMaxNode()

public static void PutChildrenOfMaxNodeAtToplevel()

static public FibonacciNode removeMaxNode()

static public void PerformPairWiseCombine(ArrayList<FibonacciNode> topLevelNodesList)

static public FibonacciNode AddNodeToSiblingList(FibonacciNode parent,FibonacciNode
node)

static public FibonacciNode CreateNewFibonacciNode(String hashTagVal, int count)
```

Flow Diagram:



Function prototypes:

`public void InsertOrIncreaseKey(String hashTagVal, int count)`

Parameters: String hashTagVal, int count

Return type: void

Description: If key is not present in hashtable then calls InsertNewFibonacciNode function, otherwise calls IncreaseKey function.

`static public void InsertNewFibonacciNode(String hashTagVal, int count)`

Parameters: String hashTagVal, int count

Return type: void

Description: Inserts a new node in hashMap and calls InsertAtTopLevelList function.

`static public void InsertAtTopLevelList(FibonacciNode newNode)`

Parameters: FibonacciNode newNode

Return type: void

Description: Inserts the new node to the right of the Max Node.

`static public void IncreaseKey(FibonacciNode node,int count)`

Parameters: FibonacciNode node,int count

Return type: void

Description: Increases the frequency of the node with given hashtag value and calls RemoveFibonacciNode and PerformChildCut function.

`static public void RemoveFibonacciNode(FibonacciNode node, FibonacciNode parent)`

Parameters: FibonacciNode node, FibonacciNode parent

Return type: void

Description: Remove Fibonacci Node from the current position and calls InsertAtTopLevelList function to add it at top level.

`static public void PerformChildCut(FibonacciNode node)`

Parameters: FibonacciNode node

Return type: void

Description: If child cut is true for the node then calls RemoveFibonacciNode and PerformChildCut function recursively.

`static public ArrayList<FibonacciNode> GetTopNHashTag(int n)`

Parameters: int n

Return type: FibonacciNode

Description: This function is called by Main() when input string is an Integer. This function calls DeleteMaxNode to delete Max Node.

`static public FibonacciNode DeleteMaxNode()`

Parameters: No parameters

Return type: FibonacciNode

Description: This method removes Max node by calling removeMaxNode function and calls PutChildrenOfMaxNodeAtToplevel function and PerformPairWiseCombine function.

`public static void PutChildrenOfMaxNodeAtToplevel()`

Parameters: No parameters

Return type: void

Description: This method removes the children list of Max node and put the children list to the right of Max Node.

`static public FibonacciNode removeMaxNode()`

Parameters: No parameters

Return type: void

Description: This method removes the children of Max node and put it to the right of Max Node.

`static public void PerformPairWiseCombine(ArrayList<FibonacciNode> topLevelNodesList)`

Parameters: ArrayList<FibonacciNode> topLevelNodesList

Return type: void

Description: This method performs pairwise combine after Max Node has been removed and calls AddNodeToSiblingList function.

`static public FibonacciNode AddNodeToSiblingList(FibonacciNode parent,FibonacciNode node)`

Parameters: FibonacciNode parent, FibonacciNode node

Return type: FibonacciNode

Description: This method puts the node in the childlist of the parent node.

`static public FibonacciNode CreateNewFibonacciNode(String hashTagVal, int count)`

Parameters: String hashTagVal, int count

Return type: FibonacciNode

Description: This method create a new Fibonacci Node and assigns the given frequency and hashtag to the new node.