Advances Data Structures Programming Project Report

SAI KIRTHI KASIMI

UFID:1628-6038
skasimi@ufl.edu

Project Description:

The goal of the project is to implement a system to count the n most popular keywords used in a search engine at any given time. Keywords are given as an input file together with their frequencies. We implement using a max priority structure to find the most popular keywords.

The following data structures are used for the implementation:

Max Fibonacci heap: to keep track of the frequencies of keywords
Hash Map: keywords are used as keys for the hash table and value is the pointer to the corresponding node in the Fibonacci heap.

Programming Environment:

Operating System: Windows 10
Language Used: Java
Development Environment: Visual Studio Code
Java Runtime Environment: Jre 1.8.0_121

Compiling and Running Instructions:

The project has been compiled and tested on thunder.cise.ufl.edu and java compiler on local machine.

Instructions:

You can remotely access the server using ssh username@thunder.cise.ufl.edu

Navigate to the directory containing the project.

Command: make (To generate the executable .class file)

Command: java keywordcounter file_name (To run the executable)

Structure of Program:

The project is implemented using a single class called keywordcounter and a subclass called Node which defines the basic structure of a node in a Fibonacci Heap.

Fields in Node class:

Class Variable	Type	Description
parent	Node	This field points to the parent node. Initially it is set to null all nodes in the root list have parent field set to null
child	Node	This field points to the child node. A node can access all its children using this pointer as the children of a node are connected using doubly linked list on either side
leftSib	Node	This field stores the pointer to the left node in the circular list.
rightSib	Node	This filed stores the pointer to the right node in the circular list.
childCut	Boolean	If childCut is false it means that the node has ever lost a child. The childCut is set to true every time a node loses its child.
degree	Node	This field is used to store the

		degree i.e count of the number of children of a node.
key	String	This field contains the keyword.
value	Integer	This field hold the count value of the keyword.

Functions in Node class:

1)**Node(String key, int value)** :
It is a constructor used to initialize key and value fields.

Global declarations:

static HashMap<String,Node> hashmap=new HashMap();

This HashMap is declared globally and stores the key(keyword) and the Node.

public Node maxNode=null;

A global variable maxNode of type Node is declared. maxNode is initially set to null and points to the node that has maximum count value. This is declared globally as it can be used in methods of the main class.

public int numNodes=0;

A global variable numNodes of type Integer is declared. numNodes is initially set to zero and it is used to keep track of number of nodes in the currently in the heap. Every time a new node is inserted in the heap its value increases by one and decreases by one every time a node is removed.

Function Name	Return Type	Description
public static void main(String args[])	void	This function read an input file containing keywords and their count. Based on the read input and certain conditions it calls functions like insertNode(), increaseKey(), removeMax(). It also puts and gets values from the HashMap.
public void insertNode(Node newNode)	void	This function is used to insert Node into the max Fibonacci Heap. If the maxNode is null then the newNode becomes the maxNode else the newNode is added to the right of the maxNode and maxNode is reassigned if necessary after comparing their values.
public void increaseKey(Node n, int i)	void	The value of Node n is changed to i. If n.value is greater than current maxNode.value then n becomes the new maxNode. Values of n and n.parent are checked and if n.value is greater than n.parent.value then n is cut from its parent and cascadingCut is performed on n.parent.
public void cut(Node x, Node y)	void	Remove Node x is child of Node y. Assign y.child if previously x is y.child. Add Node x to the rootlist of nodes, set parent of Node x to null, set childCut of Node x to false.

public void cascadingCut(Node x)	void	If the childCut of Node x is false it changes it to true.If the childCut of Node x is true then it performs cut and cascadingCut recursively till it finds a Node whose childCut is false.
public Node removeMax()	Node	The maxNode in the Fibonacci Heap is removed. All the children of maxNode are removed and placed in the rootlist of nodes and all the equal degree rootlist nodes are merged by calling mergeEqualDegree().
public void mergeEqualDegree()	void	This function merges nodes of equal degree.
public void makeChild(Node x,Node y)	void	This function removes Node y from the rootlist of nodes. Make Node x child of Node y this increases the degree of Node y. Mark the childCut of Node x to false.

Results:

The output is stored in a file called ouput_file.txt. The program is a successful implementation of Max Fibonacci Heap.