Advanced Data Structures (COP 5536)

Fall 2018

Programming Project Report

Sai Charan Kadari

UF-Id: 5174-9229

skadari@ufl.edu

Project description:

The goal of this project is to implement a system to find the n most popular keywords appeared on social media such as Facebook or Twitter. For the scope of this project keywords are taken as an input file. Basic idea for the implementation is to use a max priority structure to find out the most popular keywords.

The project uses the following data structure.
1. Max Fibonacci heap: Use to keep track of the frequencies of keywords.
2. Hash table (Hash Map in java): Key for the hash table is keyword and value is pointer to the corresponding node in the Fibonacci heap.

Max-Heap is implemented in java and the address of all the nodes is stored in a Hash Map.

Fibonacci Max Heap	
Amortised Complexity	
Space	O(1)
Search	O(1)
Insert	O(log n)
Delete	O(1)
Find Max	O(1)
Delete Max	O(log n)
Increase Key	O(1)
Merge	O(1)

Compiling and running instructions:

The program has been compiled and tested in local machine with javac as the compiler.
To execute the program:

- Extract the contents of the zip file
- Type ‘make’ without the quotes in command line.
- Type ‘java keywordcounter ‘file path/input_file_name.txt’ ’without the quotes.

Structure of the program and method descriptions:

The program consists of two classes:

- keywordcounter.java – This file contains the main class to read and write data. Also calls FibonacciHeap class for doing all the operations.
- Fibonacciheap.java – Contains all the methods needed to operate on and maintain the fibonacci heap.

The basic workflow of the program is as follows:

- keywordcounter.class takes input file and reads it.
- It then checks the input and if the input starts with a $, it checks if there is an element with that name in hash table. If it doesn't find the element it inserts it into the heap and table. If it finds the element, it performs increase key operation on the heap.
- If the input is the number, it does extract max operation on the heap based on the inputs and prints in the output file and inserts them back into heap.
- This is done until 'stop' is reached in input file.

The detailed overview of all the class functions is given below.

1. Keywordcounter.java

Variable	Type	Description
path	String	File name from input argument
file	file	To create an output file
br	BufferedReader	To read data from input file
writer	BufferedWriter	To write data into output file
s	String	Contains input line read from file
p1	Pattern	To recognize the keyword
p2	Pattern	To recognize the query
heap	FibonacciHeap	Object of FibonacciHeap
m1	Matcher	To match keyword in input data
m2	Matcher	To match query in input data
St	String	Keyword from input
num	Int	The number of times extract max must be performed
rnodes	node	Array of nodes that have been removed.
output	String	Stores all the extracted values in

		correct format.
h	HashMap	Hash map to perform all operations.

2. Node

Variable	Type	Description
degree	Int	To store degree of a node
child	Node	Store address of child node
freq	Int	Value of the node (KeyWord)
left	Node	Store address of leftNode
ight	Node	Store address of rightNode
parent	Node	Store address of parent node
childCut	Char	Store childCut value
name	String	Store keyword to which this node belongs to.

3. Fibonacciheap.java

Class variables	Type	Description
max	Node	Pointer to node with max value

a. public void insert (node n)

Description	Inserts a new node to the heap	
Parameters	node	Node that must be inserted
Return value	void	

b. public node extractMax()

Description	Extracts the max element from the list and calls pairwisecombine if necessary.	
Parameters	None	
Return value	node	Max node

c. public void pairwisecombine ()

Description	Merges all heaps with similar degrees and makes a new Fibonacci heap with distinct degree heaps.	
Parameters	None	
Return value	void	

d. public void increaseFrequency(node n, int value)

Description	Increases frequency of the node and calls cut() and cascadingCut() methods if necessary	
Parameters	N	Node whose frequency must be increased
	value	New frequency of the node
Return value	none	

e. public void cut(node parent, node child)

Description	Inserts all children of max to top level circular list	
Parameters	parent	Parent of the node that is removed from heap
	child	Root of the heap that must be removed
Return value	Void	

f. public void cascadingCut(node temp)

Description	Recursively removes heaps until childcut value is false	
Parameters	temp	Node on which cascadingCut() is applied
Return value	Void	

Results:

The code was tested for the given sample input and accurate results were achieved

Conclusions:

The problem statement of the project has been solved by implementing a FibonacciHeap to find out the top N most popular words. We used extractMax and IncreaseFrequency methods of Fibonacciheap to achieve the desired functionality.