PROJECT REPORT

Name: Xiao Hu UFID: 69363179 MAIL:xiao.hu@ufl.edu

1. Environment

Linux & C++

2. Program structure

FibonacciHeap.cpp => FibonacciHeap.h => hashtagcounter.cpp

3. FibonacciHeap.h

Classname: FibNode	

public	This file used to define classes and declare functions.
	`string key; //hashtag`
	`int freq; // frequency`
	`int degree; // Degree of one node`
	`FibNode* left; // left sibling of the node`
	`FibNode* right; // right sibling of the node`
	`FibNode* child; // first child of the node`
	`FibNode* parent; // The parent of the node`
	`bool marked; // True or false used in removemax`
	
	`FibNode(string keyword, int val) // Create and initialize a FibNode.`
	`{`
	`    key = keyword;`
	`    freq = val;`
	`    degree = 0;`

	marked = false;
	left = this;
	right = this;
	parent = nullptr;
	child = nullptr;
	}

FibNode is a Fibonacci heap node class which can contain a lot of information

(Key,freq) is the information stored in the Fibnode as a hashtable.

Degree indicates the degree of one node.

Mark is used to cascading cut.

Classname:FibHeap	

Public	Declare all Fibnacciheap operations
	// Create and initialize an empty Fibonacci Heap.
	FibHeap()
	{
	keyNum = 0;
	maxDegree = 0;
	max = nullptr;
	cons = nullptr;
	}
	~FibHeap() {}
	FibNode* insert(string key, int freq);
	void update(map<string, FibNode*> &table, string key, int newfreq);
	FibNode* removeMax();

Private	int keyNum; //Total number of the key nodes
	int maxDegree; // Max degree of the heap
	FibNode *max; // The max root value of the heap
	FibNode **cons; //When deleting the node, cons can store the node temporarily
	
	void removeNode(FibNode* node);
	void cascadingCut(FibNode* node);
	void increase(FibNode* node, int freq);
	void consolidate();
	void makeCons();
	FibNode* extractMax();
	void link(FibNode* node, FibNode* root);

4. FibonacciHeap.cpp

FibHeap::insert(string key, int freq)		
Parameters	string key	hashtag
	int freq	frequency number of the hashtag
Return value	The new inserted node	

Insert the fibnode to the fibonacci heap. If the hashtag is not in the heap then insert the node to the heap use addnode method and if the (new node->freq) > (formal max node ->freq) then max point to the new node.

FibHeap::removeNode(FibNode* node)		
Parameters	FibNode* node	node that will be deleted.
Return value	void	

Remove the node from the circular linked list, same to the remove node in the double link list.

FibHeap::cascadingCut(FibNode* node)		
Parameters	FibNode* node	node that will be cut.
Return value	void	

After calling cut we should check the mark of the parent node. If the parent node's mark of the cut node is true then we should cascading cut. CascadingCut use cut until the false node

FibHeap::increase(FibNode* node, int freq)		
Parameters	FibNode* node	node that will be increased.
	int freq	frequency that will be increased.
Return value	void	

Increase the frequency. If the hashtag of the new node already exist in the heap, just increase the corresponding frequency.

	FibHeap::update(map<string, FibNode*> &Hashtable, string key, int newfreq)	
Parameters	map<string, FibNode*> &Hashtable	hashtable used to find the aim node.
	string key	keyword of the aim node
	int newfreq	Update the new node’s frequency to new frequency
Return value	void	

Update the frequency in the node.

	FibHeap::extractMax()	
Parameters	none	none
Return value	the extracted node.	

Extract the max node from the Fibonacci Heap.

	FibHeap::link(FibNode* node, FibNode* root)	
Parameters	FibNode* node	node that will be linked.
	FibNode* root	the root for node to link.
Return value	void.	

Link the node to the root.

	FibHeap::makeCons()	
Parameters	none	none
Return value	void.	

Create the area for consolidation.

FibHeap::consolidate()		
Parameters	none	none
Return value	void.	

Consolidate the Fabonacci Heap which has the same degree.

FibHeap::removeMax()		
Parameters	None	none
Return value	the address of the max node that just be removed.	

Remove the max value in Fibonacci Heap.

5. hashtagcounter.cpp

split(const string& s,vector<std::string>& v, const string& c)		
Parameters	**const string& s**	The string that will be splited
	vector<std::string>& v	Store the final string to the vector
	const string& c	According c to split the string
Return value	The vector v that store the final string.	

split the string

StoreInfib(string buffer, map<string, FibNode*> &table, FibHeap &fibHeap)		
Parameters	**string buffer**	A line of hashtag and its frequency.
	map<string, FibNode*> &table	used to store keywords and frequency so.
	FibHeap &fibHeap	Used to call the functions of the fibnacciheap to insert of update the node.
Return value	Void.	

store the hashtag and the keynumber to the fibheap

onlyinput(string input_file)		
Parameters	**string input_file**	The name of the inputfile(argv[1])
Return value	return the temporary variable to the main function and then ouput it to the console.	

When not in the case of the specified output file, store the results after removemax to a temporary variable and then ouput it to the console.

withoutput(string input_file,string output_file)		
Parameters	string input_file	The inputfile(argv[1])
	string output_file	The outputfile(argv[2])
Return value	void.	

In the case of the specified output file, get the results after removemax to a temporary variable and then write it to the outputfile.

main(int argc, char argv)**		
Parameters	Int argc	The number of the parameters
	char argv**	The parameters.
Return value	void.	

Main function