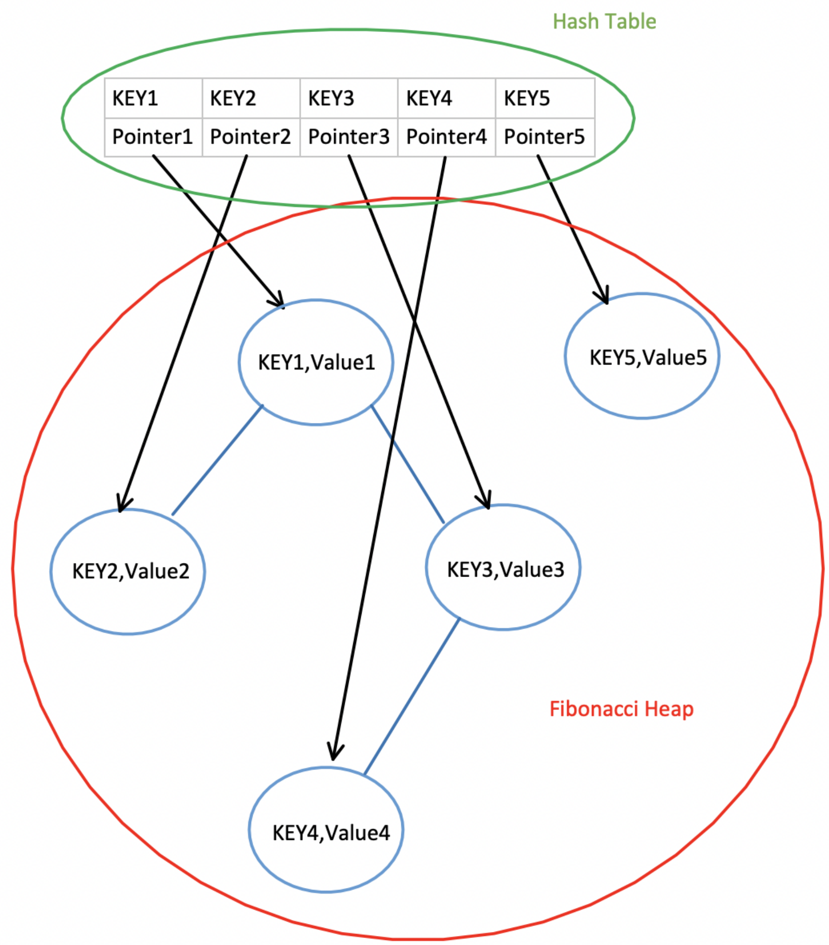
Report of the Advanced Data Structure

December 29, 2019

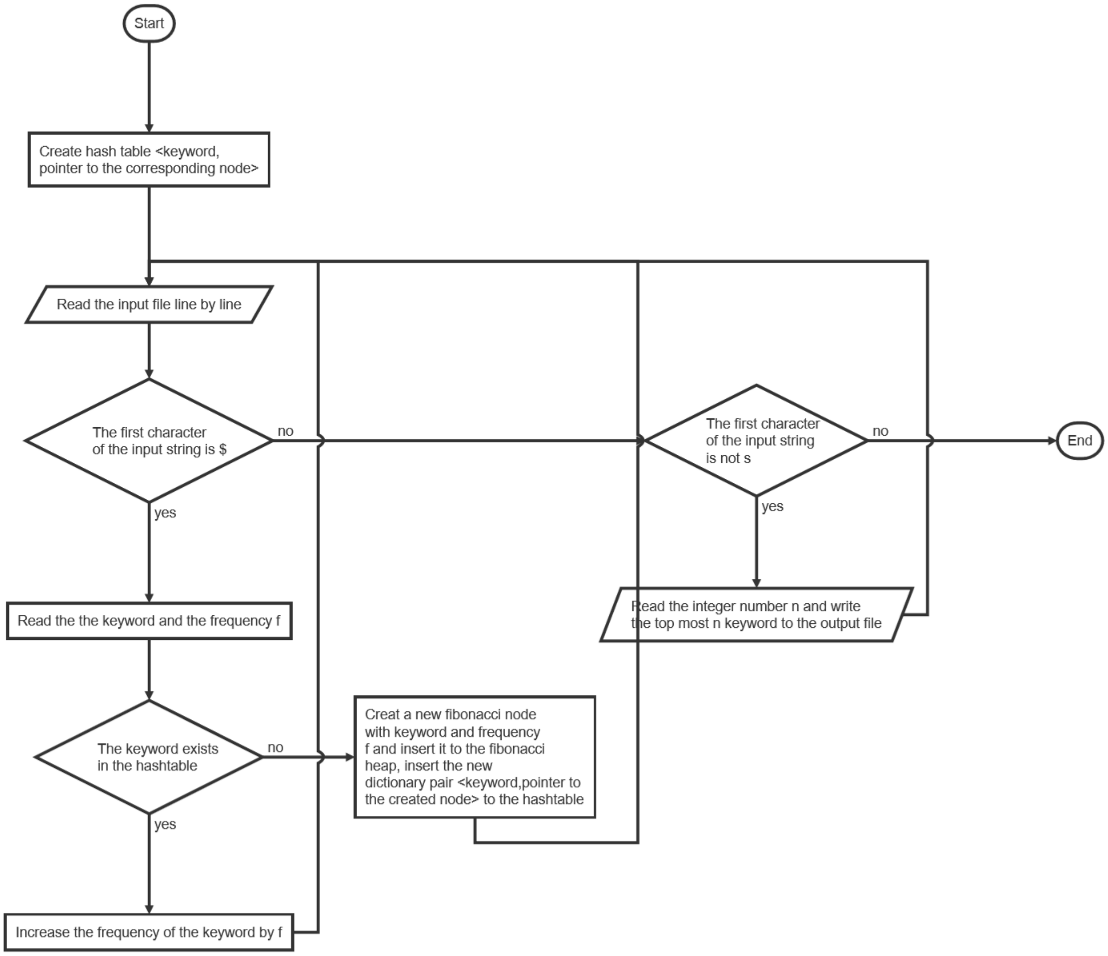
1 The Structure of My Algorithm



KEY1	KEY2	KEY3	KEY4	KEY5
Pointer1	Pointer2	Pointer3	Pointer4	Pointer5

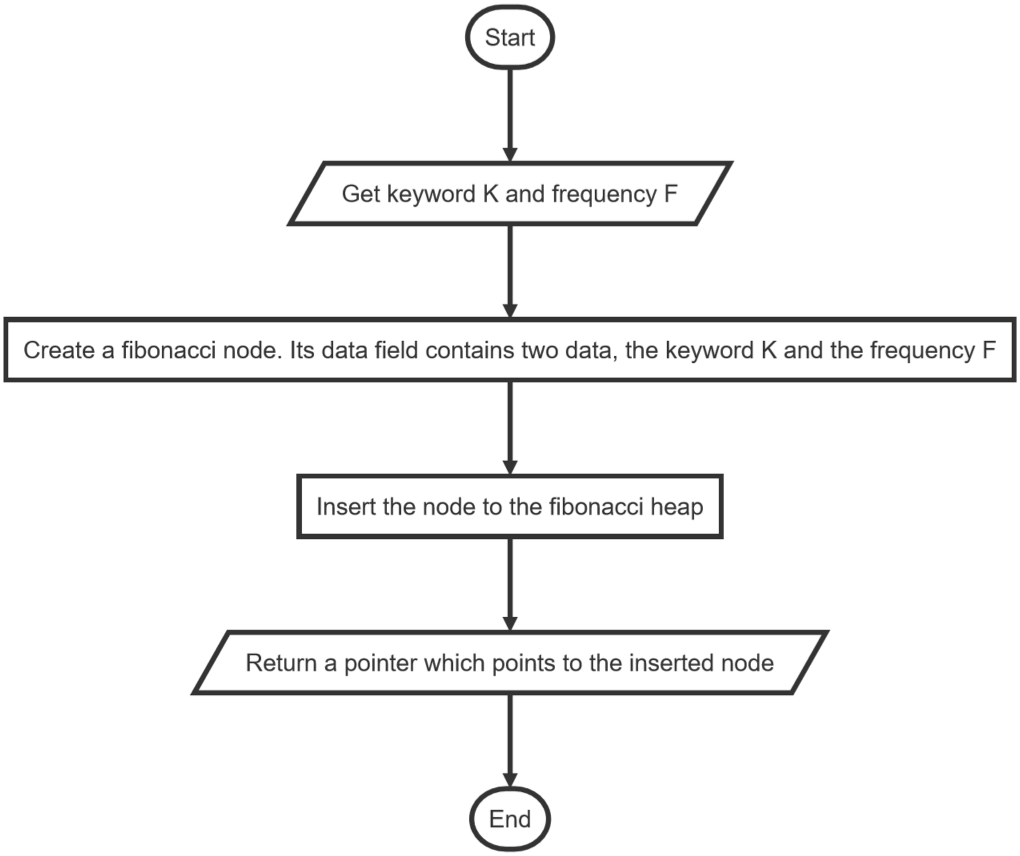
2 Flow Chart of Required Actions

2.1 the Whole Algorithm

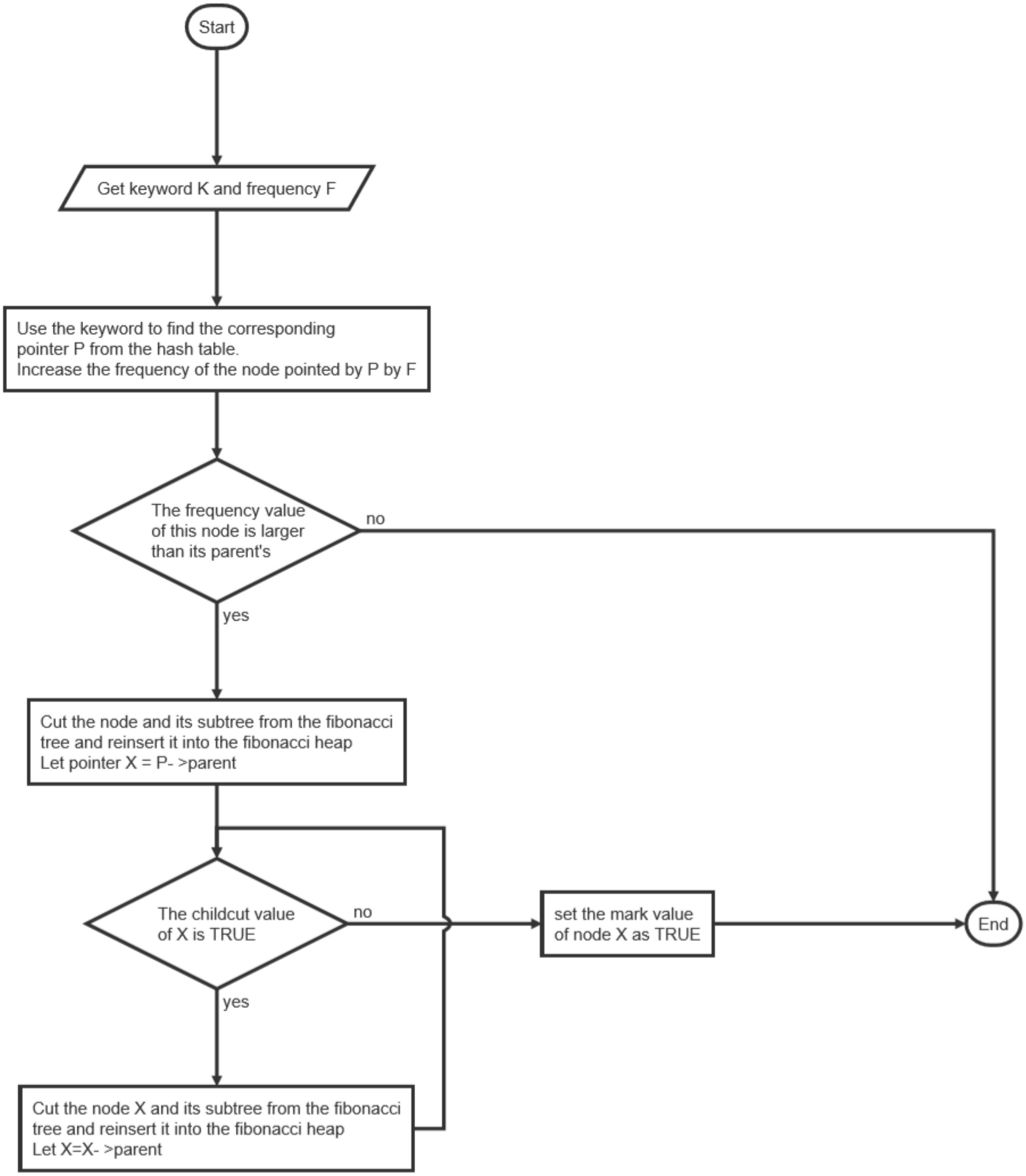


Start
Create hash table <keyword, pointer to the corresponding node>
Read the input file line by line
The first character of the input string is $
no
yes
Read the the keyword and the frequency f
The keyword exists in the hashtable
no
yes
Increase the frequency of the keyword by f
Creat a new fibonacci node with keyword and frequency f and insert it to the fibonacci heap, insert the new dictionary pair <keyword,pointer to the created node> to the hashtable
The first character of the input string is not s
no
yes
Read the integer number n and write the top most n keyword to the output file
End

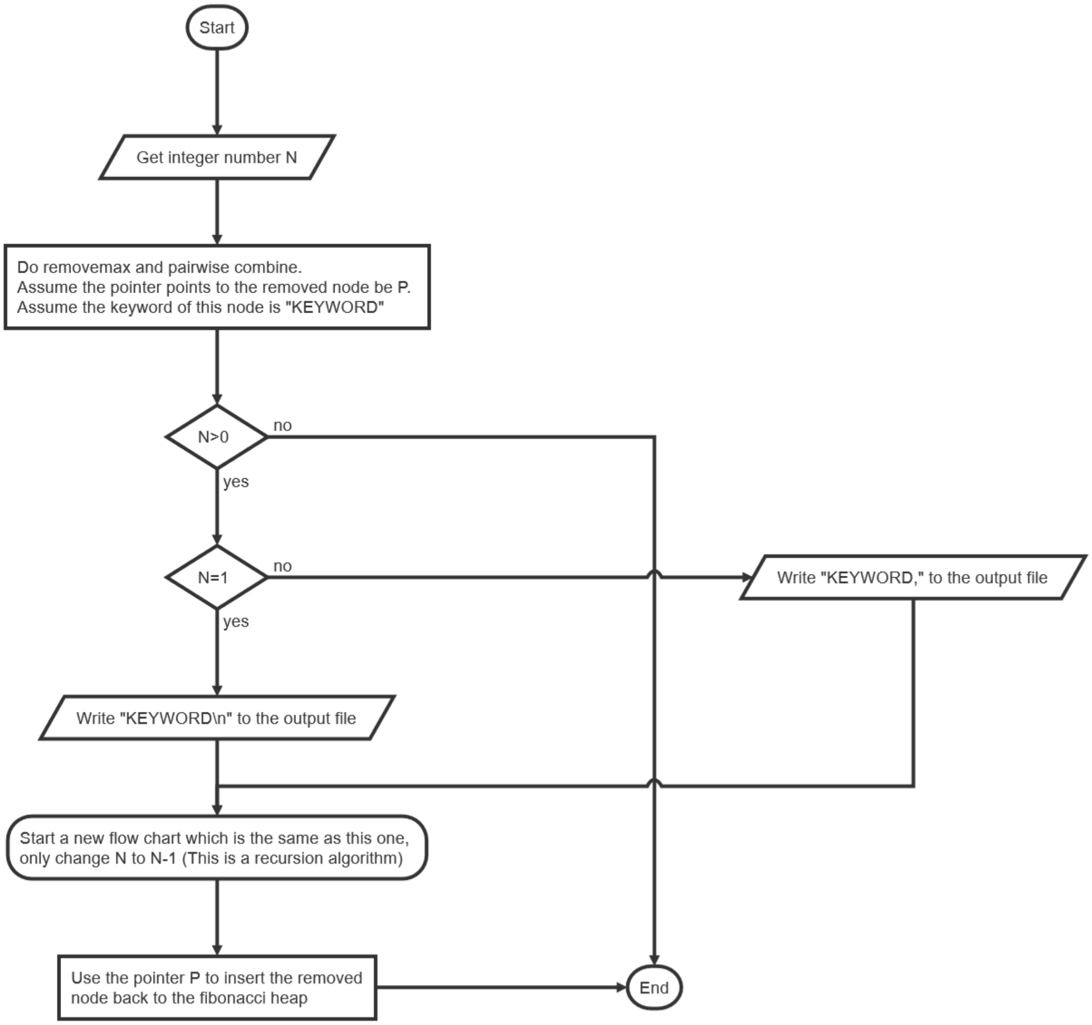
2.2 Insert


Start
Get keyword K and frequency F
Create a fibonacci node. Its data field contains two data, the keyword K and the frequency F
Insert the node to the fibonacci heap
Return a pointer which points to the inserted node
End

2.3 Increase key value


Start
Get keyword K and frequency F
Use the keyword to find the corresponding
pointer P from the hash table.
Increase the frequency of the node pointed by P by F
The frequency value
of this node is larger
than its parent's
no
yes
Cut the node and its subtree from the fibonacci
tree and reinsert it into the fibonacci heap
Let pointer X = P- >parent
The childcut value
of X is TRUE
no
set the mark value
of node X as TRUE
End
yes
Cut the node X and its subtree from the fibonacci
tree and reinsert it into the fibonacci heap
Let X=X- >parent

2.4 Find Top Most N Keywords


Start
Get integer number N
Do removemax and pairwise combine.
Assume the pointer points to the removed node be P.
Assume the keyword of this node is "KEYWORD"
N>0
no
yes
N=1
no
yes
Write "KEYWORD," to the output file
Write "KEYWORD\n" to the output file
Start a new flow chart which is the same as this one,
only change N to N-1 (This is a recursion algorithm)
Use the pointer P to insert the removed
node back to the fibonacci heap
End

3 Class FibNode

FibNode	Define the class of Fibonacci Node	
Data	int value	record the frequency of every keywords
	string key	record the keywords, such as ”youtube”, ”facebook”, etc
int degree	the degree of every node	
bool mark	the child cut indicator	
pointer	FibNode *left	point to its left sibling
	FibNode *right	point to its right sibling
	FibNode *parent	point to its parent
	FibNode *child	point to one of its children

4 Class FibHeap

FibHeap	Define the class of Fibonacci Heap	
int num	number of nodes in the heap	
int maxdegree	the maximum degree of the heap	
pointer	FibNode *max	point to the maximum node
	FibNode **cons	use for pairwise combine

4.1 Functions Required by the Project

All the functions in this section are belong to class FibHeap.

4.1.1 Public Functions

FibNode* insert(int value, string value2)	Insert a node with value ”value” and key ”value2” into the heap and return the pointer points to the inserted node.	
input parameter	int value	record the frequency of the keyword
	string value2	record the keyword
return parameter	FibNode*	A pointer which points to the inserted node

FibNode* removemax()	Remove the max node and return the pointer points to the removed node.	
input parameter	Null	Null
return parameter	FibNode*	A pointer which points to the removed node

void increase(FibNode *node, int value)	Increase the value of a node by ”value”	
input parameter	int value	record the frequency of the keyword you want to increase
	FibNode *node	points to the node you want to increase
return parameter	NULL	

void maximumn(int a, ofstream &s)	Find top most ”a” of nodes, print the key value of these nodes to the output files	
input parameter	int a	number of maximum nodes you want to find
	ofstream &s	use to print the output
return parameter	NULL	

4.1.2 Private Functions

void insert(FibNode *node)	Insert the node pointed by the pointer to the heap. This function is used by the public function ”insert”	
input parameter	FibNode *node	A pointer points to the node you want to insert
return parameter	NULL	

addNode(FibNode *node, FibNode *root)	Add a node before the root, this function is used by the private function ”void insert(FibNode *node)”	
input parameter	FibNode *node	A pointer points to the node you want to add
	FibNode *root	A pointer points to the root node.
return parameter	NULL	

void removeNode(FibNode *node)	Remove a node and its subtree from its link list.	
input parameter	FibNode *node	The node you want to remove
return parameter	NULL	

void makeCons()	Create a space for the function ”void consolidate()”
input parameter	NULL
return parameter	NULL

void consolidate()	Use pair wise combine to fix the degree of the heap
input parameter	NULL
return parameter	NULL

FibNode* extractmax()	Extract the fibonacci tree pointed by the max pointer from the fibonacci heap, this function is used for pair wise combine	
input parameter	NULL	
return parameter	FibNode*	pointer points to the extracted fibonacci tree

void combline(FibNode* node, FibNode* root)	Let the FibNode* node be the child of FibNode* root, this function is used for pairwise combine	
input parameter	FibNode* node	the fibonacci tree whose root value is smaller than FibNode* root’s root value
	FibNode* root	the fibonacci tree whose root value is bigger than FibNode* node’s root value
return parameter	NULL	

void cut(FibNode *node, FibNode *parent)	Remove a node and its subtree from its tree and add the node to the top level link list of the fibonacci heap	
input parameter	FibNode* node	the node you want to cut
	FibNode* parent	the parent node of the node you want to cut
return parameter	NULL	

void cascadingcut(FibNode *node)	cascadingcut	
input parameter	FibNode* node	if the ”mark” value of the node is TURE, then you need to do cascading cut, if not, set the ”mark” value of the node to TURE
return parameter	NULL	

4.2 Functions not Required by the Project

All the functions in this section are belong to class FibHeap. These functions are used for test.

4.2.1 Public Functions

void maximum(int &a, string &b)	Find the key and value of the maximum node, save them at ”a” and ”b”	
input parameter	int &a	the address of ”a”, used to save the ”value” of the maximum node
	string &b	the address of ”b”,used to save the ”key” of the maximum node
return parameter	NULL	NULL

void print()	Print the whole heap
input parameter	NULL
return parameter	NULL

4.2.2 Private Functions

void print(FibNode *node, FibNode *prev, int direction)	print the detail of a heap	
input parameter	FibNode *node	the node you are now printing
	FibNode *prev	the node's parent or brother
	int direction	If direction = 1, FibNode *prev is the FibNode *node's parent. If direction = 2 , FibNode *prev is the FibNode *node's brother
return parameter	NULL	NULL