ADVANCED DATA STRUCTURES PROJECT REPORT

Name: Saideep Korrapati UF Id: 9234-8134
Mail: saideepkorrapati@ufl.edu

Functions Used in the Program:

1. public HeapNode insertNewNode(String s, int count)

insertNewNode method is used to set the default values of the attributes of the node.
Parameters : String s (name of node), int count (value of node)
Return : Returns the object of the newly inserted node.

2. public HeapNode insertNode(HeapNode node, boolean existing) {

insertNode method is called when newnode is being inserted into the heap or after doing the increase key operation to insert back the freed node back into the heap.
Parameters : HeapNode node(node to be inserted), boolean existing(is present)
Return: Returns object of inserted node.

3. public void increaseCount(HeapNode node, int val)

This method increases the value of node when it is re-encountered. If the increased value is greater than the parent node then the node is removed and reinserted back into the heap.
Parameters: HeapNode node(node to increase), int val(value to add)
Returns : None

4. public HeapNode removeMaxNode()

This method removes the maximum element from the heap and return it. It first removes the children of the max and re inserts them into the heap. Now combine is done on each node with equal degree and maxpointer is updated.
Parameters: None
Returns : Returns max element in heap structure

5. private void combineHelper()

The combineHelper method is called after performing the removemax operation. It combines the nodes in the top most layer of the heap such that no two nodes have same degree.

Parameters : none

Return type: none

6. private void meld(HeapNode[] nodes, HeapNode temp) {

This method calls the isDegreePresentInHeap method to check if a node with same degree already exists . If exists it calls the combine method and joins the two nodes with the same degree recursively.

Parameters: HeapNode[] nodes(array) , HeapNode temp

Return type: none

7. private boolean isDegreePresentInHeap(HeapNode[] nodes, HeapNode temp) {

This method checks if degree of a particular node already exists in the array and returns a boolean.

Parameters:HeapNode[] nodes(array), HeapNode temp

Returns: boolean

8. private HeapNode combine(HeapNode node1, HeapNode node2)

The combine method joins two nodes of same degree. The smaller value node is made the child of the larger value node.

Parameters:HeapNode node1(node to combine1), HeapNode node2(node to combine2)

Returns : parent (node)

9. private void removeNode(HeapNode node)

This method helps to remove the node from the heap by updating the node attributes such as nodes right, parent and left nodes.

Parameters : HeapNode node(node to be removed)

Return type:none

10. private void performCascadeCut(HeapNode node, boolean isFirstCut)

performCascadecut method is called after the increase key operation is performed. Checks the childcut value of parent and performs operations accordingly. Removal of nodes is called recursively until we encounter a childcut value as true.

Parameters:HeapNode node(node to be be cascadecut), boolean isFirstCut

Return type:none

Structure of the program :

Input to the program is given from a file which contains words starting with $ and querying numbers (number of remove max operations to perform) and stop.

If the input encountered is not stop and it begins with $ we check the hashmap to see if the given input already exists or not.

If it's a new input we call insertNewNode() function and insert the incoming node into hashmap and heap.

If it's an already encountered input we call the increaseCount() and update the node value.

If the input from file is not starting with $ and its a integer. We perform those many removemaxNode() opeerations on the heap structure and append the top values of those removemaxNode operations i.e (if 3 remove max operations are called we show the top 3 highest frequencies in the loop)

If the input is stop then we come out and end.