COP 5536 Spring 2017

Programming Project Report

Vaibhav Biyani

UFID:1198-8235

Performance Comparisons of three Priority Queues to generate Huffman Tree:

For all the three Priority Queues the tree generations was done and was run 10 times and the time taken for all generations was then divided by 10. The time taken was as below(all times are in milliseconds):

Note: The time taken is the time to insert all the frequency and then generate Huffman tree. (Not printing the Huffman values Just generating and implementing the priority queues)

1)Priority Queue implementation using Binary min Heap:

The time taken was 6385.0 milliseconds for 10 iterations for Binary min heap.

Runtime of each of the 10iterations

656, 652, 632, 639, 636, 635, 635, 633, 632, 633

Complete runtime

6385.0

2) Priority Queue implementation using 4way optimised Heap:

The time taken was 6093.0 milliseconds for 10 iterations for 4way min heap.

Runtime of each of the 10iterations

612, 607, 631, 569, 599, 611, 596, 583, 687, 592

Complete runtime

6093.0

3) Priority Queue implementation using Pairing Heap:

The time taken was 10534.0

milliseconds for 10 iterations for 4way min heap.

Runtime of each of the 10iterations

1127, 1155, 1095, 1171, 1139, 1147, 1108, 1193, 1144, 1150

Complete runtime

10534.0

Based on the above comparison 4way heap was found to be the most efficient way of generating the Huffman Tree and encoding and decoding was done using it.

Implementation of all the three priority Queues:

Binary Minheap:

This is implemented in a class known as Minheap.java:

The functions in binary Minheap are:

public void initialinsert(**int**[] integers):

This function inserts all the non zero elements into a heap and builds heap first time and insert() is called.

public void insert (Node item):
This function inserts into heap and checks continuously with parent and swaps until the parent is smaller than child.

public void generateHuffman():

This function calls 2 extractMin() operations and add it to a Node type object. A new node is created which has the frequency of 2extract min(). Node is a class with left, right, name & frequency.

public void extractMin():

Removes the smallest element which is the top of the array and then calls minheapify() function.

public void minheapify():

Again, makes a min heap and restores the minheap property, checks with the 2 children and compares the smaller one.

public void getmin():

Return the first element of the list which is the smallest.

private int right(int n):

Returns the right child of integer is n which is 2n+2 in arraylist.

private int left(int n):

Returns the left child of integer is n which is 2n+1 in arraylist.

public void buildheap()

Calls minheapify() from n/2 to 0 size.

4-ary heap cache optimized:

This is implemented in dary.java class

public void initialinsert(**int**[] integers):

This function inserts all the non zero elements into a heap and builds heap first time and insert() is called.

public void insert (Node item):
This function inserts into heap and checks continuously with parent and swaps until the parent is smaller than child.

public void generateHuffman():

This function calls 2 extractMin() operations and add it to a Node type object. A new node is created which has the frequency of 2extract min(). Node is a class with left, right, name & frequency.

public void buildheap()

Calls minheapify() from n/4 to 0 size.

public void getmin():

Return the first element of the list which is the smallest.

public void extractMin():

Removes the smallest element which is the top of the array and then calls minheapify() function.

public void generateHuffman():

This function calls 2 extractMin() operations and add it to a Node type object. A new node is created which has the frequency of 2extract min(). Node is a class with left, right, name & frequency.

public void minheapify():

Again, makes a min heap and restores the minheap property, checks with the 4 children and compares the smaller one and restores the heap.

Pairing heap:

This is implemented in PairHeap.Java class

This class methods are called using phpmain which than calls initialinsert and generateHuffman().

public void initialinset(int[] array):

All the non zero elements are inserted to build a Pairheap initially.

Public PairNode insert(int x):

Inserts elements in the existing Pairing heap or creates a new one. If existing calls meld().

private PairNode meld (PairNode first, PairNode second):

Melds two pairing heap after comparing the parents.

public int deleteMin():
The root is the smallest element after removing that combineSiblings() is called and returned.

private PairNode combineSiblings(PairNode firstSibling)
2way merging and done. If the number of trees are odd, we get the last one and returns the root of the PairingHeap which is still the minimum.

encoder.Java:

This is the main class from where all the methods are called such as MinHeap, daryHeap and Pairing Heap which is all called with an argument which is an array consisting of the initial frequency values.

Next the encoder.java writes code_table.txt and encoded.bin using the Vector of result which is returned by the best performing priority queue.

decoder.Java:

The decoder.java reads the code_table and encoded.bin which is passed to the decoded.java class which returns the arrraylist result which is finally written into the decoded.txt file.

Decoded.java

Creates the decoder tree and than visits the tree for all the encoded.bin bits and return the arraylist consisting all the values which are visited and are at the leaf level.

The complexity of the decoder is O(n) for reading the encoded.bin file where n is the total number of bits in the encoded.bin file and O(nh) where n is the number of messages(i.e. frequency count) and h is height of the decoded tree which is O(logn).