|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Huffman Coding  Data Structures and Operating Systems |  |
|  |  |  |  |
|  | | | |

By: Vickshan Vicknakumaran

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | | |
|  |  | TABLE OF CONTENTS | |  |
|  |  | Contents  [1](#_Toc92890369)  [1](#_Toc92890370)  [TABLE OF CONTENTS 2](#_Toc92890371)  [SECTION 1 3](#_Toc92890372)  [SECTION 2 5](#_Toc92890373)  [References 7](#_Toc92890374) |



|  |  |  |
| --- | --- | --- |
| SECTION 1 |  | |
| Encode  The function of the encode method was to create a Huffman tree using the methods given in the Classes. Based on the Huffman theory, I had to start with the frequency table. I used the “FreqTable” method to include an if statement and a for loop. The worst-case time-complexity of this is O(n) as the iteration used inside the method all have O(n) as their time complexity. Using the frequency table, I had to create a Huffman tree. Therefore, I had used the “treeFromFreqTable” method to include a new Priority queue using the “PQueue” Class, a for loop to create and add a new leaf (Using the Leaf Class) to the priority queue and a while loop to create a branch using the branch Class. The worst-case time-complexity of this is O(n\*2). This is because I have used a while loop which has a nested for loop.  Basing on the Huffman tree theory, I had to implement the traverse method to go through the Huffman tree which is made. To achieve this, I had to add traverse method into the leaf and branch classes. The traverse method in the Branch class is responsible for mapping the characters and its path. As it is using if statements to condition the binary code, it has a time-complexity of O(n) as well. In the Leaf class, I had created a new HashMap where it would store the pair of the character (the leaf label) and the list of Booleans (paths of the leaf). Due to this, the time-complexity would be O(n) as it includes insertion to the HashMap. | |  |
| Next, I had to traverse through the tree. Therefore, I used the buildcode method to create an empty array and inherit the treverse method onto the array. The time complexity of this method is O(n) as I used if statements and an array.  The encode methods gathers up all the other methods for the whole huffman tree process including the FreqTable, treeFromFreqTable and the PQueue. In order to do this, I had set each process to a variable and used a for loop to encode the input data and add the code of the character to a list. The time complexity of this method is O(n) as the previous methods all have the same time complexity. | |  |
|  | |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
| Decode  The function of a decode method is to decode data using the characters and code. To do this, I have used the “treeFromCode” method as it will be responsible of reconstructing the tree using the Characters and their codes. I have used a while loop nested in a for loop to go through the characters and the codes. The worst time-complexity of this method is O(n^2) as the while loop loops for each for loop. Next in the decode method, I had used a for statement to go through each item in data list and an if statement to find whether a Branch and a Leaf is present in CurrentNode. The worst time-complexity is O(n^2) as it uses the treeFromCode method which has nested loops.  To implement a fully working compression/decompression tool that store compressed data in binary codes is by giving the ability to input their own strings and save their output. | | |  |
| Text  Description automatically generated  Figure 1 - encode method | Text  Description automatically generated  Figure 2 - Decode method | |  |

|  |  |  |
| --- | --- | --- |
| SECTION 2 |  | |
| Data Structure and Algorithms  A data structure is used to store and arrange data in the most efficient and effective way. An algorithm is a set of instructions followed to solve a certain problem. In this report, I will be explaining some of the examples that implement data structures and algorithm in the operating system.  A Buddy System is an algorithm that splits larger memory blocks into small blocks called as buddies to fulfil the request. The buddy system uses a binary algorithm to “indicate whether the block is free.” (Doeppner, 2010). Based on the address of the block, it compares the block size and sets it to a binary. “If the block is of size 2^k, then the rightmost k-1 bits of its address are zeros. The next bit (to the left) is zero in one buddy and one in the other:” (Doeppner, 2010). The binary is then used to get the buddy’s address. It uses two types of data structures. An array and a linked list. It uses array list to store a linked list of free blocks which contain the power of 2 and the free blocks are linked using the field. “An array of doubly linked lists of free blocks, one list for each power of two ….. linked together using fields within them” (Doeppner, 2010).  A Hash Function maps input to a fixed size value and uses it as an index in the hash table. Hash functions are used in data structures such as page tables, file systems and directory listings. The hash function is used in the page table to hash the virtual pages in the virtual addresses higher than 32 bits. These hash data that are stored in a hash table have a linked list of elements that hash to the same location. (Hashed Page Table In Operating Systems, 2020). The hash function is used in file systems to prevents from duplication of files, this is identified by strings of digits of certain, different, or changing lengths comparing the hash value of each file. A hashed file directory will use the hash function on the directory path. This will allow files to be searched quickly using a hash value compared to an array or a list. | |  |

|  |  |  |
| --- | --- | --- |
|  |  | |
| References (n.d.). In Tanenbaum, *Modern Operating Systems.*  Doeppner, T. W. (2010). Operating Systems in Depth. 127.  *Hashed Page Table In Operating Systems*. (2020). Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/hashed-page-tables-in-operating-system/  Operating System : Design and Implementation. (n.d.). In Tanenbaum. | |  |
|  | |  |
|  | |  |