**Analysis of Algorithms**

Spring 2020

**Members Details**

| Group ID | CSS311-G38 |
| --- | --- |
| Registration Number of Group Members | 2018-CS-53 , 2018-CS-68 |
| Section | B |

**Project Details**

|  |  |
| --- | --- |
| ***Project*** |  |
| Project Title | File Compression Tool |
| Executive Summary | File compression tool is used to reduce storage requirement such as hard disk space .The main objective of this project is to achieve faster file transfer .Data or file compression is an art of representing data in compact form .file compression is a coding process that used number of bits to encode information or data. The main principal of file compression is to transfer string of character into new string which contain same information but with a new length as small as possible  For compression of data we used “**HUFFMAN CODING”**. Huffman coding used for data compression. In this algorithm, a variable-length code is assigned to input different characters. The code length is related to how frequently characters are used. Most frequent characters have the smallest codes and longer codes for least frequent characters. Letter with high frequency assign smallest code and letter with low frequency assign longest code .Every letter is assigned a binary string either ‘0’ or ‘1’.Huffman coding is greedy algorithm that used greedy algorithm in its implementation. Greedy algorithm is more efficient than dynamic programming .Greedy algorithm is faster than dynamic programming .Greedy algorithm is subset of dynamic programming .dynamic programming is slow .in greedy algorithm we make a choice and never look at back and hope that our solution is optimal and it gives minimum cost .Main concept of greedy algorithm is that we divide problem into sub problems and for each step we take best choice and hope that globally optimal solution |
| ***Business Case*** |  |
| Outline the business need for the project | File compression allows you to store and back up significantly more data, faster. To effectively post files on a web page for someone else to download or to send large documents as email attachments. Files can become corrupted when they are transferred over the internet in an uncompressed format. |
| End user of the product | Students, Teachers can use this to compress file which is large in size and take more time to transfer. |
| Motivation for Project | I choose this project because of its high usage in almost every sector where data transfer through files. |
| Description of the project objective(s) | The main objective of project is to design a file compression tool which reduces disk space use by file without losing data. |
| State the level of impact expected should the project proceed and implications of not proceeding | This project can convert only compress those files that had txt extension. It could not convert any other file e.g. (pdf, zip, docx) to compress file. |
| Functional Requirements | The following functionality is implemented.   * **PrintFrequency**   This function print frequency or probability of character. Every character is given frequency. .Most frequent letter has smallest code and longer code for least frequent   * **BuildTree**   This function build Huffman tree by adding least frequency of two node in this way two node combine and give one node and so no of nodes will reduce   * **SetHuffcode**   When this function is called every character is given a specific code .letter with high frequency has smallest code and letter with small frequency has longer code   * **Ascii**    The ASCII function converts a string in EBCDIC code into ASCII code. An expression evaluating to the string to be converted. The ASCII function converts each character of the given expression from its EBCDIC representation value to its ASCII representation value.   * **FileWrite**   A file can be open in read or write mode we can write to a file after creating its name, by using the function FileWrite   * **PrintCode**   This function converts code begins edited into an html file display it by browser and print it   * **HuffTree**   This function made huff tree and give minimum cost by adding nodes who has least frequency   * **compressFile**   This function compress file with same information but with small length as possible as   * **Extension**   This function give extension of compressed file |
| ***Benefits*** |  |
| What benefits are expected/ anticipated? | The **main advantages of compression** are reductions in storage hardware, data transmission time, and communication bandwidth. This can result in significant cost savings. **Compressed files** require significantly less storage capacity than uncompressed **files**, meaning a significant decrease in expenses for storage.  AS compressed files are small in size so their speed of transfer also increased which is very beneficial. |
| ***Implementation Details*** |  |
| Link to Github Repository | <https://github.com/sana8aslam/CS311S20PID38> |
| Total Number of commits in repository before 30 July 2020 | 75 commits |
| Exact contribution of each member | 31 by 2018-cs-53 and 30 by 2018-cs -68 |
| ***Commits in github repository by each member*** | |
| |  |  | | --- | --- | | **Member Registration No.** | **Total Commits** | | 2018-cs-53 | 38 | | 2018-cs-68 | 32 | |  |  | | |
| **Details of commits** | |
| |  |  |  |  | | --- | --- | --- | --- | | **Sr. No.** | **Details of commit** | **Date** | **Member Reg No.** | | 1 | Create psuedocode.txt file which contain initial pseudo code like description or example type | June 30, 2020 | 2018-CS-53 | | 2 | Write pseudo code following proper protocol, Analyze problem to write pseudo code. | June 30, 2020 | 2018-CS-53 | | 3 | Make changes in pseudo code take example and update pseudo code in direction of this example. | June 30, 2020 | 2018-CS-53 | | 4 | Write description of project in Readme file. | June 30, 2020 | 2018-CS-53 | | 5 | Update pseudocode.txt file and try to write Huffman pseudo code with correction. | Jul 1,2020 | 2018-CS-53 | | 6 | Delete pseudocode.txt file because I submit it on eduko. | Jul 1,2020 | 2018-CS-53 | | 7 | I hufmann Huffman coding docs contain contain basic algorithm to build huff tree. | Jul 4, 2020 | 2018-CS-53 | | 8 | I delete previous docs and upload new Huffman coding docs (1) in which update algorithm. I use priority key for adding and deleting nodes. | Jul 4, 2020 | 2018-CS-53 | | 9 | I again update algorithm and this time I try to write it by following all rules of algorithm .First write problem and then analyze problem with example and write pseudo code and then write algorithm from the pseudo code. | Jul 4, 2020 | 2018-CS-53 | | 10 | I update algorithm by including decoding example and describe process of decoding step by step. | Jul 4, 2020 | 2018-CS-53 | | 11 | The new update made in algorithm is od building decoding algorithm from analysis of decoding tree by giving example. | Jul 4, 2020 | 2018-CS-53 | | 12 | Final Huffman decoding docx (1) uploaded with all changes above and deleted previous one. | Jul 4, 2020 | 2018-CS-53 | | 13 | Upload 3rd milestone named docs which contain time complexity analysis of pseudo code. | Jul 8, 2020 | 2018-CS-53 | | 14 | I delete 3rd mile stone docs and upload new which also contain updation in time complexity of algorithm. | Jul 8, 2020 | 2018-CS-53 | | 15 | I again updated complexity of algorithm and this time I calculate running time line by line of each step and upload changes. | Jul 8, 2020 | 2018-CS-53 | | 16 | I made some changes in correctness of algorithm that construct by my group fellow and upadated file upload. | Jul 8, 2020 | 2018-CS-53 | | 17 | I again updated correctness docx and add conclusion and upload the updated file. | Jul 16, 2020 | 2018-CS-53 | | 18 | I upload implementation file which is HeapNode.cs contain node class which contain variables and constructor and function of comparison of frequency of node. | Jul 18, 2020 | 2018-CS-53 | | 19 | I upload tree.cs file which contain tree file in which implement function following function:   * PrintFrequency * SetHuffcode * BuildTree * PrintCode * Ascii * HuffTree * FileWrite * compressFile | Jul 18, 2020 | 2018-CS-53 | | 20 | I upload file of program.cs in which we call the function by creating object of tree and then call functions define in tree in main file by object of tree. | Jul 18, 2020 | 2018-CS-53 | | 21 | I made changes in HeapNode.cs file and update the constructor of Node class by passing one parameter instead of two and update code of constructor. | Jul 19, 2020 | 2018-CS-53 | | 22 | I made changes in tree.cs file and update function of hufftree by passing nodelist we traverse the tree through each node and get characters again. | Jul 19, 2020 | 2018-CS-53 | | 23 | I again update tree.cs file and add extension function which change the extension of compress file from txt to cmp. | Jul 19, 2020 | 2018-CS-53 | | 24 | Now I update program.cs file and add extension function call. We change extension of compress file by calling extension function define in tree class. | Jul 19, 2020 | 2018-CS-53 | | 25 | Create implementation file to pace all code in one folder and merge the whole code in one folder. | Jul 20, 2020 | 2018-CS-53 | | 26 | Upload the folder which contain whole implementation of compressing a file named as HiffmanTest which contain tree.cs file, HeapNode.cs ,Program.cs file ,compress.cmp file and scheme file.txt. | Jul 20, 2020 | 2018-CS-53 | | 27 | Changes makes in scheme file which show now Asscii code, frequency, and huffcode generated by BuildHuff function. | Jul 25, 2020 | 2018-CS-53 | | 28 | Changes made in tree.cs file. The function of Compress File updated and saved changes in tree file. | Jul 25, 2020 | 2018-CS-53 | | 29 | Again tree.cs file updated by making changes in extension function.which change extension of compress file.The issue of path is resolved in this change. | Jul 25, 2020 | 2018-CS-53 | | 30 | Initial design of interfaces uploaded that just contain the design of interface, no functionality added now. | Jul 25, 2020 | 2018-CS-53 | | 31 | Readme file is updated guidelines to project is added. | Aug 4, 2020 | 2018-CS-53 | | 32 | AoaProject folder contain front end backend and configuration document uploaded. | Aug 5, 2020 | 2018-CS-53 | | 34 | Front end is again updated. In front end open button functionality is changed from folder to file. | Aug 6,2020 | 2018-CS-53 | | 35 | Folder with updated in backend front end by name Final is uploaded and committed. | Aug 8.2020 | 2018-CS-53 | | 36 | After Resolving issue1Final project folder is uploaded which contain updated configuration document. | Aug 8,2020 | 2018-CS-53 | | 37 | Updated FinalProject after resolving issue 2 of interface improvement is uploaded. | Aug 13,2020 | 2018-CS-53 | | 38 | Read me file updated by adding guidelines to readme file. | Aug 13, 2020 | 2018-CS-53 | | 39 | AOA third milestone of project uploaded that contain pseudo code and correctness of Huffman algorithm | Jul 9,2020 | 2018-CS-68 | | 40 | I again uploaded 3td milestone of project (1) .docx that contain new observation of correctness of algorithm | Jul 9,2020 | 2018-CS-68 | | 41 | 3rd milestone final correct .docx is uploaded it contain correctness, pseudo code and conclusion of all observation | Jul 10,2020 | 2018-CS-68 | | 42 | I uploaded AOA interface that contain all graphical user interface of project | Jul 17,2020 | 2018-CS-68 | | 43 | I again uploaded AOA interface (1) .zip that contain updated frames of project | Jul 17,2020 | 2018-CS-68 | | 44 | File compression .zip is uploaded that contain panel frames of all interfaces with different design | Jul 17 ,2020 | 2018-CS-68 | | 45 | GUI is again updated a file of name file compression (1).zip that consist of frames until encoded of file | Jul 17,2020 | 2018-CS-68 | | 46 | Now File compression updated.zip is uploaded that consist of frames of Selection, Compression Decompression, encoded, decoded, merge of file and design of frames is according to the project | Jul 17,2020 | 2018-CS-68 | | 47 | Decompree.zip file is uploaded that contain interface of decompression of file with new design (by changing its color and coding) | Jul 17,2020 | 2018-CS-68 | | 48 | AOA GUI is uploaded that contain all frames of project | Jul 18,2020 | 2018-CS-68 | | 49 | Decompress and compress GUI.zip that consist of new frames of compression or decompression of file | Jul 18,2020 | 2018-CS-68 | | 50 | File compression updated.zip is uploaded that contain file compression interface using button next or back for moving to next frames. | Jul 18,2020 | 2018-CS-68 | | 51 | New compression interface is uploaded that contain only wireframe of compression of file | Jul 18,2020 | 2018-CS-68 | | 52 | AOA project.zip is uploaded that contain all working of wireframes of interfaces from selection to merge of file | Jul 19,2020 | 2018-CS-68 | | 53 | All interface.zip uploaded that contain all wireframes of project that worked by selection of file using list view | Jul 19,2020 | 2018-CS-68 | | 54 | All interfaces-(1) is uploaded which contain frames according to file compression project (same design and color) | Jul 19,2020 | 2018-CS-68 | | 55 | Final GUI FC .zip which contain final frames of project | Jul 21,2020 | 2018-CS-68 | | 56 | Interfaces .zip file is uploaded | Jul 21,2020 | 2018-CS-68 | | 57 | Milestone-(1) is uploaded because previous file is not working right (some issues of visual studio) | Jul 21,2020 | 2018-CS-68 | | 58 | Milestone updated is uploaded which contain all new updated frames from selection of file to decoded of file. merge of file frame is delete because its coding not working | Jul 21,2020 | 2018-CS-68 | | 59 | Window frames .zip is uploaded which consist of frame from selection of file to decode of file with compression button coding | Jul 21,2020 | 2018-CS-68 | | 60 | Graphical user interface is uploaded which consist of frames and two buttons previous and back for shifting from one frame to other | Jul 26,2020 | 2018-CS-68 | | 61 | Milestone.zip is uploaded that contain class of node and Huffman tree coding | Jul 26,2020 | 2018-CS-68 | | 62 | Project.zip is upload that contain class of node and tree and compression button coding | Jul 26,2020 | 2018-CS-68 | | 63 | Work.zip is uploaded that contain final work of file compression project with all interfaces and coding of tree and node class | Jul 26,2020 | 2018-CS-68 | | 64 | PROJECT REPORT . docx ., This contain introduction . End user input or output format pseudocode , interface of project | July 29,2020 | 2018-CS-68 | | 65 | AoA report Template .. this file contains all detail about project . Pseudo code and it's correctness and all commits on GitHub | Aug 6, 2020 | 2018-CS-68 | | 66 | I again uploaded updated CS configuration documents which contain step by step project. Working and installation | Aug 13,2020 | 2018-CS-68 | | |
| Have you used built in algorithms or you have implemented yourself? | I take lectures of Huffman coding from you tube and understand its working and then write pseudo code by myself. |
| Formats of input | Input format of file compression is that which file is selected for compression of data .in graphical user interface we use list view to show all folder in required directory .and there we show path in text box and button of selection where we can select ‘txt’ file .when selection button is clicked, there comes a list of folder from where selection of file is done .at the time when selection of file is done in text box there show a path of selected file  *.* |
| Validations | The input file should be a file with txt extension. |
| Format of output | A compress file with cmp extension generated. A scheme file with txt extension also generated. |
| Deployment | No I did not deploy my project. I put it just on Github. |
| ***Details of algorithms*** | |
| **Algorithm Huffman(X)**  **Input:** String x of length n with d distinct characters  **Output:** coding tree for X   1. Compute the frequency f (c) of each character c of X. 2. (c1, f[c1] ) , (c2, f[c2] ) ,…, (cn, f[cn] ) 3. Initialize a priority queue Q 4. For i=1 to n-1 Do 5. Create a single-node binary tree storing c 6. Insert T into Q with key f (c). 7. While Size ( )> 1 do 8. *F1* Q.min() 9. *T1* Q.removeMin() 10. *F2* Q.min() 11. *T2* Q.removeMin() 12. Create a new binary tree T with left sub tree T1 and right sub tree T2 13. T.left=T1 T.right=T2 Insert T into Q with key f1 +f2 14. T.f = T1.f1 + T2.f2 15. return Tree Q.removeMin()   **Time Complexity**  Huffman’s algorithm proceeds as shown in above. Since the alphabet contains 6 letters, the initial queue size is n, and the final tree represents the optimal prefix code. The code word for a letter is the sequence of edge labels on the simple path from the root to the letter.   * Line 1-2 compute the frequencies of all characters from c1 to cn in input string X. * Line 3-6 initializes the min-priority queue Q with the characters in X. **For** loop is used to store all character’s frequency in priority queue Q. * Lines 7–14 repeatedly extracts the two nodes T1 and T2 of lowest frequency from the queue using while loop, replacing them in the queue with a new node T, representing their merger. The frequency of T’ is computed as the sum of the frequencies of T1 and T2 in line 15. * The node T’ has T1 as its left child and T2 as its right child. (This order is arbitrary; switching the left and right child of any node yields a different code of the same cost.) * After (n −1) mergers, line 15 returns the one node left in the queue, which is the root of the code tree.   .  To analyze the running time of Huffman’s algorithm, we assume that Q is implemented as a binary min-heap. For a set X of n characters, we can initialize Q using **For loop** in line 3-6 in O (n) time using the **BUILD-MIN-HEAP** procedure. The **While loop** in lines 7-14 executes exactly when only one last node left in queue (**size=1)** which takes n-1 iterations, and since each heap operation requires time O(lgn), the loop contributes O(nlgn) to the running time. Thus, the total running time of HUFFMAN on a set of n characters is **O(nlgn).**  **Correctness of Algorithm:**  The question is that greedy algorithm is correct or incorrect , that is does it compute the tree that minimizes the expected encoding length and optimal substructure properties the cost of any encoding tree T is **B(T) = ∑ x p(x) dT (x). where p(x) is the probability and d(x) is length.** Our approach will be to show that any tree that differs from the one constructed by Huffman’s algorithm can be converted into one that is equal to Huffman’s tree without increasing its cost.. Our approach based on the following observations.   * Huffman tree is full binary tree * in any optimal code tree * Huffman’s algorithm produces an optimal prefix code tree   **Basis:**  Suppose there are only two characters in C, the algorithm encodes each of them by one bit and this is optimal.  **Inductive Step:**  **Observation 1:**  Consider the two characters, x and y with the smallest probabilities. Then there is an optimal code tree in which these two characters are siblings at the maximum depth. Let T be any optimal prefix code tree, and let b and c be two siblings at the maximum depth of the tree. (There may be many such siblings, and if so pick any such pair.) If {x, y} = {b, c} we are done. Otherwise, from the fact that x and y have the lowest probabilities, we may label the nodes such that p(b) ≤ p(c) and p(x) ≤ p(y). Now, since x and y have the two smallest probabilities it follows that p(x) ≤ p(b) and p(y) ≤ p(c). (In both cases they may be equal.) Because b and c are at the deepest level of the tree we know that dT (b) ≥ dT (x) and dT (c) ≥ dT (y). (Again, they may be equal.) Thus, we have p(b) − (x) ≥ 0 and dT (b)−dT (x) ≥ 0, and hence their product is nonnegative. Now, suppose that we switch the positions of x and b in the tree, resulting in a new tree T. Next let us see how the cost changes as we go from T to T’ . Almost all the nodes contribute the same to the expected cost in both trees. The only exceptions are nodes x and b. By subtracting the old contributions of these nodes and adding in the new contributions we have  B(T ´ ) = B(T) − (old cost for b and x) + (new cost for b and x)  = B(T) − (p(x)dT (x) + p(b)dT (b)) + (p(x)dT (b) + p(b)dT (x)).  With a little algebraic manipulation we obtain  B(T ´ ) = B(T) + p(x)(dT (b) − dT (x)) − p(b)(dT (b) − dT (x))  = B(T) − (p(b) − p(x))(dT (b) − dT (x))  ≤ B(T),  where the last step follows because (p(b) − p(x))(dT (b) − dT (x)) ≥ 0. Thus the cost does not increase. (Given our assumption that T was already optimal, it certainly cannot decrease either, since otherwise we would have a contradiction.) Since T was an optimal tree, T’ is also an optimal tree. By a similar argument, we can switch y with c to obtain a new tree T’’. Again, the same sort of argument implies that T’’ is also optimal. The final tree T’ satisfies the statement of the claim. The above claim applies to just one pair of nodes, those with the lowest probabilities. To show that the entire Huffman tree is optimal, we need to extend this argument do this by induction. In order to reduce from n characters to n − 1, we will do the same reduction that Huffman’s algorithm does; namely we will merge characters x and y into a new meta-character z, whose probability is the sum of the probabilities of x and y.  **Observation 2:**  Let Tn be any prefix-code tree that satisfies the property of Claim 1 (lowest probability symbols x and y are siblings at the deepest level). Let Tn−1 be the tree that results by replacing these two nodes and their parent with a single leaf node z of probability  p(z) = p(x) + p(y). Then B(Tn) = B(Tn−1) + p(z).  Let d denote the depths of x and y in Tn. Clearly, z is at depth d − 1 in Tn−1  Because z replaces x and y the costs of the two trees satisfies  B(Tn) = B(Tn−1) − (z’s cost in B(Tn−1)) + (x and y’s costs in B(Tn))  = B(Tn−1) − p(z)(d − 1) + (p(x)d + p(y)d)  = B(Tn−1) − p(z)(d − 1) + p(z)d  = B(Tn−1) + p(z).  Note that the cost of trees Tn and Tn−1 differ only by the fixed term p(z), which does not depend on the tree’s structure. Therefore (subject to this replacement), minimizing the cost of Tn is equivalent to minimizing the cost of Tn−1. This allows us to prove our main result.  **Observation 3:**  Huffman’s algorithm produces an optimal prefix code tree.  The proof is by induction on n, the number of characters. The basis case (n = 1) is trivial, since there is only one tree possible. If n ≥ 2, then by Claim 1, we know that the two characters x and y of lowest probability are siblings at the deepest level of an optimal tree. Huffman’s algorithm replaces these nodes by a character z whose probability is the sum of their probabilities. By induction, Huffman’s algorithm computes the optimum tree over the resulting alphabet of n − 1 symbols. Call it Tn−1. Replacing z with nodes x and y results in a tree Tn whose cost is higher by the fixed amount p(z) = p(x) + p(y). Since Tn−1 is optimal, and the cost of replacement does not depend on the tree’s structure, Tn is also optimal.  **Conclusion:**  Thus by mathematical induction, function Huffman creates the Huffman tree with minimum external path length. | |
| ***Interfaces for your project*** | |
| **Selection of interface:**  **Description:**  In this interface we select file which need to be compressed by clicking on button open there will open all folder then select txt file .in text box there show path of selected file.    **Compression of file:**  **Description:**  In this interface we compress the selected file by clicking on compress button .in compress button many functionalities has been done like printfrequency, BuildTree, setHuffcode, ASCII, FileWrite, PrintCode, HuffTree, CompressFile or Extension.    **Decompression of file:**  **Description:**  In this interface after selection of file, compression of file is done now compressed file is decompress by clicking on decompress button    **Encode of file:**  **Description:**  In this interface file is encode by clicking on encode button encoding is a process of converting data from one form to other form .by encoding the file it can be save into more efficient compressed format.    **Decode of file:**  **Description:**  In this interface file is decode by clicking on decode button. Decoded is opposite process it is conversion of encoded file into original file or sequence of character | |
| ***Integration*** | |
| As code was implemented by one member and interfaces made by other member.When I have to integrate both my laptop was not working and I try to told my group mate on whatsapp integration of code but it could not go well because actually understand errors on hers laptop. | |
| ***Change Requests*** | |
| Yes, I do some changes in algorithm while implementing. First in my algorithm I am using queue to add or remove nodes but when I face difficulty in implementing queue I go to option of list/array which is working for me. I do this change with the permission of sir samyan. The time complexity of building min heap is O(n) .Due to this change we will not use min heap instead use list which also take O(n) time to maintain ,So no effect on complexity of function. | |
| ***Testing*** | |
| Issues:Issue 1: User Interface Improvements #1  * There is no color scheme defined. * There should be a proper font style defined with a consistent font size use on every page. * The open and compress button has a typo. * Buttons should have a hover pointer over them. * Buttons should change colors when being hovered over them. * Button's text should be properly aligned. * The overall application text alignment is not properly done.  Issue 2: Configuration Document #2 The configuration document that is given in the AoaProject folder does not include the steps to start the application. Issue 3: Exception Handling #3 **Description**  The project is throwing an exception.  **To Reproduce** Steps to reproduce the behavior:   1. Extract AOA PROJECT FC.zip. 2. Go to milestone folder. 3. Open milestone.sln file. 4. Click on **Start** on Visual Studio to start the project. 5. Click on Open. 6. Click in Next. 7. Click on Compress.   **Expected behavior** The project should compress a file without any error whatsoever.  **Solved Issues:**  **Solution of issue1:** it is solved by updating interface all the points mentioned by Saad have been considered and folder with updated interface is uploaded by name FINAL PROJECT.  **Solution of issue2:** Iupdated configuration document and explain all guidelines comprehensively**.** Thefolder with updated configuration document is uploaded by name FINAL PROJECT.  **Solution of Issue 3:** The system is throwing exception because integration of code with interfaces could not succeeded .Simple code without interface integration is running but fails when I integrate it with interfaces. I try my best to resolve this issue but it could not resolve from me. | |
| ***Technology*** |  |
| Programming Language | C# and c#.net. |
| Platform | Desktop Application |