Decoding Compressed Image TIFF files of Candidates Response Sheets in Objective Type Entrance Tests

*Rajdeep Kaur, **Parneet Kaur, ***Navneet Singh
*Research Scholar, Department of Information Technology,
Adesh Institutes of Engg. & Technology, Faridkot
**Assistant Professor, Department of Information Technology,
Adesh Institutes of Engg. & Technology, Faridkot
***Associate Professor, Department of Information Technology, Adesh Institutes of Engg. & Technology, Faridkot

Abstract

Evolution of Information Technology has brought about many changes in our daily life. As the applications of Information Technology vary from the simplest one like word processing to the highly sophisticated systems like satellite launching systems. This paper relates to a combination of Optical Mark Recognisition (OMR) & Data/Image Compression & Decompression techniques. The paper is to be used for decoding & evaluating candidates' Responses in objective type entrance tests using OMR. The basic problem is of the storage space required for storing the image files of the candidates' Responses for decoding & result evaluation. These image files, if are uncompressed, take a lot of storage space. Using compressed image files the required storage space can be reduced but it is at the cost of slight increase in decoding time. This approach saves about 75 percent of storage keeping in view the number of the candidates appearing in various entrance tests, use of the proposed method results in considerable savings in terms of storage space, required for storage of images TIFF has been used.

1 INTRODUCTION

Recent advances in communication and computer technologies have made possible the exchange and retrievals of information through, the electronics media. To store and transmit digital data efficiently, variety of data compression algorithms has been developed. Yet, information in compressed data cannot be easily retrieved or modified without decompression. While the research body on compression schemes is quite rich, algorithms for retrieving and modifying information in compressed data have not been widely integrated. The objective of this research is to identify operations that can be applied directly and efficiently to digital information encoded by a given compression algorithm. A formal method of analysis is planned. This paper mainly focuses on lossless compression techniques and images processing and decoding algorithms. This paper mainly relies on two technologies viz. Optical Mark Recognition (OMR) and Image Compression/Decompression.

Optical Mark Recognition-OMR refers to the technique of converting a handwritten mark into an ASCII value. Filling a circle or a box or a special form, which is designed as per the given specifications, makes a mark. The presence or absence of a mark in a specific location

is then converted into a value such as a selection in multi-choice question, the selection of one item in a list of several or even to code a specific numeric or alphanumeric value.

Methods of Data Compression/Decompression

Statistical Method:- This method is based on the probability of occurrence of the symbols in the input stream. In this method we encode a single symbol at a time depending on the number of times that particular symbol occurs in the input streams. These methods are implemented statically or dynamically. In static method we first calculate the number of times a particular symbol occurs in the input streams and then we find its probability of occurrence depending on which we encode the symbol. In dynamically method, we do not find the probability of the symbol in the whole of the input streams but keep on finding the probability and encoding the symbol, as we keep moving through the input stream. Although at the end of the reading of the input stream the total probability of each symbol found equals that found for the static method in the beginning.

Dictionary Method:- This method is not concerned with the probability of occurrence of particulars symbols and then encoding them. In this method, we encode string of symbols with the single code .In this method we read the data from the input-stream and look for the group of symbols in the dictionary. If a match is found then we pass an index or pointer to that word in the dictionary, if no match is found we add the word to the dictionary.

Methodology used:-

Lempel - Ziv - Welch (LZW) Algorithm: LZW is a way of compressing the data that takes the advantage of repetition of the strings in the data. LZW compression replaces strings of characters with single code. It just adds every new string of characters it sees to a table of strings. Compression occurs when a single code is output instead of a string of characters. The code that the LZW algorithm outputs can be of any length, but it must have more bits in it than a single character.LZW manipulates three objects in both compression & decompression namely char stream, the code stream, and string table. In compression, the char stream is the input & the output is the code stream in decompression the code stream is the input & the char stream is the output. The string table is the product of the both the compression & the decompression.

Run-Length Encoding (RLE) algorithm: - The idea in this scheme is to recode the data with regard to the repetition frames. A frame is one or more bytes that occur one or several times. Each of the coded values have the following describing the following three or four points.

1. Value specifying if there is repetition or not.
2. Value telling now many repetitions / non repetitions.
3. Value of the length of the frame.
4. Value of the frame to repeat (or not) following are the varying algorithms for run length encoding compression scheme:

CCITT Compression:- The CCITT T.4 and T.6 specifications outline many differ kinds of requirement for the facsimile apparatus, including the resolution of scanning and printing, dimensional tolerances, timing constraints etc. The sections which tend to be used in the imaging world are those which describe the compression schemes and those are often referred to as group 3 and group 4 in their own right. The compression and decompression algorithms are quite complex as will be seen and must be implemented either in hardware or supplicated software in order to achieve good performance.

Huffman Algorithms – in this method, we calculate the probability of occurrence of each symbol in the whole of the data to be compressed.The algorithms are as follows for Huffman algorithms:-

- Calculate the probability of occurrence of each symbol in the input file.
- Arrange the symbols in the order of their decreasing probabilities, i.e. the symbol with the highest prob. the bottom.
- Pick the two nodes with the highest frequency probability of occurrence n join then together with the help of the parent node created above it. The wt. of the parent node is equal to the total of the weights of the 2 child nodes. This node is given any name/index no. the node on the left of the parent node is assigned the 0 bit and has weight equal to or less than the child node on the right of the parent node, which is assigned the bit 1.
- Now rearrange the symbols again, with the symbols of the child nodes that have been joined together replaced by the name of the parent node n its weight in the using order of their frequencies.

- If we r not left with just a single root called the root node then repeat the process step 3 again.
-

Results & discussions

From figures, we see that the times taken to process/decode the compressed files are more than the time for processing of uncompressed file. This time difference is not in processing files but is in decompression of compressed image data.

Comparison of Decoding Time:-

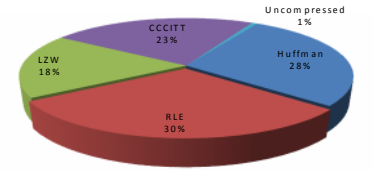

Figure 1: shows the decoding time of 50 questions

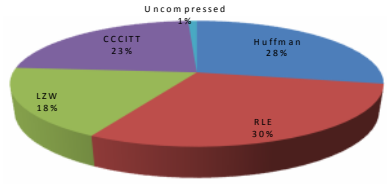

Fig 2: shows the decoding time of 100 questions

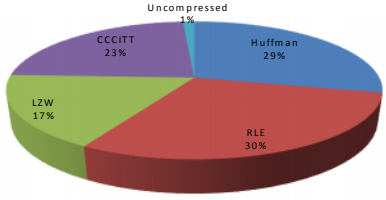

Fig 3: shows the decoding time of 150 questions

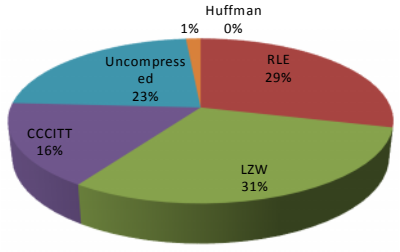

Fig 4: shows the decoding time of 200 questions

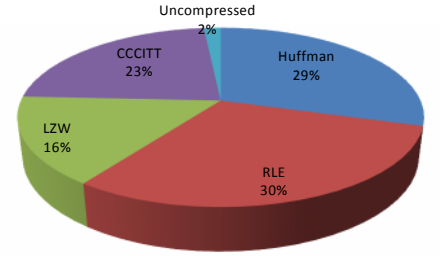

Fig 5: shows the decoding time of 250 questions

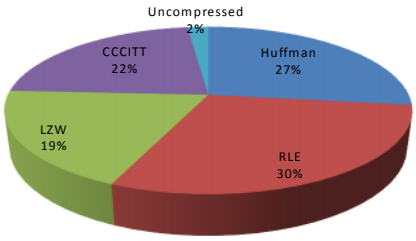

Fig 6: shows the decoding time of 300 questions

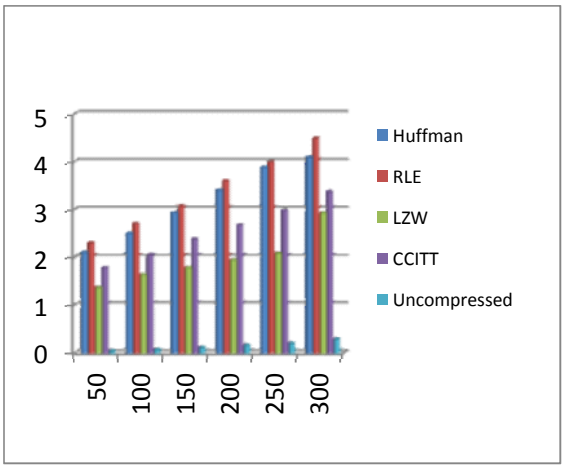

Figure 7: shows the comparison of decoding time of various algorithms

Comparison of Storage Space

Below we have discussed the storage space of various algorithms like Huffman's algorithms, RLE, LZW, CCITT, along with uncompressed data. Each algorithm have to differentiate with different questions having 50,100,150,200,,250,250,300 questions. Each chart will explained their difference as shown in figures:-

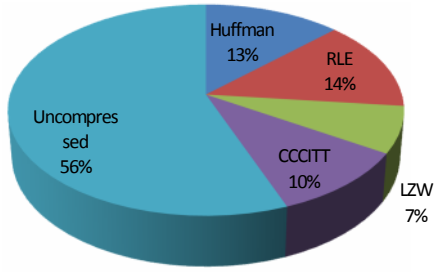

Fig 8: shows the storage space having 50 questions

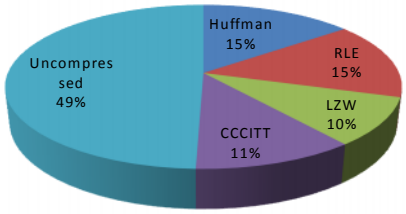

Fig 9: shows the storage space having 100 questions

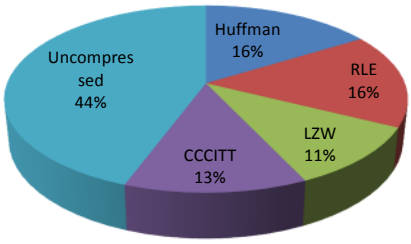

Fig 10: shows the storage space having 150 questions

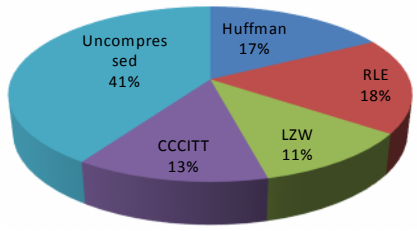

Fig 11: shows the storage space having 200 questions

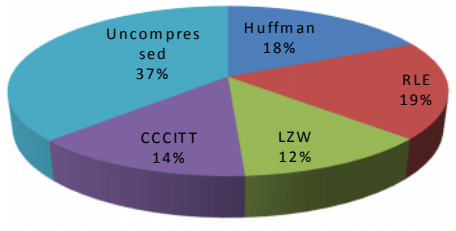

Fig 12: shows the storage space having 250 questions

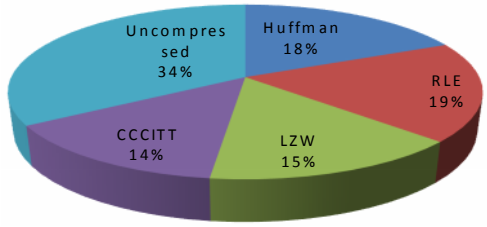

Fig 13: shows the storage space having 300 questions

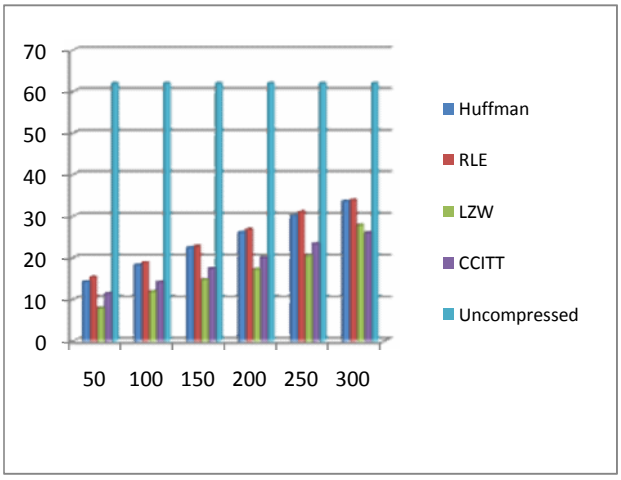

Fig 14: shows comparison of storage space of various algorithms.

Conclusion & further Scope

From the results obtained above we find that although the speed of the process is slow but still we are able to have a lot saving in terms of storage space. Usually a lot of space is taken up when we are evaluate examination papers for any entrance test, as the number of candidates appearing in examination papers is quite large. Hence, at places where the storage space is a factor of consideration we can use technique described here for processing.We had only considered LZW decompression here as the majority of scanning software's for TIFF files use LZW compression techniques. The others methods of compressions can also be implemented for the compression of TIFF files. One of the ways for improving the processing speed of the method describe here would be to use the binary search tree for the storage of the strings as speed for searching in binary tree is high as compared to searching is an array. Another way for improving the processing speed is to scan a document and then instead of storing the scanned image in a file we can directly send this scanned image dates for decoding. This will help in further reduction in storage space, as we would not need to store any file.

References:-

[1] K.saywood, Introduction to Data Compression, Morgan Kaufamann publishers, Inc., 1996.
[2] International standard ISO/IEC 11544:1993 and ITU-F Recommendation T.82 (1993) information Technology-coded representation of picture and audio information-progressive Bi-Level image compression.
[3] On the average redundancy rate of the Lempel-Ziv code (with G. Louchard, IEEE information technology theory, 2-8, 43, 1997.
[4] Average profile for the generalize digital search tree and the generalized Lempel-Ziv Algorithms, (with G.Louchard andJ.Tang),SIAM.J.Computing,To appear.
[5] Terry Welch," A Technique for Higher-performance Data Compression", computer, June 1984.
[6] J.Ziv and A. Lempel," A Universal Algorithms for Sequential Data Compression" IEEE Transactions on Information Theory, May 1977.
[7] Aldus TIFF 6.0 specifications.
[8] Nelson, Mark, The Data Compression Book, 2nd ed. New Delhi, India: BPB publication 1996.
[9] David Bourgin, http:\\www.frima.Img.fr
[10] Mark Nelson," LZW Data compression ", Dr. Dobb's journal, 1989.
[11]Terry Welch," A Technique for high-performance data compression", computer, June 1984.
[12]J.Ziv and A. Lempel," A Universal Algorithms for Sequential Data Compression", http://www.ieee.org IEEE Transactions or Information Theory, May 1977.
[13] "Optical Mark Recognition", as gives in website: http://www.aimglobal.org/technology/
[14]"Introduction to Mark Recognition", as given in website: http://www.imageprocessingtools.com/
[15] "What is OMR", as given in website: http://www.omrform.com/

[16] Image processing in C- analyzing and enhancing digital images, Dwayne Philips, BPB publication.
[17]P.J. Burt, E.H. Adelson, The Laplacian Pyramid as a Compact Image Code, IEEE Trans. on Communications, pp. 532–540, April 1983.
[18]D. Shao, L.A. Mateos, W.G. Kropatsch, Irregular Laplacian Graph Pyramid, CVWW 2010.
[19]Lynch, Thomas D., Data Compression Techniques and Applications, Lifetime Learning Publications, Belmont, CA, 1985.
[20]Nelson, Mark R., The Data Compression Book, M&T Books, Redwood City, CA, 1991.
[21]Storer James A., Data Compression: Methods and Theory, Computer Science Press, Rockville, MD, 1988.
[22]Garey, M. R. 1974. Optimal Binary Search Trees with Restricted Maximal Depth. SIAM J. Comput. 3, 2 (June), 101-110.
[23]Gilbert, E. N. 1971. Codes Based on Inaccurate Source Probabilities. IEEE Trans. Inform. Theory 17, 3, (May), 304-314.
[24]Huffman, D. A. 1952. A Method for the Construction of Minimum-Redundancy Codes. Proc. IRE vol. 40, 9 (Sept.), 1098-1101.
[25]Jacob Ziv and Abraham Lempel; Compression of Individual Sequences Via Variable-Rate Coding, IEEE Transactions on Information Theory, September 1978.
[26]Ken Huffman. Profile: David A. Huffman, Scientific American, September 1991, pp. 54–58
[27]Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 16.3, pp. 385–392.
[28]Salomon, Data Compression, 2nd Edition. Springer, 2001.
[29]D.E. Knuth — Dynamic Huffman Coding — Journal of Algorithms, 6, 1983 pp. 163-180.
[30]Phillips, Dwayne, "LZW Data Compression," The Computer Applications Journal, Circuit Cellar Ink, vol. 27, June/July 1992, pp. 36-48.
[31]Ziv, J., and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding," IEEE Transactions on Information Theory, vol. 24, no. 5, September 1978.
[32]S. Khalid," Introduction to Data Compression", 2nd edition, San Mateo, CA: Morgan Kaufmann, 2000.
[33]Wei Cui, "New LZW Data Compression Algorithm and Its FPGA Implementation", School of Information Science and Technology, Beijing Institute of Technology, Beijing, 100081, China.
[34]J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, May 1977
[35]Rudy Rucker, "Mind Tools", Houghton Mifflin Company, 1987
[36]Held, Gilbert, Data Compression: Techniques and Applications, Hardware and Software Considerations, second edition, John Wiley & Sons, New York, NY, 1987.