Huffman Coding

Abdallah T.Reda
abdallahtamer.ali98@eng-st.cu.edu.eg

Abdallah M.Shehata
abdozza.206@gmail.com(for git) abdallah.mohamad98@eng-st.cu.edu.eg

Ezz eldin I.Ezzeldin
Ezzeldeen.esmail99@enf-st.cu.edu.eg

Youssef s.Azmi
youssefsamir831999@gmail.com (for git) youssef.saleb99@eng-st.cu.edu.eg

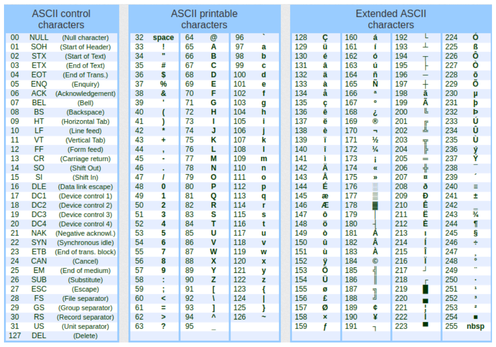
Pola N.kamal
pola.gadallah98@eng-st.cu.edu.eg

June 7, 2020

1 Introduction

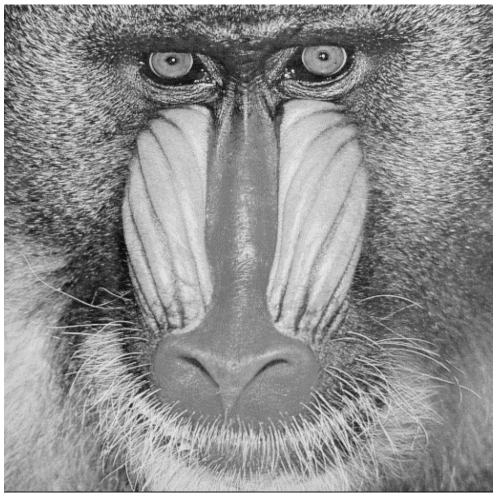
Before we get into our project we need get some points.
First of all what is the aim of Huffman. Huffman is a technique used for encoding [1].
It is used to compress the size of files, but how??

It uses the ascii extended table as the following figure [2]:



ASCII control characters			ASCII printable characters								Extended ASCII characters							
00	NULL	(Null character)	32	space	64	@	96	`	128	Ç	160	á	192	└	224	Ó
01	SOH	(Start of Header)	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
02	STX	(Start of Text)	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
03	ETX	(End of Text)	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
04	EOT	(End of Trans.)	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
05	ENQ	(Enquiry)	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
06	ACK	(Acknowledgement)	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
07	BEL	(Bell)	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
08	BS	(Backspace)	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
09	HT	(Horizontal Tab)	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	LF	(Line feed)	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	VT	(Vertical Tab)	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	FF	(Form feed)	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	CR	(Carriage return)	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	SO	(Shift Out)	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	SI	(Shift In)	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	DLE	(Data link escape)	48	0	80	P	112	p	144	É	176	░	208	ð	240	≡
17	DC1	(Device control 1)	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	DC2	(Device control 2)	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
19	DC3	(Device control 3)	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	DC4	(Device control 4)	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	NAK	(Negative acknowl.)	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	SYN	(Synchronous idle)	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	ETB	(End of trans. block)	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	CAN	(Cancel)	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	EM	(End of medium)	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	SUB	(Substitute)	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	ESC	(Escape)	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	FS	(File separator)	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
29	GS	(Group separator)	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	RS	(Record separator)	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	US	(Unit separator)	63	?	95	_			159	ƒ	191	┐	223	▀	255	nbsp
127	DEL	(Delete)														

So that as in our example which is a picture it uses pixels in form of ascii numbers and get the count for each ascii number.
It will then form a tree depending on these counters associated with the ascii given. This tree is formed using min heap which was given in sections or we can use priority queue as we used in our project. After we formed the tree whatever the algorithm we used, we will try to trace this tree or traverse it to get our binary encoded. The binary depends on the path of the root of the tree. Most of algorithms in google uses 1 for right and 0 for left. But as I remember that in lectures we used the opposite like(anatomical positions),but any way it won't change the output if we use a single numbering for the whole algorithm.
Lets now get closer to our aim. (the pgm pictures). Pgm refers to portable gray map. It has 2 types, p2 and p5.[3]
p2 is responsible for storing pixels in ascii, while p5 (which we took in our samples) used to store pixels in binary. Sothat p5 is compressed while p2 is not. The next picture is a pgm one[4].



2 Motivation

This project(Huffman encoding) makes us know about a new field which we think it can be an intro to image processing, making a project in this field will help the team members make a good decision if this field is good for them or not. We searched about the main algorithm of Huffman then we needed to search about types of pictures which made us need to search for types of pgm and how it differs. So we got a lot of knowledge about many fields such as coding, algorithms, and differentiate between types of image and how to deal with them using different methods.

3 Resources

As we understand from this title we are asked to write the libraries used. First and main library to use if fstream and sstream, we used them to get inputs or outputs between files and cpp. We used the command ios::app for example to just copy a text or data from cpp to a file without deleting them, we used also ios::binary to deal with a binary data file such as (p5 pgm). We used also string and string.h so that we can deal with string through store, print or either know length of string. Queue also used to make a priority queue that arranges the nodes due to some parameter (which in our case is counters) there are many other small new commands we used such as vectors , dynamic arrays with 2 dimensions and many others.

4 Challenges and Problems

In our project, we met many problems.
a)On of the main problems is main idea, at first, the algorithm was difficult to be converted into codes, how to make nodes that merge together forming new nodes and rearrange them and repeat this cycle till having only one node which is the root. But using the sections videos we get the first idea when we remembered the relation between huffman and min heap so that after period of time we reached another point which is priority queue so the first problem solved.
b)The second is the pictures and how to deal with it without using matlab for example, after we made a code to read the picture we found a big problem destroyed our code. Which is pgm has types and we are using another type. It took about 2 days to know our problem and after many times of trying we made a suitable code for it.
c)The last problem was data types. And we are sorry ta say that there is still problem and it until the moment we wrote this report the problem present. The program is working perfect but the problem is the txt file. The memory stored take a great amount as we store binary data in form of integers. It is the only problem that exist and we are trying hardly to manage.

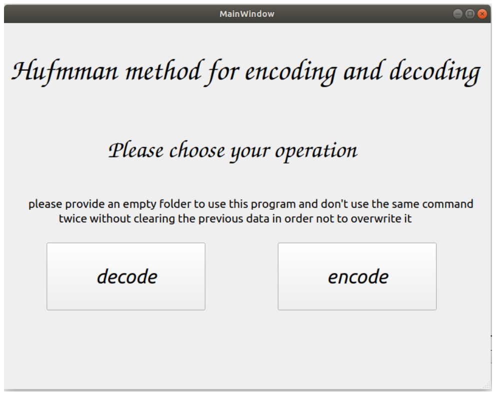
5 User manual for the system

our program is so easy to use just follow the next steps:
Frist thing that a main window will appear and you can choose the operation encode or decode, but please take care of several things:
a) make an empty folder that will contain all files that will be made during the running of the program.
b) don't make the same command twice as you will have a problem with overwrite the data, delete the old files first then you can use it again.
Now we can get to our 2 conditions which are encoding and decoding:
1) in case of encoding:
A) you will be asked to enter picture location in your operating system such as "/home/abdallah/Desktop/data/NORMAL2-IM-1430-0001.pgm" then you will enter the location for the empty folder that will contain the source, asci , encode files.
b) enter the names for the files that will be created.
c) 3 new files will be created in the location you wrote, which are:
i) source file that will store number of cols, rows and max value.
ii) asci file that will store the frequency table for the picture.
iii)output file that will include the encoded binary for the whole picture.
2) in case of decoding:
A) it is similar to encoding, first you will asked to insert the location for the folder containing the ascii, source and output files which are previously made in encoding and the decoded file for returning back the decimal numbers from the binary, and the new

image that will be formed with this decoded file which must be the same as the original file to make sure that it is decoded right.
B) press on decode button, wait few milliseconds till a message say that the decoded files have been created.
c) check your empty file , you will see all previous files were added with the image

6 Results

First of all we used qmake instead of cmake in the gui version while the terminal using version is with cmake.
First thing you will see when you open the program is that window

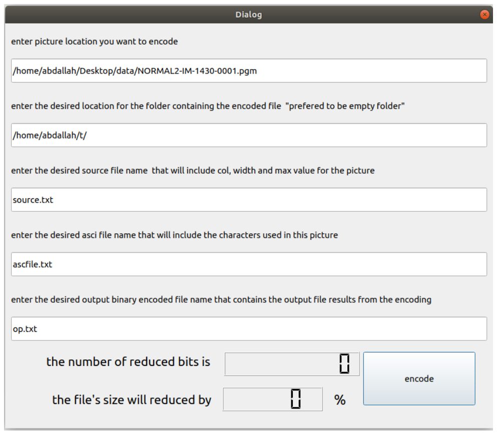

MainWindow

Hufmman method for encoding and decoding

Please choose your operation

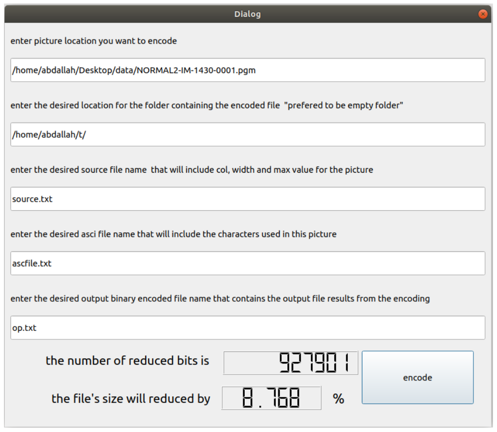
please provide an empty folder to use this program and don't use the same command twice without clearing the previous data in order not to overwrite it

Lets assume that you clicked the encode button, the next window will appear.
This window has simple inputs to get.
1) wherever your picture location it doesn't matter, only copy its location and paste it here.
2)make an empty folder then get its location and paste it as the example written in the text box.
3,4,5,6) just type names for each file and let the program create them for you.


Dialog

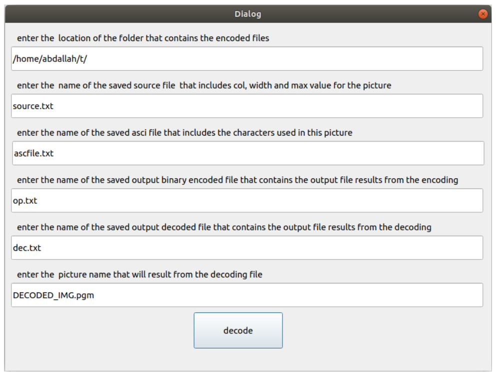
enter picture location you want to encode
/home/abdallah/Desktop/data/NORMAL2-IM-1430-0001.pgm
enter the desired location for the folder containing the encoded file "prefered to be empty folder"
/home/abdallah/t/
enter the desired source file name that will include col, width and max value for the picture
source.txt
enter the desired asci file name that will include the characters used in this picture
ascfile.txt
enter the desired output binary encoded file name that contains the output file results from the encoding
op.txt
the number of reduced bits is 0
the file's size will reduced by 0 %

After you finish typing, just click on encode button, the number of reduced bits and its percentage will appear on the lcds as follows.


Dialog

enter picture location you want to encode
/home/abdallah/Desktop/data/NORMAL2-IM-1430-0001.pgm
enter the desired location for the folder containing the encoded file "prefered to be empty folder"
/home/abdallah/t/
enter the desired source file name that will include col, width and max value for the picture
source.txt
enter the desired asci file name that will include the characters used in this picture
ascfile.txt
enter the desired output binary encoded file name that contains the output file results from the encoding
op.txt
the number of reduced bits is 927901
the file's size will reduced by 8.768 %

Now we have finished encoding, lets return back to our main page and click on decoding button, you will find similar inputs like for encoding.
1) we will enter the same folder we just made for encoding that will have now the encoded files.
2,3,4) enter the same names you have chosen for these file in encoding.
5,6) enter also names for decoded file and picture.


Dialog

enter the location of the folder that contains the encoded files

/home/abdallah/t/

enter the name of the saved source file that includes col, width and max value for the picture

source.txt

enter the name of the saved asci file that includes the characters used in this picture

ascfile.txt

enter the name of the saved output binary encoded file that contains the output file results from the encoding

op.txt

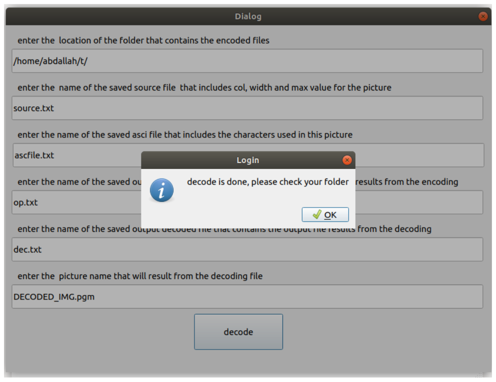
enter the name of the saved output decoded file that contains the output file results from the decoding

dec.txt

enter the picture name that will result from the decoding file

DECODED_IMG.pgm

Now just press on decoding button and immediately a message we appear that decoding is finished.


Dialog

enter the location of the folder that contains the encoded files

/home/abdallah/t/

enter the name of the saved source file that includes col, width and max value for the picture

source.txt

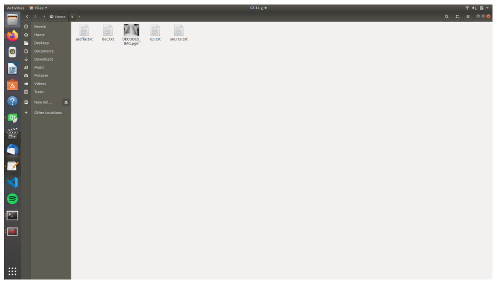
enter the name of the saved asci file that includes the characters used in this picture

ascfile.txt

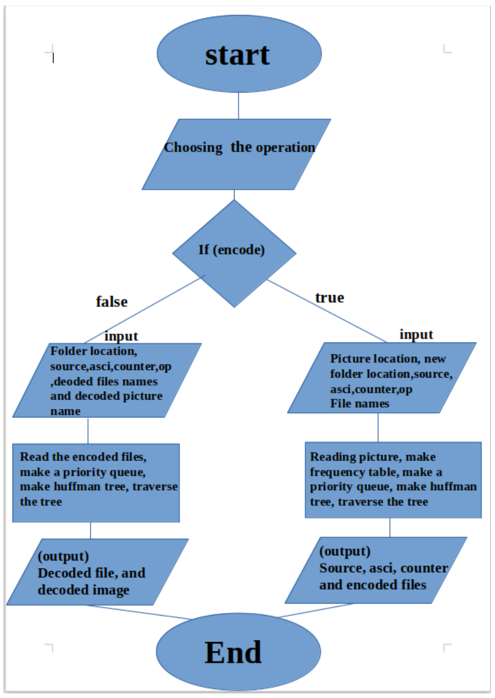
Login

decode is done, please check your folder

That it, we finished, lets take a look on our empty folder, we will see that it contains 4 files and one picture, 3 file from encoding step, and the final file and the picture are from the decoding stage.



Flow chart for the OUR GUI that shows in few words our path from the beginning till the end.


start
Choosing the operation
If (encode)
false
true
input
Folder location, source,asci,counter,op ,deoded files names and decoded picture name
input
Picture location, new folder location,source, asci,counter,op File names
Read the encoded files, make a priority queue, make huffman tree, traverse the tree
Reading picture, make frequency table, make a priority queue, make huffman tree, traverse the tree
(output) Decoded file, and decoded image
(output) Source, asci, counter and encoded files
End

7 Contributions

We need to say first that we didn't make a clear tasks foe every one but we can say that in general each one of us has a major task so that he can help another team member and the rest of members could help him.
1) Abdallah Mohammad Shehata : for Huffman algorithm including encoding and decoding algorithms (build tree.h) file And report making.
2 ,3) Ezzeldeen esmail , Abdallah Tamer: making frequency table and sorting (frequency table.h) and report making
4) pola nagy : getting informations from the picture (pic-in.h)
5) youssef samir : determining the operation and some decoding codes and reading binary encoded file and asci and counter one (main.cpp)

References

[1] B. Furht, ed., *Huffman Coding*, pp. 278–280. Boston, MA: Springer US, 2006.

[2] C. S. Wiki, "Ascii — computer science wiki,," 2019. [Online; accessed 29-May-2020].

[3] Wikipedia contributors, "Pgm — Wikipedia, the free encyclopedia," 2019. [Online; accessed 29-May-2020].

[4] people.sc.fsu.edu, "Binary portable gray map graphics files,," 201q. [Online; accessed 29-May-2020].