國立臺南大學資訊工程學系

資工三「演算法」課程

第四次作業

**題目: Huffman Coding 壓縮**

|  |  |  |
| --- | --- | --- |
| 班級 | ： | 資工三 |
| 姓名 | ： | 曾俊杰 |
| 學號 | ： | S11059003 |

老師：陳宗禧

中華民國 113年1月1日

**目錄**

[(一) 簡介及問題描述 3](#_Toc155016826)

[1. 簡介 3](#_Toc155016827)

[2. 問題 3](#_Toc155016828)

[(二) 理論分析 3](#_Toc155016829)

[1. 壓縮(Compression) 3](#_Toc155016830)

[2. 解壓縮(Uncompression) 3](#_Toc155016831)

[(三) 演算法則 4](#_Toc155016832)

[1. 第一個演算法(Algorithm) 4](#_Toc155016833)

[i. 演算法時間複雜度(time complexity): O() 5](#_Toc155016834)

[ii. 演算法空間複雜度(space complexity): O() 5](#_Toc155016835)

[2. 第二個演算法(Algorithm) 6](#_Toc155016836)

[i. 演算法時間複雜度(time complexity): O() 8](#_Toc155016837)

[ii. 演算法空間複雜度(space complexity): O(1) 8](#_Toc155016838)

[(四) 程式設計環境架構 8](#_Toc155016839)

[1. 程式語言 8](#_Toc155016840)

[2. 程式開發工具 8](#_Toc155016841)

[3. 電腦硬體 8](#_Toc155016842)

[4. 作業系統 8](#_Toc155016843)

[(五) 程式 (含source code, input code, and output code) 9](#_Toc155016844)

[1. 主程式 9](#_Toc155016845)

[2. Input Code Format 24](#_Toc155016846)

[3. Output Code Format 24](#_Toc155016847)

[(1) Test1.pgm 24](#_Toc155016848)

[(2) Test2.pgm 32](#_Toc155016849)

[(3) Test3.pgm 52](#_Toc155016850)

[(六) 執行結果、討論與心得 73](#_Toc155016851)

[1. 執行結果 73](#_Toc155016852)

[2. 討論 73](#_Toc155016853)

[(1) Running Time with n\*m pixels(\_bitset = 32) 73](#_Toc155016854)

[(2) Bit\_size at line 11(#define \_bitset) with compression effection 74](#_Toc155016855)

[(3) Bit\_size at line 11(#define \_bitset) with running time 74](#_Toc155016856)

[3. 心得 75](#_Toc155016857)

[參考文獻 76](#_Toc155016858)

**(一) 簡介及問題描述**

1. 簡介

給定一個 PGM image 檔案，利用 Huffman Coding 演算法設計與實作檔案壓縮程式。求encode\_image(Huffman\*)。

2. 問題

Read\_pgm(vector<vector<int>>, vector<vector<int>>): 讀入所有pixel的資訊

Build\_heap(vector<vector<int>>): 把所有pixel value和frequency建立成min heap

Build\_tree(vector<Huffman\*>): 以min heap的delete min函式建立霍夫曼樹。

Encode\_table(Huffman\*): 為每個pixel value建立相對應的encode table、decode table

Encode\_image(unordered\_map<int, string>, vector<vector<int>>): 把對應的pixel value根據encode table建立成壓縮圖片

Decode\_image(unordered\_map<string, int>): 讀入壓縮圖片，根據decode table解壓縮

**(二) 理論分析**

1. 壓縮(Compression)

先把每一列的pixel value根據encode table轉換成huffman code儲存，查看儲存的字串是否大小是8的倍數，如果不是需要在最尾補0到大小成立為止，同時記錄補齊前後的大小。用bitset<8>的形式將字串以二進制的形式寫到.hc檔裡面，重複動作直到最後一列。

2. 解壓縮(Uncompression)

根據補齊後的大小以bitset<8>的形式讀取二進制壓縮檔案，跟補齊前的大小比較，檢查是否有補0，若有則將多餘的0捨棄，將正確的子字串加到主字串，重複動作直到子字串無法再讀取。先寫入PGM image的版本、備註、長乘寬的像素值和最大灰值，然後處理主字串，用字元為單位逐漸讀入，直到可以從decode table找到對應的值，由於PGM image中僅存在數字、空格、換行三種資料，所以只要判斷不是換行或空格，就能直接寫入數字，換行判斷：ascii code是10，前一個字元是空格，位於該列的最後一行；空格判斷：ascii code是32，前一個字元是數字；其餘的就直接將ascii code輸出即可。

**(三) 演算法則**

1. 第一個演算法(Algorithm)

一張含有 文字, 螢幕擷取畫面, 字型, 設計 的圖片

自動產生的描述

func write\_encode\_file**(**hash**,** pgm, bit\_size**):** write\_len is a 2**-**D array

choose write file with binary format

**for** i **(**0 to pgm**.**size**())** **do**

helper**.**clear**()**

**for** j **(**0 to pgm**[**i**].**size**())** **do**

helper **<-** helper **+** hash**[**pgm**[**i**][**j**]]**

end **for**

write**.**resize**(**i**+**1**)**

write\_len**[**i**].**push**(**helper**.**size**())**

fill **<-** helper**.**size**()** **%** bit\_size

**if** fill **!=** 0 then

fill **<-** bit\_size **-** fill

**for** k **(**0 to fill**)** **do**

helper **<-** helper **+** "0"

end **for**

end **if**

write\_len**[**i**].**push**(**helper**.**size**())**

**for** l **(**0 to helper**.**size**()** step **+**bit\_size**)** **do**

sub **<-** helper**.**substr**(**l**,** bit\_size**)**

bitset**<**bit\_size**>** binary**(**sub**)**

write binary to output file

end **for**

end **for**

close output file

**return** write\_len

1. 演算法時間複雜度(time complexity): O()

Because the nest loop is dependent on the number of rows and columns of pixels, so the worst case is n by n pixel, which is O().

1. 演算法空間複雜度(space complexity): O()

I need a 2-D array to store the length of every row of pixels before and after it go through the padding checking. So there are two elements for every row of pixels, only O(2n).

2. 第二個演算法(Algorithm)

一張含有 文字, 字型, 螢幕擷取畫面, 黑與白 的圖片

自動產生的描述

func read\_encode\_file\_to\_pgm**(**hash**,** information**,** pgm\_len**,** write\_len**,** bit\_size**)**

open the input file

**while** input file has next to read **do**

**for** i **(**0 to write\_len**[**row\_count**][**1**]** **/** bit\_size**)** **do**

all\_helper**.**clear**()**

all\_helper **<-** all\_helper **+** bitset**<**bit\_size**>** read from input file

end **for**

**if** write\_len**[**row\_count**][**1**]** **!=** write\_len**[**row\_count**][**0**]** then

all **<-** all **+** all\_helper**.**substr**(**0**,** write\_len**[**row\_count**][**0**])**

**else**

all **<-** all **+** all\_helper

end **if**

row\_count**++**

end **while**

**for** j **(**0 to information**.**size**())** **do**

write information

end **for**

row\_count **<-** 0

**for** k **(**0 to all**.**size**())** **do**

buffer**.**clear**()**

**while** buffer can't find in hash do

buffer **<-** buffer **+** next char

end **while**

helper **=** hash**[**buffer**]**

**if** helper **=** 10 **and** previous **=** ' ' **and** pgm\_len**[**row\_count**]** **=** column **+** 1 then

write '\n' to output file

previous **<-** '\n'

column\_count **<-** 0

row\_count **<-** row\_count **+** 1

**else** **if** helper **=** 32 **and** previous is a number then

write ' ' to output file

previous **<-** ' '

**else**

write helper to output file

previous **<-** '0'

end **if**

column\_count **<-** column\_count **+** 1

end **for**

close the output file

1. 演算法時間複雜度(time complexity): O()

Because the nest for loop is worst to n\*n pixels, I think the read from binary file won’t over the write operation of decoding. So the time complexity is O().

1. 演算法空間複雜度(space complexity): O(1)

I don’t use any extra array to store data but some variables, so the space complexity is O(1).

**(四) 程式設計環境架構**

1. 程式語言

C++ in Microsoft Windows10 64-bits operating system

2. 程式開發工具

Microsoft Visual Studio 2019

3. 電腦硬體

CPU: 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz 2.30 GHz

Main Memory: 16.0 GB

顯示卡1: Intel® UHD Graphics

顯示卡2: NVIDIA GeForce RTX 3050 Ti Laptop GPU

4. 作業系統

Windows 10 家用版

版本: 22H2

**(五) 程式 (含source code, input code, and output code)**

1. 主程式

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <unordered\_map>

#include <iomanip>

#include <bitset>

#include <sys/stat.h>

**using** **namespace** std**;**

#define \_bitset 32 //如果字串有太多補0，反而會造成size變大

struct huffman **{**

int key**;**

int frequency**;**

struct huffman**\*** left**;**

struct huffman**\*** right**;**

**};**

vector**<**vector**<**int**>>** read\_pgm**(**vector**<**vector**<**int**>>&,** vector**<**string**>&,** vector**<**int**>&);**

void table**(**vector**<**vector**<**int**>>&,** int**);**

vector**<**huffman**\*>** min\_heap**(**vector**<**vector**<**int**>>);**

void heap\_insertion**(**vector**<**huffman**\*>&,** huffman**\*);**

huffman**\*** heap\_min\_deletion**(**vector**<**huffman**>&);**

void swap**(**huffman**\*\*,** huffman**\*\*);**

void build\_huffman\_tree**(**vector**<**huffman**\*>&);**

huffman**\*** new\_huffman\_node**(**int**,** int**);**

huffman**\*** joint**(**huffman**\*,** huffman**\*);**

void build\_encode\_table**(**huffman**\*,** unordered\_map**<**int**,** string**>&,** unordered\_map**<**string**,** int**>&,** string**,** vector**<**string**>&);**

void write\_encode\_file**(**unordered\_map**<**int**,** string**>,** vector**<**vector**<**int**>>,** vector**<**vector**<**int**>>&);**

void read\_encode\_file\_to\_pgm**(**unordered\_map**<**string**,** int**>,** vector**<**string**>,** vector**<**int**>,** vector**<**vector**<**int**>>&);**

void print\_frequency**(**vector**<**vector**<**int**>>);**

void sort\_and\_print\_hash\_table**(**vector**<**string**>,** unordered\_map**<**string**,** int**>);**

void quick\_sort**(**vector**<**string**>&,** int**,** int**);**

void print\_file\_size**(**string**,** string**);**

int main**(**void**)** **{**

vector**<**vector**<**int**>>** key**,** pgm**,** write\_len**;**

vector**<**int**>** pgm\_length**;**

vector**<**string**>** information**,** hash\_code**;**

vector**<**huffman**\*>** tree**;**

unordered\_map**<**int**,** string**>** encode**;**

unordered\_map**<**string**,** int**>** decode**;**

double start**,** end**;**

start **=** clock**();**

pgm **=** read\_pgm**(**key**,** information**,** pgm\_length**);**

tree **=** min\_heap**(**key**);**

build\_huffman\_tree**(**tree**);**

build\_encode\_table**(**tree**[**0**],** encode**,** decode**,** ""**,** hash\_code**);**

write\_encode\_file**(**encode**,** pgm**,** write\_len**);**

read\_encode\_file\_to\_pgm**(**decode**,** information**,** pgm\_length**,** write\_len**);**

print\_frequency**(**key**);**

sort\_and\_print\_hash\_table**(**hash\_code**,** decode**);**

end **=** clock**();**

cout **<<** "execution time: " **<<** end **-** start **<<** "ms.\n"**;**

**return** 0**;**

**}**

//讀pgm, 拿到pixel value+frequency, 前4列, 每列行數

vector**<**vector**<**int**>>** read\_pgm**(**vector**<**vector**<**int**>>&** key**,** vector**<**string**>&** information**,** vector**<**int**>&** pgm\_len**)** **{**

int i**;**

int row\_count **=** 1**,** count **=** 1**,** element **=** 0**;** //count: 指向最後一個value的index

string file\_name**,** buffer**,** checker**;**

vector**<**vector**<**int**>>** pgm**;**

cout **<<** "Enter the input file name with .pgm format: "**;**

cin **>>** file\_name**;**

cout **<<** "\n"**;**

ifstream file**(**file\_name**);**

getchar**();** //清空換行符號

print\_file\_size**(**file\_name**,** "original"**);**

**if** **(!**file**.**is\_open**())** **{**

cout **<<** "Failed to open the file.\n"**;**

exit**(**1**);**

**}**

**for** **(**i **=** 0**;** i **<** 4**;** i**++)** **{** //版本+備註+長\*寬+最大灰度

getline**(**file**,** buffer**);**

information**.**push\_back**(**buffer**);**

**}**

buffer**.**clear**();**

pgm**.**resize**(**row\_count**);**

**while** **(**getline**(**file**,** buffer**))** **{**

**if** **(!**file**.**eof**())** **{** //getline會吃掉換行

buffer **+=** "\n"**;**

**}**

pgm**.**resize**(**row\_count**);**

**for** **(**i **=** 0**;** i **<** buffer**.**size**();** i**++)** **{** //value是空格、換行或數字

**if** **(**buffer**[**i**]** **==** ' '**)** **{**

element **=** **(**int**)**' '**;**

pgm**[**row\_count **-** 1**].**push\_back**(**element**);**

**}**

**else** **if** **(**buffer**[**i**]** **==** '\n'**)** **{**

element **=** **(**int**)**'\n'**;**

pgm**[**row\_count **-** 1**].**push\_back**(**element**);**

row\_count**++;**

**}**

**else** **{**

**for** **(**checker**.**clear**();** isdigit**(**buffer**[**i**]);** i**++)** **{**

checker **+=** buffer**[**i**];**

**}**

i**--;**

element **=** stoi**(**checker**);**

pgm**[**row\_count **-** 1**].**push\_back**(**element**);**

**}**

table**(**key**,** element**);**

**}**

pgm\_len**.**push\_back**(**pgm**[**row\_count **-** 2**].**size**());** //一列有多少element

buffer**.**clear**();**

**}**

file**.**close**();**

**return** pgm**;**

**}**

void table**(**vector**<**vector**<**int**>>&** key**,** int element**)** **{** //創建pixel value+frequency

int i**;**

**for** **(**i **=** 0**;** i **<** key**.**size**();** i**++)** **{** //重複value，frquency+1

**if** **(!**key**[**i**].**empty**()** **&&** key**[**i**][**0**]** **==** element**)** **{**

key**[**i**][**1**]++;**

**return;**

**}**

**}**

key**.**push\_back**({** element**,** 1 **});** //新value，frequency是1

**}**

vector**<**huffman**\*>** min\_heap**(**vector**<**vector**<**int**>>** key**)** **{**

vector**<**huffman**\*>** result**;**

huffman**\*** buffer**;**

int i**;**

**for** **(**i **=** 0**;** i **<** key**.**size**();** i**++)** **{**

buffer **=** new\_huffman\_node**(**key**[**i**][**0**],** key**[**i**][**1**]);**

heap\_insertion**(**result**,** buffer**);**

**}**

**return** result**;**

**}**

void heap\_insertion**(**vector**<**huffman**\*>&** heap**,** huffman**\*** buffer**)** **{**

int current **=** heap**.**size**();**

heap**.**push\_back**(**buffer**);** //insert到最尾

**while** **(**heap**[**current**]->**frequency **<** heap**[(**current **-** 1**)** **/** 2**]->**frequency**)** **{** //檢查有沒有比父小

swap**(&**heap**[**current**],** **&**heap**[**current **/** 2**]);**

current **=** **(**current **-** 1**)** **/** 2**;** //更新index

**}**

**}**

huffman**\*** heap\_min\_deletion**(**vector**<**huffman**\*>&** heap**)** **{**

int current **=** 0**;**

int last **=** heap**.**size**()** **-** 1**;**

huffman**\*** result **=** heap**[**0**];**

swap**(&**heap**[**0**],** **&**heap**[**last**]);** //root跟最尾交換

heap**.**erase**(**heap**.**begin**()** **+** last**);** //最尾(最小root)消除

**while** **((**2 **\*** current **+** 1**)** **<** heap**.**size**())** **{** //有左邊才能換，沒左邊一定沒右邊

**if** **((**2 **\*** current **+** 2**)** **<** heap**.**size**()** **&&** heap**[**2 **\*** current **+** 2**]->**frequency **<** heap**[**2 **\*** current **+** 1**]->**frequency **&&** heap**[**2 **\*** current **+** 2**]->**frequency **<** heap**[**current**]->**frequency**)** **{** //右存在 + 右 < 左 + 右 < 父

swap**(&**heap**[**current**],** **&**heap**[**2 **\*** current **+** 2**]);**

current **=** 2 **\*** current **+** 2**;**

**}**

**else** **if** **(**heap**[**current**]->**frequency **>** heap**[**2 **\*** current **+** 1**]->**frequency**)** **{** //左 < 父

swap**(&**heap**[**current**],** **&**heap**[**2 **\*** current **+** 1**]);**

current **=** 2 **\*** current **+** 1**;**

**}**

**else** **if** **(**heap**[**current**]->**frequency **<=** heap**[**2 **\*** current **+** 1**]->**frequency**)** **{** //左右都無法swap

**break;**

**}**

**}**

**return** result**;**

**}**

void swap**(**huffman**\*\*** a**,** huffman**\*\*** b**)** **{**

huffman**\*** temp**;**

temp **=** **\***a**;**

**\***a **=** **\***b**;**

**\***b **=** temp**;**

**}**

void build\_huffman\_tree**(**vector**<**huffman**\*>&** heap**)** **{**

huffman**\*** a**,** **\*** b**,** **\*** buffer**;**

**while** **(**heap**.**size**()** **>** 1**)** **{**

a **=** heap\_min\_deletion**(**heap**);**

b **=** heap\_min\_deletion**(**heap**);**

buffer **=** joint**(**a**,** b**);**

heap\_insertion**(**heap**,** buffer**);**

**}**

**}**

huffman**\*** new\_huffman\_node**(**int key**,** int frequency**)** **{**

huffman**\*** result **=** **new** huffman**();**

result**->**key **=** key**;**

result**->**frequency **=** frequency**;**

result**->**left **=** **NULL;**

result**->**right **=** **NULL;**

**return** result**;**

**}**

huffman**\*** joint**(**huffman**\*** a**,** huffman**\*** b**)** **{**

huffman**\*** root **=** **new** huffman**;**

root**->**key **=** **-**1**;** //不是葉子(external)節點，假設沒有灰值是-1的pixel

root**->**frequency **=** a**->**frequency **+** b**->**frequency**;**

root**->**left **=** a**;**

root**->**right **=** b**;**

**return** root**;**

**}**

void build\_encode\_table**(**huffman**\*** tree**,** unordered\_map**<**int**,** string**>&** encode**,** unordered\_map**<**string**,** int**>&** decode**,** string code**,** vector**<**string**>&** codes**)** **{**

**if** **(**tree**)** **{** //Inorder拜訪

build\_encode\_table**(**tree**->**left**,** encode**,** decode**,** **(**code **+** "0"**),** codes**);**

**if** **(**tree**->**key **!=** **-**1**)** **{**

encode**[**tree**->**key**]** **=** code**;**

decode**[**code**]** **=** tree**->**key**;**

codes**.**push\_back**(**code**);**

**}**

build\_encode\_table**(**tree**->**right**,** encode**,** decode**,** **(**code **+** "1"**),** codes**);**

**}**

**}**

//壓縮圖片

void write\_encode\_file**(**unordered\_map**<**int**,** string**>** hash**,** vector**<**vector**<**int**>>** pgm**,** vector**<**vector**<**int**>>&** write\_len**)** **{**

ofstream output\_file**(**"test.hc"**,** ios**::**binary**);**

string helper**,** sub**;**

int i**,** j**,** fill**;**

**for** **(**i **=** 0**;** i **<** pgm**.**size**();** i**++)** **{**

helper**.**clear**();**

**for** **(**j **=** 0**;** j **<** pgm**[**i**].**size**();** j**++)** **{**

helper **+=** hash**[**pgm**[**i**][**j**]];**

**}**

write\_len**.**resize**(**i **+** 1**);**

write\_len**[**i**].**push\_back**(**helper**.**size**());**

fill **=** helper**.**size**()** **%** \_bitset**;** //如果不是倍數個bit，最尾補0

**if** **(**fill **!=** 0**)** **{**

fill **=** \_bitset **-** fill**;**

**for** **(**j **=** 0**;** j **<** fill**;** j**++)** **{**

helper **+=** "0"**;**

**}**

**}**

write\_len**[**i**].**push\_back**(**helper**.**size**());**

**for** **(**j **=** 0**;** j **<** helper**.**size**();** j **+=** \_bitset**)** **{**

sub **=** helper**.**substr**(**j**,** \_bitset**);**

bitset**<**\_bitset**>** binary**(**sub**.**c\_str**());**

output\_file**.**write**(reinterpret\_cast<**const char**\*>(&**binary**),** **sizeof(**binary**));**

**}**

**}**

print\_file\_size**(**"test.hc"**,** "compressed"**);**

cout **<<** "\n"**;**

output\_file**.**close**();**

**}**

//解壓縮圖片

void read\_encode\_file\_to\_pgm**(**unordered\_map**<**string**,** int**>** hash**,** vector**<**string**>** information**,** vector**<**int**>** pgm\_len**,** vector**<**vector**<**int**>>&** write\_len**)** **{**

int i**,** helper**;**

int row\_count **=** 0**,** column\_count **=** 0**;**

char previous**;** //previous: 紀錄上一個是數字、空格或換行

string all**,** all\_helper**,** buffer**;**

ifstream input\_file**(**"test.hc"**,** ios**::**binary**);**

ofstream output\_file**(**"testd.pgm"**);**

bitset**<**\_bitset**>** binary**;**

**while** **(**input\_file**.**read**(reinterpret\_cast<**char**\*>(&**binary**),** **sizeof(**binary**)))** **{** //一次能讀bit\_size個bit

all\_helper**.**clear**();**

all\_helper **+=** binary**.**to\_string**();**

**for** **(**i **=** 1**;** i **<** write\_len**[**row\_count**][**1**]** **/** \_bitset**;** i**++)** **{**

input\_file**.**read**(reinterpret\_cast<**char**\*>(&**binary**),** **sizeof(**binary**));**

all\_helper **+=** binary**.**to\_string**();**

**}**

**if** **(**write\_len**[**row\_count**][**1**]** **!=** write\_len**[**row\_count**][**0**])** **{** //判斷最尾端是否有補0

all **+=** all\_helper**.**substr**(**0**,** write\_len**[**row\_count**][**0**]);**

**}**

**else** **{**

all **+=** all\_helper**;**

**}**

row\_count**++;**

**}**

**for** **(**i **=** 0**;** i **<** information**.**size**();** i**++)** **{** //寫入版本+備註+長\*寬+最大灰值

output\_file **<<** information**[**i**];**

output\_file **<<** "\n"**;**

**}**

row\_count **=** 0**;**

**for(**i **=** 0**;** i **<** all**.**size**();** i**++)** **{**

buffer**.**clear**();**

**for** **(**buffer **+=** all**[**i**];** hash**.**find**(**buffer**)** **==** hash**.**end**();** buffer **+=** all**[**i**])** **{**

i**++;**

**}**

helper **=** hash**[**buffer**];**

**if** **(**helper **==** 10 **&&** previous **==** ' ' **&&** pgm\_len**[**row\_count**]** **==** column\_count **+** 1**)** **{** //換行判斷，換行只出現在每列最尾

output\_file **<<** char**(**helper**);**

previous **=** char**(**helper**);**

column\_count **=** 0**;**

row\_count**++;**

**continue;**

**}**

**else** **if** **(**helper **==** 32 **&&** isdigit**(**previous**))** **{** //空格判斷，空格只出現在數字後面

output\_file **<<** char**(**helper**);**

previous **=** char**(**helper**);**

**}**

**else** **{** //數字判斷

output\_file **<<** helper**;**

previous **=** '0'**;**

**}**

column\_count**++;**

**}**

**}**

void print\_frequency**(**vector**<**vector**<**int**>>** key**)** **{**

int i**;**

cout **<<** "Keys with their frequencies shown:\n"**;**

**for** **(**i **=** 0**;** i **<** key**.**size**();** i**++)** **{**

cout **<<** setiosflags**(**ios**::**left**)** **<<** setw**(**5**)** **<<** key**[**i**][**0**]** **<<** " appeared " **<<** setw**(**6**)** **<<** key**[**i**][**1**]** **<<** " times." **<<** "\n"**;**

**}**

cout **<<** "\n\n"**;**

**}**

void sort\_and\_print\_hash\_table**(**vector**<**string**>** code**,** unordered\_map**<**string**,** int**>** hash**)** **{**

int i**;**

quick\_sort**(**code**,** 0**,** code**.**size**()** **-** 1**);**

cout **<<** "Encoded codes with pixel values shown:\n"**;**

**for** **(**i **=** 0**;** i **<** code**.**size**();** i**++)** **{**

cout **<<** setw**(**15**)** **<<** code**[**i**]** **<<** " is " **<<** setw**(**5**)** **<<** hash**[**code**[**i**]]** **<<** "\n"**;**

**}**

cout **<<** "\n\n"**;**

**}**

void quick\_sort**(**vector**<**string**>&** code**,** int left**,** int right**)** **{**

**if** **(**left **>=** right**)** **{**

**return;**

**}**

int i **=** left**;**

int j **=** right**;**

string key **=** code**[**left**];**

**while** **(**i **!=** j**)** **{**

**while** **(**i **<** j **&&** code**[**j**]** **>** key**)** **{**

j **-=** 1**;**

**}**

**while** **(**i **<** j **&&** code**[**i**]** **<** key**)** **{**

i**++;**

**}**

**if** **(**i **<** j**)** **{**

string temp**;**

temp **=** code**[**i**];**

code**[**i**]** **=** code**[**j**];**

code**[**j**]** **=** temp**;**

**}**

**}**

code**[**left**]** **=** code**[**i**];**

code**[**i**]** **=** key**;**

quick\_sort**(**code**,** i **+** 1**,** right**);**

quick\_sort**(**code**,** left**,** i **-** 1**);**

**}**

void print\_file\_size**(**string file\_name**,** string status**)** **{**

struct stat stat\_buffer**;**

int size **=** stat**(**file\_name**.**c\_str**(),** **&**stat\_buffer**);** //抓取返回值，0是成功

**if** **(**size **==** 0**)** **{**

cout **<<** "The " **<<** status **<<** " file size is " **<<** stat\_buffer**.**st\_size **<<** " bytes\n"**;**

**}**

**else** **{**

cout **<<** "Can't not get the file size\n"**;**

**}**

**}**

2. Input Code Format

Enter the input file name with .pgm format:

(1) Test1.pgm

(2) Test2.pgm

(3) Test3.pgm

3. Output Code Format

* Key with its frequency ordered by “time to enter the vector”
* Pixel value with its encode code order by “encode code string comparison”
* The file size format is given as the following example graph.

一張含有 文字, 螢幕擷取畫面, 軟體, 數字 的圖片

自動產生的描述

(1) Test1.pgm

The original file size is 233798 bytes

The compressed file size is 40960 bytes

Keys with their frequencies shown:

255 appeared 35144 times.

32 appeared 57600 times.

10 appeared 3360 times.

254 appeared 5747 times.

252 appeared 1013 times.

251 appeared 613 times.

253 appeared 1957 times.

250 appeared 524 times.

248 appeared 297 times.

249 appeared 282 times.

240 appeared 124 times.

226 appeared 97 times.

231 appeared 75 times.

220 appeared 54 times.

239 appeared 117 times.

210 appeared 65 times.

244 appeared 160 times.

247 appeared 234 times.

243 appeared 185 times.

223 appeared 65 times.

207 appeared 91 times.

196 appeared 89 times.

189 appeared 235 times.

212 appeared 61 times.

237 appeared 111 times.

175 appeared 67 times.

183 appeared 769 times.

202 appeared 74 times.

229 appeared 66 times.

238 appeared 95 times.

224 appeared 63 times.

195 appeared 87 times.

185 appeared 530 times.

184 appeared 600 times.

178 appeared 293 times.

234 appeared 101 times.

180 appeared 485 times.

176 appeared 111 times.

217 appeared 46 times.

246 appeared 215 times.

215 appeared 74 times.

209 appeared 75 times.

186 appeared 441 times.

182 appeared 835 times.

245 appeared 184 times.

187 appeared 376 times.

174 appeared 60 times.

206 appeared 51 times.

225 appeared 60 times.

192 appeared 127 times.

233 appeared 93 times.

181 appeared 649 times.

191 appeared 137 times.

204 appeared 62 times.

188 appeared 257 times.

177 appeared 247 times.

190 appeared 163 times.

208 appeared 71 times.

193 appeared 140 times.

205 appeared 68 times.

179 appeared 375 times.

221 appeared 52 times.

216 appeared 54 times.

203 appeared 72 times.

213 appeared 85 times.

228 appeared 76 times.

194 appeared 86 times.

241 appeared 128 times.

173 appeared 35 times.

170 appeared 9 times.

242 appeared 192 times.

201 appeared 84 times.

222 appeared 102 times.

211 appeared 66 times.

235 appeared 83 times.

198 appeared 99 times.

230 appeared 79 times.

200 appeared 72 times.

197 appeared 103 times.

199 appeared 87 times.

232 appeared 84 times.

172 appeared 24 times.

218 appeared 60 times.

214 appeared 90 times.

227 appeared 89 times.

236 appeared 92 times.

219 appeared 60 times.

171 appeared 20 times.

166 appeared 3 times.

169 appeared 12 times.

168 appeared 8 times.

162 appeared 1 times.

167 appeared 3 times.

164 appeared 2 times.

163 appeared 1 times.

Encoded codes with pixel values shown:

0 is 32

10 is 255

11000000 is 248

110000010 is 191

110000011 is 190

1100001 is 184

1100010 is 251

1100011000 is 215

1100011001 is 209

110001101 is 244

1100011100 is 163

11000111010 is 169

11000111011 is 223

1100011110 is 239

1100011111 is 240

1100100 is 181

1100101000 is 230

1100101001 is 194

1100101010 is 232

1100101011 is 213

110010110 is 243

1100101110 is 231

1100101111 is 202

1100110000 is 235

1100110001 is 214

1100110010 is 233

1100110011 is 201

1100110100 is 227

1100110101 is 196

1100110110 is 207

1100110111 is 236

11001110 is 179

11001111 is 187

1101 is 254

1110000 is 183

1110001000 is 199

1110001001 is 234

11100010100 is 170

111000101010 is 162

1110001010110 is 212

1110001010111 is 164

11100010110 is 224

11100010111 is 216

111000110 is 242

1110001110 is 222

1110001111 is 197

1110010 is 182

111001100 is 246

1110011010 is 176

11100110110 is 221

11100110111 is 219

11100111 is 186

11101 is 10

111100 is 253

111101000 is 247

111101001 is 189

11110101 is 180

111101100 is 177

11110110100 is 198

11110110101 is 172

1111011011 is 241

1111011100 is 192

111101110100 is 210

111101110101 is 166

11110111011 is 225

111101111 is 188

1111100 is 252

11111010 is 250

111110110000 is 167

111110110001 is 174

11111011001 is 211

11111011010 is 173

11111011011 is 238

11111011100 is 175

11111011101 is 229

11111011110 is 195

11111011111 is 217

11111100 is 185

11111101000 is 218

11111101001 is 228

11111101010 is 205

111111010110 is 168

111111010111 is 204

111111011 is 249

1111111000 is 193

11111110010 is 208

11111110011 is 203

111111101 is 178

11111111000 is 200

111111110010 is 171

111111110011 is 220

11111111010 is 226

11111111011 is 206

1111111110 is 237

1111111111 is 245

execution time: 2997ms.

(2) Test2.pgm

The original file size is 730995 bytes

The compressed file size is 286720 bytes

Keys with their frequencies shown:

0 appeared 98139 times.

32 appeared 265203 times.

10 appeared 32539 times.

1 appeared 5397 times.

2 appeared 3265 times.

3 appeared 2371 times.

5 appeared 1804 times.

7 appeared 2142 times.

13 appeared 1660 times.

11 appeared 3461 times.

12 appeared 2653 times.

17 appeared 1100 times.

22 appeared 916 times.

23 appeared 868 times.

37 appeared 888 times.

65 appeared 714 times.

56 appeared 764 times.

51 appeared 933 times.

55 appeared 885 times.

54 appeared 932 times.

60 appeared 819 times.

81 appeared 434 times.

98 appeared 256 times.

92 appeared 283 times.

79 appeared 414 times.

69 appeared 620 times.

68 appeared 659 times.

70 appeared 617 times.

72 appeared 583 times.

85 appeared 374 times.

99 appeared 263 times.

75 appeared 498 times.

90 appeared 296 times.

86 appeared 351 times.

78 appeared 454 times.

82 appeared 380 times.

77 appeared 450 times.

66 appeared 662 times.

63 appeared 796 times.

64 appeared 728 times.

74 appeared 521 times.

57 appeared 806 times.

50 appeared 1134 times.

44 appeared 1046 times.

35 appeared 1146 times.

31 appeared 1085 times.

29 appeared 904 times.

6 appeared 2405 times.

4 appeared 2329 times.

15 appeared 1265 times.

21 appeared 912 times.

20 appeared 1026 times.

30 appeared 1173 times.

73 appeared 561 times.

49 appeared 1223 times.

53 appeared 890 times.

59 appeared 858 times.

43 appeared 944 times.

52 appeared 961 times.

45 appeared 1163 times.

47 appeared 1090 times.

58 appeared 802 times.

46 appeared 1211 times.

80 appeared 410 times.

61 appeared 727 times.

42 appeared 966 times.

48 appeared 1160 times.

71 appeared 557 times.

41 appeared 951 times.

62 appeared 725 times.

76 appeared 434 times.

83 appeared 379 times.

28 appeared 1294 times.

39 appeared 927 times.

25 appeared 829 times.

14 appeared 1526 times.

27 appeared 879 times.

91 appeared 277 times.

88 appeared 372 times.

96 appeared 252 times.

102 appeared 266 times.

67 appeared 703 times.

94 appeared 291 times.

84 appeared 351 times.

110 appeared 314 times.

113 appeared 449 times.

38 appeared 932 times.

34 appeared 1032 times.

18 appeared 1039 times.

8 appeared 3479 times.

9 appeared 5441 times.

40 appeared 936 times.

97 appeared 256 times.

95 appeared 279 times.

89 appeared 267 times.

112 appeared 352 times.

108 appeared 298 times.

107 appeared 297 times.

93 appeared 278 times.

33 appeared 1027 times.

121 appeared 399 times.

105 appeared 325 times.

24 appeared 842 times.

19 appeared 964 times.

16 appeared 1222 times.

123 appeared 327 times.

104 appeared 268 times.

116 appeared 392 times.

101 appeared 238 times.

106 appeared 255 times.

111 appeared 315 times.

118 appeared 333 times.

26 appeared 920 times.

143 appeared 367 times.

132 appeared 338 times.

119 appeared 373 times.

133 appeared 273 times.

87 appeared 302 times.

109 appeared 272 times.

126 appeared 334 times.

125 appeared 325 times.

142 appeared 311 times.

115 appeared 417 times.

103 appeared 233 times.

122 appeared 337 times.

36 appeared 895 times.

120 appeared 337 times.

127 appeared 358 times.

124 appeared 418 times.

117 appeared 367 times.

114 appeared 317 times.

100 appeared 256 times.

128 appeared 356 times.

137 appeared 326 times.

129 appeared 341 times.

135 appeared 351 times.

141 appeared 305 times.

140 appeared 418 times.

138 appeared 272 times.

130 appeared 342 times.

134 appeared 308 times.

139 appeared 318 times.

152 appeared 189 times.

161 appeared 160 times.

153 appeared 195 times.

144 appeared 267 times.

156 appeared 205 times.

151 appeared 208 times.

154 appeared 208 times.

147 appeared 255 times.

167 appeared 178 times.

183 appeared 134 times.

172 appeared 130 times.

145 appeared 256 times.

131 appeared 310 times.

162 appeared 172 times.

175 appeared 148 times.

146 appeared 294 times.

148 appeared 246 times.

174 appeared 160 times.

159 appeared 181 times.

166 appeared 158 times.

149 appeared 219 times.

181 appeared 171 times.

168 appeared 159 times.

178 appeared 148 times.

173 appeared 146 times.

163 appeared 166 times.

160 appeared 176 times.

169 appeared 150 times.

165 appeared 168 times.

157 appeared 156 times.

177 appeared 159 times.

170 appeared 145 times.

158 appeared 189 times.

164 appeared 240 times.

180 appeared 139 times.

136 appeared 303 times.

150 appeared 194 times.

186 appeared 122 times.

198 appeared 132 times.

155 appeared 183 times.

190 appeared 117 times.

206 appeared 131 times.

192 appeared 145 times.

171 appeared 155 times.

184 appeared 138 times.

182 appeared 143 times.

201 appeared 143 times.

197 appeared 147 times.

200 appeared 151 times.

179 appeared 146 times.

176 appeared 150 times.

204 appeared 127 times.

195 appeared 151 times.

205 appeared 123 times.

203 appeared 156 times.

187 appeared 127 times.

191 appeared 116 times.

194 appeared 155 times.

196 appeared 147 times.

188 appeared 141 times.

185 appeared 116 times.

193 appeared 208 times.

211 appeared 124 times.

202 appeared 120 times.

199 appeared 125 times.

210 appeared 131 times.

189 appeared 131 times.

207 appeared 125 times.

217 appeared 159 times.

223 appeared 193 times.

232 appeared 64 times.

239 appeared 44 times.

241 appeared 43 times.

242 appeared 47 times.

243 appeared 39 times.

244 appeared 118 times.

245 appeared 51 times.

246 appeared 47 times.

247 appeared 45 times.

248 appeared 60 times.

249 appeared 66 times.

250 appeared 78 times.

251 appeared 129 times.

252 appeared 106 times.

240 appeared 34 times.

238 appeared 26 times.

235 appeared 34 times.

229 appeared 78 times.

226 appeared 153 times.

221 appeared 145 times.

213 appeared 151 times.

208 appeared 125 times.

209 appeared 120 times.

255 appeared 19047 times.

254 appeared 207 times.

237 appeared 42 times.

233 appeared 41 times.

227 appeared 111 times.

220 appeared 140 times.

215 appeared 135 times.

253 appeared 103 times.

236 appeared 42 times.

224 appeared 180 times.

234 appeared 50 times.

230 appeared 59 times.

216 appeared 143 times.

218 appeared 159 times.

228 appeared 78 times.

214 appeared 141 times.

222 appeared 151 times.

212 appeared 127 times.

225 appeared 163 times.

231 appeared 62 times.

219 appeared 151 times.

Encoded codes with pixel values shown:

0 is 32

100000000 is 40

1000000010 is 172

1000000011 is 154

1000000100 is 87

1000000101 is 126

1000000110 is 106

10000001110 is 227

100000011110 is 253

100000011111 is 162

1000001000 is 150

1000001001 is 77

10000010100 is 226

10000010101 is 134

1000001011 is 158

10000011 is 28

1000010 is 12

1000011000 is 105

1000011001 is 123

100001101 is 68

1000011100 is 81

1000011101 is 98

1000011110 is 130

1000011111 is 135

100010 is 1

100011 is 9

1001000000 is 108

1001000001 is 121

1001000010 is 111

10010000110 is 240

10010000111 is 84

100100010 is 67

10010001100 is 160

10010001101 is 224

1001000111 is 127

10010010000 is 245

100100100010 is 181

100100100011 is 149

10010010010 is 184

10010010011 is 182

100100101 is 62

10010011 is 14

100101000 is 51

1001010010 is 90

1001010011 is 114

1001010100 is 69

1001010101 is 195

1001010110 is 112

10010101110 is 218

10010101111 is 91

10010110 is 7

100101110 is 103

100101111000 is 221

100101111001 is 156

10010111101 is 124

1001011111 is 237

100110000 is 63

100110001 is 58

10011001000 is 223

10011001001 is 193

10011001010 is 92

10011001011 is 199

100110011 is 60

1001101 is 2

1001110000 is 148

100111000100 is 185

100111000101 is 211

10011100011 is 118

1001110010 is 167

10011100110 is 125

10011100111 is 137

10011101 is 13

1001111 is 11

1010000 is 8

10100010000 is 249

10100010001 is 119

10100010010 is 248

10100010011 is 82

101000101 is 55

101000110 is 37

101000111 is 53

101001000 is 236

1010010010 is 23

1010010011 is 27

101001010 is 36

10100101100 is 100

10100101101 is 214

10100101110 is 231

101001011110 is 79

101001011111 is 238

1010011000 is 145

10100110010 is 74

10100110011 is 186

1010011010 is 152

1010011011 is 65

10100111 is 5

101010000 is 29

101010001 is 21

1010100100 is 146

10101001010 is 116

101010010110 is 232

101010010111 is 188

101010011 is 38

101010100 is 22

10101010100 is 80

10101010101 is 96

1010101011 is 147

101010110 is 26

101010111 is 39

10101100000 is 128

10101100001 is 252

10101100010 is 213

10101100011 is 122

101011001 is 41

1010110100 is 113

101011010100 is 198

101011010101 is 190

10101101011 is 97

101011011 is 52

101011100 is 42

101011101000 is 230

101011101001 is 169

10101110101 is 102

1010111011 is 75

10101111000 is 183

10101111001 is 166

101011110100 is 216

101011110101 is 222

101011110110 is 212

101011110111 is 144

101011111 is 19

1011 is 10

110000000 is 99

110000001000 is 109

110000001001 is 132

110000001010 is 208

110000001011 is 209

1100000011 is 59

11000001000 is 165

11000001001 is 234

1100000101 is 56

110000011 is 20

110000100 is 33

110000101 is 34

1100001100 is 140

110000110100 is 251

110000110101 is 94

11000011011 is 153

1100001110 is 66

1100001111 is 88

11000100000 is 244

110001000010 is 229

110001000011 is 139

11000100010 is 85

11000100011 is 179

110001001 is 18

110001010 is 44

11000101100 is 254

110001011010 is 161

110001011011 is 225

11000101110 is 89

110001011110 is 206

110001011111 is 201

1100011000 is 43

1100011001 is 204

11000110100 is 104

11000110101 is 250

1100011011 is 61

11000111000 is 174

11000111001 is 163

11000111010 is 136

110001110110 is 207

1100011101110 is 247

11000111011110 is 187

11000111011111 is 203

110001111 is 31

1100100000 is 197

11001000010 is 131

110010000110 is 177

11001000011100 is 246

11001000011101 is 219

1100100001111 is 133

110010001 is 47

1100100100 is 117

1100100101 is 64

110010011 is 17

1100101000 is 73

11001010010 is 176

11001010011 is 115

110010101 is 50

11001011000 is 168

110010110010 is 170

110010110011 is 180

11001011010 is 164

11001011011 is 78

1100101110 is 54

1100101111 is 151

1100110000 is 120

1100110001 is 57

110011001 is 35

110011010 is 48

11001101100 is 141

11001101101 is 138

1100110111 is 72

1100111000 is 83

110011100100 is 110

1100111001010 is 220

1100111001011 is 159

11001110011 is 175

110011101 is 45

1100111100 is 233

11001111010 is 215

110011110110 is 235

110011110111 is 143

1100111110 is 228

11001111110 is 25

11001111111 is 24

11010 is 255

11011000 is 4

110110010000 is 189

110110010001 is 194

11011001001 is 107

11011001010 is 200

11011001011 is 76

110110011 is 30

11011010 is 3

11011011000 is 95

110110110010 is 192

110110110011 is 171

11011011010 is 155

110110110110 is 217

110110110111 is 93

1101101110000 is 205

1101101110001 is 239

110110111001 is 243

110110111010 is 210

1101101110110 is 191

1101101110111 is 196

110110111100 is 241

110110111101 is 202

1101101111100 is 242

1101101111101 is 157

110110111111 is 101

11011100 is 6

110111010 is 46

110111011 is 16

110111100000 is 178

110111100001 is 173

11011110001 is 142

1101111001 is 70

110111101 is 49

1101111100 is 71

11011111010 is 129

11011111011 is 86

110111111 is 15

111 is 0

execution time: 11225ms.

(3) Test3.pgm

The original file size is 6718100 bytes

The compressed file size is 2801664 bytes

Keys with their frequencies shown:

22 appeared 21168 times.

32 appeared 2122249 times.

23 appeared 20036 times.

28 appeared 28165 times.

29 appeared 28586 times.

21 appeared 21271 times.

61 appeared 14445 times.

90 appeared 8543 times.

82 appeared 9648 times.

89 appeared 8585 times.

87 appeared 9258 times.

93 appeared 8491 times.

10 appeared 144842 times.

80 appeared 10176 times.

86 appeared 9341 times.

85 appeared 9905 times.

79 appeared 10571 times.

84 appeared 9880 times.

64 appeared 14531 times.

16 appeared 18067 times.

78 appeared 11009 times.

104 appeared 6403 times.

92 appeared 8651 times.

91 appeared 8494 times.

65 appeared 13566 times.

33 appeared 24217 times.

13 appeared 16411 times.

37 appeared 26676 times.

99 appeared 7197 times.

88 appeared 8718 times.

100 appeared 7196 times.

77 appeared 10964 times.

96 appeared 7493 times.

81 appeared 9902 times.

62 appeared 14465 times.

15 appeared 18146 times.

11 appeared 14543 times.

20 appeared 19655 times.

24 appeared 21304 times.

27 appeared 25788 times.

26 appeared 22636 times.

3 appeared 14239 times.

4 appeared 14868 times.

2 appeared 16726 times.

7 appeared 14684 times.

9 appeared 14612 times.

40 appeared 23634 times.

14 appeared 17229 times.

17 appeared 16582 times.

18 appeared 17624 times.

19 appeared 18491 times.

105 appeared 6274 times.

209 appeared 2042 times.

195 appeared 2357 times.

197 appeared 2285 times.

200 appeared 2271 times.

196 appeared 2259 times.

198 appeared 2217 times.

203 appeared 2198 times.

141 appeared 3455 times.

146 appeared 3314 times.

159 appeared 2712 times.

144 appeared 3405 times.

129 appeared 3981 times.

130 appeared 3912 times.

123 appeared 4405 times.

176 appeared 2357 times.

194 appeared 2323 times.

193 appeared 2410 times.

191 appeared 2394 times.

45 appeared 21672 times.

49 appeared 22957 times.

46 appeared 19537 times.

38 appeared 23225 times.

39 appeared 23238 times.

52 appeared 19567 times.

69 appeared 12608 times.

44 appeared 24217 times.

43 appeared 25539 times.

42 appeared 25751 times.

41 appeared 24431 times.

47 appeared 21387 times.

50 appeared 22509 times.

53 appeared 17491 times.

157 appeared 2889 times.

174 appeared 2204 times.

173 appeared 2355 times.

171 appeared 2431 times.

137 appeared 3674 times.

125 appeared 4109 times.

98 appeared 7192 times.

76 appeared 11016 times.

56 appeared 17739 times.

118 appeared 4866 times.

143 appeared 3392 times.

119 appeared 4768 times.

70 appeared 12817 times.

71 appeared 12937 times.

73 appeared 11727 times.

95 appeared 7835 times.

153 appeared 3024 times.

145 appeared 3308 times.

185 appeared 2271 times.

158 appeared 2735 times.

152 appeared 3116 times.

182 appeared 2212 times.

101 appeared 6705 times.

66 appeared 13018 times.

54 appeared 17857 times.

48 appeared 22653 times.

34 appeared 24120 times.

36 appeared 28432 times.

30 appeared 26365 times.

25 appeared 19625 times.

8 appeared 14349 times.

6 appeared 14360 times.

5 appeared 14945 times.

12 appeared 14720 times.

31 appeared 24545 times.

106 appeared 6188 times.

114 appeared 5278 times.

108 appeared 5814 times.

122 appeared 4565 times.

115 appeared 5130 times.

109 appeared 5605 times.

126 appeared 4205 times.

116 appeared 5001 times.

132 appeared 3875 times.

102 appeared 6389 times.

120 appeared 4672 times.

124 appeared 4342 times.

148 appeared 3450 times.

140 appeared 3570 times.

154 appeared 3140 times.

68 appeared 12643 times.

103 appeared 6527 times.

55 appeared 18096 times.

35 appeared 28459 times.

57 appeared 17336 times.

67 appeared 12656 times.

74 appeared 11079 times.

107 appeared 5928 times.

113 appeared 5280 times.

112 appeared 5473 times.

121 appeared 4662 times.

117 appeared 4977 times.

128 appeared 4131 times.

147 appeared 3272 times.

150 appeared 3413 times.

131 appeared 3844 times.

135 appeared 3799 times.

51 appeared 21500 times.

188 appeared 2164 times.

187 appeared 2331 times.

166 appeared 2630 times.

178 appeared 2237 times.

175 appeared 2291 times.

94 appeared 8243 times.

97 appeared 7465 times.

63 appeared 14550 times.

214 appeared 2081 times.

224 appeared 2423 times.

226 appeared 2375 times.

233 appeared 1978 times.

238 appeared 1743 times.

237 appeared 1802 times.

221 appeared 2394 times.

210 appeared 1937 times.

127 appeared 4355 times.

111 appeared 5409 times.

139 appeared 3485 times.

58 appeared 16422 times.

60 appeared 14608 times.

75 appeared 10905 times.

219 appeared 2290 times.

242 appeared 1246 times.

241 appeared 1385 times.

240 appeared 1470 times.

244 appeared 1033 times.

246 appeared 806 times.

245 appeared 903 times.

247 appeared 768 times.

236 appeared 1851 times.

248 appeared 612 times.

243 appeared 1203 times.

249 appeared 448 times.

239 appeared 1674 times.

83 appeared 9732 times.

59 appeared 15590 times.

72 appeared 12496 times.

142 appeared 3518 times.

133 appeared 3820 times.

134 appeared 3817 times.

151 appeared 3149 times.

192 appeared 2349 times.

156 appeared 2953 times.

167 appeared 2493 times.

165 appeared 2665 times.

186 appeared 2244 times.

170 appeared 2464 times.

155 appeared 3090 times.

161 appeared 2692 times.

160 appeared 2794 times.

136 appeared 3672 times.

138 appeared 3616 times.

110 appeared 5532 times.

169 appeared 2482 times.

216 appeared 2066 times.

228 appeared 2139 times.

235 appeared 1894 times.

231 appeared 1950 times.

218 appeared 2163 times.

205 appeared 2384 times.

208 appeared 1981 times.

199 appeared 2306 times.

162 appeared 2710 times.

168 appeared 2552 times.

163 appeared 2798 times.

172 appeared 2398 times.

184 appeared 2239 times.

183 appeared 2198 times.

189 appeared 2233 times.

213 appeared 2007 times.

225 appeared 2386 times.

229 appeared 2053 times.

232 appeared 1875 times.

217 appeared 2037 times.

202 appeared 2182 times.

149 appeared 3345 times.

164 appeared 2754 times.

250 appeared 346 times.

177 appeared 2229 times.

204 appeared 2272 times.

223 appeared 2360 times.

230 appeared 1946 times.

181 appeared 2256 times.

211 appeared 2051 times.

220 appeared 2470 times.

180 appeared 2257 times.

179 appeared 2212 times.

227 appeared 2180 times.

206 appeared 2174 times.

212 appeared 2115 times.

215 appeared 2139 times.

201 appeared 2173 times.

234 appeared 1985 times.

190 appeared 2315 times.

222 appeared 2424 times.

207 appeared 2038 times.

1 appeared 21610 times.

252 appeared 140 times.

251 appeared 241 times.

254 appeared 70 times.

253 appeared 118 times.

255 appeared 30 times.

0 appeared 13675 times.

Encoded codes with pixel values shown:

0 is 32

1000000 is 31

10000010 is 69

100000110 is 187

1000001110 is 149

1000001111 is 102

100001000 is 166

100001001 is 83

10000101 is 72

100001100 is 128

100001101 is 117

100001110 is 112

1000011110 is 116

1000011111 is 113

10001000 is 68

10001001 is 67

1000101 is 43

1000110 is 42

1000111 is 27

100100000 is 224

10010000100 is 183

10010000101 is 184

10010000110 is 200

10010000111 is 182

10010001 is 11

10010010 is 71

10010011000 is 231

10010011001 is 218

1001001101 is 192

1001001110 is 167

10010011110 is 234

10010011111 is 207

1001010 is 30

1001011 is 37

1001100000 is 169

1001100001 is 153

1001100010 is 156

10011000110 is 209

10011000111 is 208

10011001 is 3

100110100 is 101

1001101010 is 243

1001101011 is 110

10011011 is 65

10011100 is 0

10011101000 is 219

100111010010 is 236

1001110100110 is 158

10011101001110 is 246

10011101001111 is 216

1001110101 is 188

100111011 is 175

10011110 is 61

1001111100 is 89

10011111010 is 245

10011111011 is 237

100111111 is 196

1010000 is 28

1010001 is 36

1010010 is 35

1010011 is 29

10101000 is 8

10101001 is 6

10101010 is 62

10101011 is 64

10101100 is 63

10101101 is 60

10101110 is 9

1010111100 is 138

1010111101 is 136

1010111110 is 142

1010111111 is 134

10110000 is 4

10110001 is 5

1011001000 is 141

1011001001 is 140

1011001010 is 150

1011001011 is 135

10110011 is 2

101101000 is 223

10110100100 is 252

10110100101 is 254

10110100110 is 78

10110100111 is 233

10110101 is 59

1011011000 is 201

1011011001 is 109

1011011010 is 215

10110110110 is 222

10110110111 is 147

101101110 is 95

1011011110 is 227

1011011111 is 107

10111000 is 13

101110010 is 230

1011100110 is 250

10111001110 is 106

10111001111 is 99

10111010 is 58

10111011 is 17

101111000 is 86

101111001 is 97

101111010 is 185

101111011 is 7

1011111000 is 126

10111110010 is 213

10111110011 is 189

10111110100 is 163

10111110101 is 235

1011111011 is 133

1011111100 is 157

1011111101 is 125

1011111110 is 171

10111111110 is 244

101111111110 is 145

101111111111 is 146

1100000000 is 104

1100000001 is 178

110000001 is 92

11000001 is 57

110000100 is 96

110000101 is 85

110000110 is 88

1100001110 is 202

11000011110 is 177

11000011111 is 124

11000100 is 53

11000101 is 18

110001100 is 249

110001101 is 14

11000111 is 56

11001000000 is 217

11001000001 is 164

11001000010 is 232

11001000011 is 198

1100100010 is 103

1100100011 is 176

11001001 is 54

11001010 is 16

11001011 is 55

11001100 is 15

11001101000 is 181

11001101001 is 203

1100110101 is 120

11001101100 is 197

11001101101 is 199

110011011100 is 255

110011011101 is 211

11001101111 is 168

110011100 is 247

1100111010 is 94

110011101100 is 212

110011101101 is 228

110011101110 is 195

110011101111 is 205

11001111 is 19

11010 is 10

1101100000 is 221

1101100001 is 98

1101100010 is 119

1101100011 is 118

11011001 is 46

1101101000 is 127

1101101001 is 111

11011010100 is 172

11011010101 is 220

11011010110 is 165

11011010111 is 186

11011011 is 52

11011100 is 25

11011101 is 20

110111100 is 84

110111101 is 81

110111110 is 82

1101111110 is 129

1101111111 is 105

11100000 is 23

111000010 is 100

111000011 is 66

1110001000 is 229

11100010010 is 131

11100010011 is 122

1110001010 is 90

1110001011 is 210

1110001100 is 115

11100011010 is 162

11100011011 is 159

111000111 is 79

11100100 is 22

11100101 is 21

11100110 is 24

11100111 is 47

1110100000 is 91

1110100001 is 238

1110100010 is 226

11101000110 is 180

111010001110 is 132

1110100011110 is 251

1110100011111 is 193

11101001 is 51

11101010 is 1

11101011 is 45

1110110000 is 225

1110110001 is 93

111011001 is 75

1110110100 is 87

1110110101 is 239

111011011 is 77

11101110 is 34

1110111100 is 170

1110111101 is 161

111011111 is 12

111100000 is 76

111100001 is 74

1111000100 is 137

11110001010 is 114

11110001011 is 206

11110001100 is 123

11110001101 is 121

1111000111 is 214

11110010 is 50

11110011 is 26

11110100 is 48

11110101 is 49

11110110000 is 144

111101100010 is 253

111101100011 is 179

11110110010 is 190

11110110011 is 143

11110110100 is 194

11110110101 is 154

11110110110 is 204

11110110111 is 130

11110111 is 38

11111000 is 39

1111100100 is 108

11111001010 is 148

11111001011 is 191

111110011 is 73

11111010 is 40

11111011000 is 240

111110110010 is 241

111110110011 is 173

11111011010 is 174

11111011011 is 139

111110111 is 70

1111110000 is 80

11111100010 is 242

11111100011 is 248

11111100100 is 160

11111100101 is 155

11111100110 is 152

11111100111 is 151

11111101 is 44

11111110 is 33

11111111 is 41

execution time: 56788ms.

**(六) 執行結果、討論與心得**

1. 執行結果

1. Test1.pgm:[NUTN\_Algoritm/HW4\_Huffm\_Coding/1 at main · Star369666/NUTN\_Algoritm · GitHub](https://github.com/Star369666/NUTN_Algoritm/tree/main/HW4_Huffm_Coding/1)
2. Test2.pgm:[NUTN\_Algoritm/HW4\_Huffm\_Coding/2 at main · Star369666/NUTN\_Algoritm (github.com)](https://github.com/Star369666/NUTN_Algoritm/tree/main/HW4_Huffm_Coding/2)
3. Test3.pgm:[NUTN\_Algoritm/HW4\_Huffm\_Coding/3 at main · Star369666/NUTN\_Algoritm (github.com)](https://github.com/Star369666/NUTN_Algoritm/tree/main/HW4_Huffm_Coding/3)
4. Test4.pgm:[NUTN\_Algoritm/HW4\_Huffm\_Coding/4 at main · Star369666/NUTN\_Algoritm (github.com)](https://github.com/Star369666/NUTN_Algoritm/tree/main/HW4_Huffm_Coding/4)

2. 討論

1. Running Time with n\*m pixels(\_bitset = 32)

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

經過測試後，圖片像素大小與執行時間成正相關。

1. Bit\_size at line 11(#define \_bitset) with compression effection

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

經過測試後，我覺得Bit\_size設為32，可以應用在大部分的pgm圖片壓縮，且是達到最佳壓縮的效果。

1. Bit\_size at line 11(#define \_bitset) with running time

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

經過測試後，發現Bit\_size不太會影響到執行時間。

3. 心得

本次實作可以說是我在四個作業裡面查找資料並反覆修改最多的一次，也是唯一一次我求助於同學的意見。首先是建立huffman tree的時候，我發現用一個root由上面往下建立是完全錯誤的方法，只存pixel value跟frquency的vector<vector<int>>根本就沒辦法在把external node結合成一個小子樹後，又是記錄它們的root，又是可以把它們放回去min heap，所以我在這裡將min heap建構成vector<huffman\*>，這樣才能確保建立好的root插入回min heap得以保存；再來是到解壓縮的時候，我又不斷卡在如何判斷ascii code = 10或32是真的數字10還是換行或空格，最後冷靜下來拋開一切想法，只思考換行跟空格出現在哪，判斷式才呼之欲出，根本不需要考慮其他案例；成功解壓縮後，看到壓縮檔居然比原圖還要胖，就來到最艱辛的地方，因為一開始用數字去轉huffman code的結果是”0101”之類的，但是它們都是一個個字元，直接寫進file會變得很胖，所以又在網上不斷搜尋用二進制寫入檔案的方法，得到file.write()跟file.read()後，還得把輸入的字串在最尾端補0到是8的倍數，才可以放進bitset<8>，然後以二進制寫入file，讀出二進制編碼後，根據已儲存的資料判斷有沒有補0，有的話還得把那些0丟棄，才是正確的字串，而且這是我第一次把一大坨的01全都丟在一個字串中，我還特地去查字串的大小上限，得到答案是只要還有記憶體就可以不斷增加，最終好不容易使得二進制寫入讀出了，然後解壓縮的圖片也是對的，結果對於test1跟test4而言，這個演算法壓縮效果也不好，僅僅是百位數個byte而已，但是對於test2跟test3甚至是倍數增胖，根據同學所言，只要用二進制寫入，就一定會讓.hc檔變小，所以我只好再去查資料，最終把bitset<8>調整到bitset<32>才算是大功告成，至於為什麼在字串都有好好補齊的情況下，理論上來說沒有任何一個bitset的空間被浪費，結果還是造成壓縮檔大小的差距如此巨大，這個我就無從得知了。總之，無論是使用二進制寫入、讀出檔案，複習min heap、quick sort，實作huffman tree，又或者是拿到檔案的大小，都是一次很好的學習體驗。

**參考文獻**

1. <https://www.adobe.com/tw/creativecloud/file-types/image/raster/pgm-file.html>
2. <https://www.shubo.io/binary-heap/>
3. <https://stackoverflow.com/questions/15054442/how-to-read-write-ascii-pgm-file>
4. h[ttps://shengyu7697.github.io/std-vector/](https://shengyu7697.github.io/std-vector/)
5. <https://stackoverflow.com/questions/20731/how-do-you-clear-a-stringstream-variable>
6. <https://home.gamer.com.tw/creationDetail.php?sn=4114818>
7. <https://blog.csdn.net/jzwong/article/details/45022589>
8. <https://www.delftstack.com/zh-tw/howto/cpp/how-to-determine-if-a-string-is-number-cpp/>
9. <http://squall.cs.ntou.edu.tw/cpp/102spring/lab04/ifstream.eof.html>
10. <https://www.ascii-code.com/>
11. <https://cplusplus.com/reference/algorithm/lower_bound/>
12. <https://stackoverflow.com/questions/15843525/how-do-you-insert-the-value-in-a-sorted-vector>
13. <https://stackoverflow.com/questions/51159287/function-that-returns-an-unordered-map>
14. <https://shengyu7697.github.io/cpp-write-binary-file/>
15. <https://cplusplus.com/forum/beginner/106521/>
16. <https://www.twblogs.net/a/5c91f169bd9eee35fc15aa21>
17. [filesize - 如何在 C++ 中獲取檔的大小？- 堆疊溢出 (stackoverflow.com)](https://stackoverflow.com/questions/5840148/how-can-i-get-a-files-size-in-c)