

# #Assignment 3

通訊四 108503303 馬寧

## ● Huffman\_Coding

### 1. 編譯結果

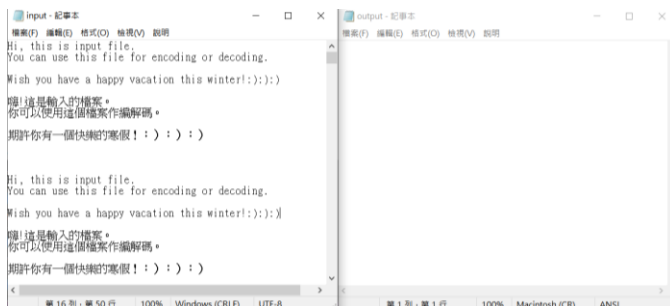
```
PS D:\文件\大四上\資料結構\Arithmetic_coding\assignment_3\src> cd D:\文件\大四上\資料結構\Huffman_coding\assignment_3\src
PS D:\文件\大四上\資料結構\Huffman_coding\assignment_3\src> gcc -o main main.c huffman.c utils.c
PS D:\文件\大四上\資料結構\Huffman_coding\assignment_3\src> ./main
```

### 2. 執行結果

#### Test\_1(編碼)

在 Huffman\_coding\assignment\_3\src 文件夾中增加 3 個 txt 檔案，分別為 input.txt、output.txt、decode.txt，其中 input.txt 檔案中有資料內容。

| 大四上 > 資料結構 > Huffman_coding > assignment_3 > src |                    |                |       |  |
|--|--------------------|----------------|-------|--|
| 名稱   | 修改日期               | 類型             | 大小    |  |
| decode   | 2023/1/11 下午 05:29 | 文字文件           | 0 KB  |  |
| encode   | 2023/1/11 下午 05:22 | 文字文件           | 11 KB |  |
| huffman  | 2023/1/11 下午 05:28 | C 來源檔案         | 8 KB  |  |
| huffman  | 2014/6/19 下午 06:19 | C Header File  | 2 KB  |  |
| input  | 2023/1/11 下午 05:16 | 文字文件           | 14 KB |  |
| main   | 2023/1/11 下午 04:46 | C 來源檔案         | 2 KB  |  |
| main   | 2023/1/11 下午 05:28 | 應用程式           | 49 KB |  |
| output   | 2023/1/11 下午 05:43 | 文字文件           | 0 KB  |  |
| README   | 2023/1/11 下午 04:19 | Markdown 來源... | 1 KB  |  |
| utils  | 2014/6/19 下午 06:19 | C 來源檔案         | 3 KB  |  |
| utils  | 2014/6/19 下午 06:19 | C Header File  | 2 KB  |  |



選擇 mode 為 0 (encoding)

輸入欲編碼檔案的檔案名字

輸入編碼後儲存資料的檔案名字

輸出編碼所花費的時間

```
PS D:\文件\大四上\資料結構\Huffman_coding\assignment_3\src> ./main
Please input the mode [0(encoding)/1(decoding)/2(quick encoding)]: 0
Please input the input_file name: input.txt
Please input the output_file name: output.txt
Input file: input.txt
Output file: output.txt
Mode: encoding

time cost: 0.033000
Input successfully processed.
```

## ● Arithmetic\_Coding

### 1. 編譯結果

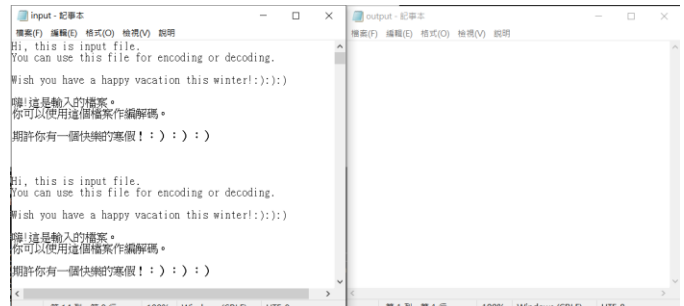
```
PS D:\文件\大四上\資料結構\Huffman_coding\assignment_3\src> cd D:\文件\大四上\資料結構\Arithmetic_coding\assignment_3\src
PS D:\文件\大四上\資料結構\Arithmetic_coding\assignment_3\src> gcc -o main main.c arth.c bitstream.c
PS D:\文件\大四上\資料結構\Arithmetic_coding\assignment_3\src> ./main
```

### 2. 執行結果

#### Test\_1(編碼)

在 Arithmetic\_coding\assignment\_3\src 文件夾中增加 3 個 txt 檔案，分別為 input.txt、output.txt、decode.txt，其中 input.txt 檔案中有資料內容。

| 大四上 > 資料結構 > Arithmetic_coding > assignment_3 > src |                     |               |           |  |
|---|---------------------|---------------|-----------|--|
| 名稱  | 修改日期                | 類型            | 大小        |  |
| arth  | 2023/1/11 下午 03:55  | C 來源檔案        | 3 KB      |  |
| arth  | 2001/12/20 下午 10:35 | C Header File | 1 KB      |  |
| bitstream   | 2023/1/11 下午 03:52  | C 來源檔案        | 1 KB      |  |
| bitstream   | 2001/12/20 下午 04:56 | C Header File | 1 KB      |  |
| decode  | 2023/1/11 下午 05:20  | 文字文件          | 33,844 KB |  |
| encode  | 2023/1/11 下午 05:18  | 文字文件          | 11 KB     |  |
| input   | 2023/1/11 下午 05:16  | 文字文件          | 14 KB     |  |
| main  | 2023/1/11 下午 05:06  | C 來源檔案        | 2 KB      |  |
| main  | 2023/1/11 下午 05:51  | 應用程式          | 46 KB     |  |
| output  | 2023/1/11 下午 05:20  | 文字文件          | 0 KB      |  |



選擇 mode 為 e (encoding)

輸入欲編碼檔案的檔案名字

輸入編碼後儲存資料的檔案名字

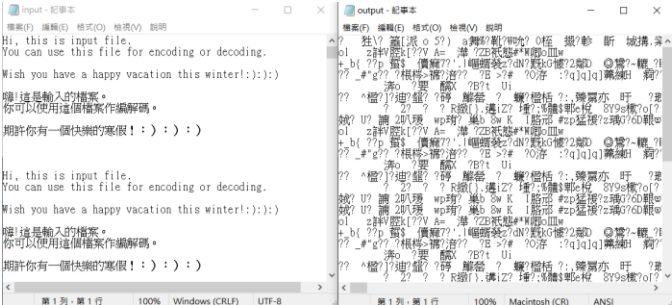
輸出編碼所花費的時間

```
PS D:\文件\大四上\資料結構\Arithmetic_coding\assignment_3\src> ./main
Please input the mode [e(encoding)/d(decoding)]: e
Please input the input_file name: input.txt
Please input the output_file name: output.txt
Input file: input.txt
Output file: output.txt
Mode: encoding

start encoding...
time cost: 0.006000
```

可觀察到 output.txt 的檔案大小增加了，內容為編碼後的資料，但其檔案大小會少於 input.txt

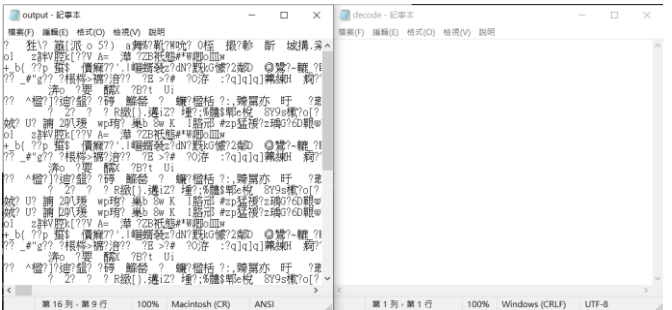
| 名稱      | 修改日期               | 類型             | 大小    |
|---------|--------------------|----------------|-------|
| decode  | 2023/1/11 下午 05:29 | 文字文件           | 0 KB  |
| encode  | 2023/1/11 下午 05:22 | 文字文件           | 11 KB |
| huffman | 2023/1/11 下午 05:28 | C 來源檔案         | 8 KB  |
| huffman | 2014/6/19 下午 06:19 | C Header File  | 2 KB  |
| input   | 2023/1/11 下午 05:16 | 文字文件           | 14 KB |
| main    | 2023/1/11 下午 04:46 | C 來源檔案         | 2 KB  |
| main    | 2023/1/11 下午 05:28 | 應用程式           | 49 KB |
| output  | 2023/1/11 下午 05:47 | 文字文件           | 10 KB |
| README  | 2023/1/11 下午 04:19 | Markdown 來源... | 1 KB  |
| utils   | 2014/6/19 下午 06:19 | C 來源檔案         | 3 KB  |
| utils   | 2014/6/19 下午 06:19 | C Header File  | 2 KB  |



## Test\_2(解碼)

將剛剛編碼後的資料解碼，並將解碼後的資料儲存於 decode.txt，decode.txt 初始的檔案大小為 0

| 名稱      | 修改日期               | 類型             | 大小    |
|---------|--------------------|----------------|-------|
| decode  | 2023/1/11 下午 05:50 | 文字文件           | 0 KB  |
| huffman | 2023/1/11 下午 05:28 | C 來源檔案         | 8 KB  |
| huffman | 2014/6/19 下午 06:19 | C Header File  | 2 KB  |
| input   | 2023/1/11 下午 05:16 | 文字文件           | 14 KB |
| main    | 2023/1/11 下午 04:46 | C 來源檔案         | 2 KB  |
| main    | 2023/1/11 下午 06:09 | 應用程式           | 49 KB |
| output  | 2023/1/11 下午 06:10 | 文字文件           | 10 KB |
| README  | 2023/1/11 下午 04:19 | Markdown 來源... | 1 KB  |
| utils   | 2014/6/19 下午 06:19 | C 來源檔案         | 3 KB  |
| utils   | 2014/6/19 下午 06:19 | C Header File  | 2 KB  |

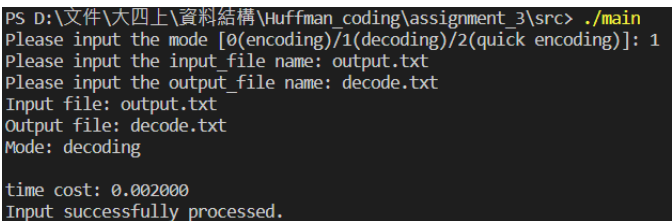


選擇 mode 為 1 (decoding)

輸入欲解碼檔案的檔案名字

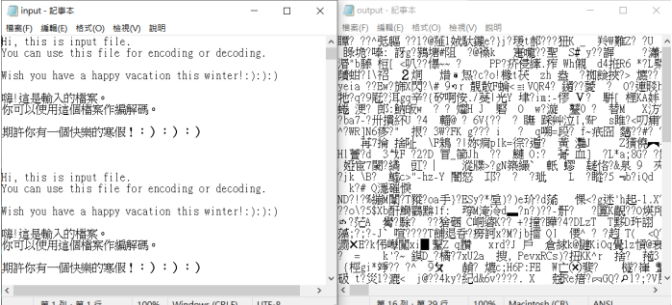
輸入解碼後儲存資料的檔案名字

輸出解碼所花費的時間



可觀察到 output.txt 的檔案大小增加了，內容為編碼後的資料，但其檔案大小會少於 input.txt

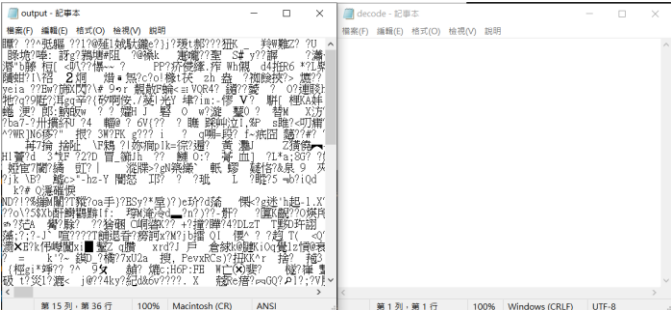
| 名稱         | 修改日期                | 類型            | 大小    |
|------------|---------------------|---------------|-------|
| arithmetic | 2023/1/11 下午 03:55  | C 來源檔案        | 3 KB  |
| arithmetic | 2001/12/20 下午 10:35 | C Header File | 1 KB  |
| bitstream  | 2023/1/11 下午 03:52  | C 來源檔案        | 1 KB  |
| bitstream  | 2001/12/20 下午 04:56 | C Header File | 1 KB  |
| decode     | 2023/1/11 下午 05:53  | 文字文件          | 0 KB  |
| encode     | 2023/1/11 下午 05:18  | 文字文件          | 11 KB |
| input      | 2023/1/11 下午 05:16  | 文字文件          | 14 KB |
| main       | 2023/1/11 下午 05:06  | C 來源檔案        | 2 KB  |
| main       | 2023/1/11 下午 05:51  | 應用程式          | 46 KB |
| output     | 2023/1/11 下午 05:54  | 文字文件          | 10 KB |



## Test\_2(解碼)

將剛剛編碼後的資料解碼，並將解碼後的資料儲存於 decode.txt，decode.txt 初始的檔案大小為 0

| 名稱         | 修改日期                | 類型            | 大小    |
|------------|---------------------|---------------|-------|
| arithmetic | 2023/1/11 下午 03:55  | C 來源檔案        | 3 KB  |
| arithmetic | 2001/12/20 下午 10:35 | C Header File | 1 KB  |
| bitstream  | 2023/1/11 下午 03:52  | C 來源檔案        | 1 KB  |
| bitstream  | 2001/12/20 下午 04:56 | C Header File | 1 KB  |
| decode     | 2023/1/11 下午 06:04  | 文字文件          | 0 KB  |
| input      | 2023/1/11 下午 05:16  | 文字文件          | 14 KB |
| main       | 2023/1/11 下午 05:06  | C 來源檔案        | 2 KB  |
| main       | 2023/1/11 下午 05:51  | 應用程式          | 46 KB |
| output     | 2023/1/11 下午 06:01  | 文字文件          | 10 KB |

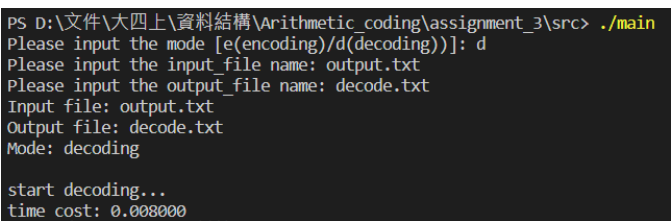


選擇 mode 為 d (decoding)

輸入欲解碼檔案的檔案名字

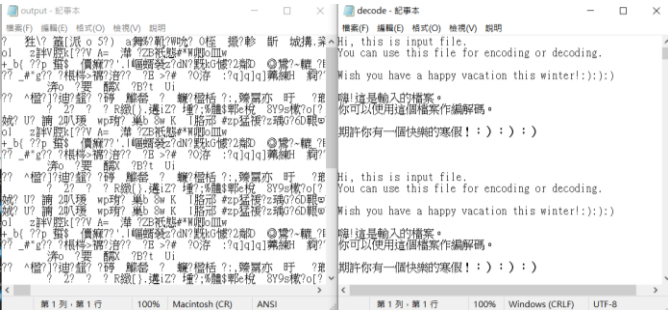
輸入解碼後儲存資料的檔案名字

輸出解碼所花費的時間



可發現 decode.txt 與 input.txt 的檔案大小及內容相同

| 大四上 > 資料結構 > Huffman_coding > assignment_3 > src |                    |               |       |  | 搜尋 src |
|--|--------------------|---------------|-------|--|--------|
| 名稱   | 修改日期               | 類型            | 大小    |  |        |
| decode   | 2023/1/11 下午 06:11 | 文字文件          | 14 KB |  |        |
| huffman  | 2023/1/11 下午 05:28 | C 來源檔案        | 8 KB  |  |        |
| huffman  | 2014/6/19 下午 06:19 | C Header File | 2 KB  |  |        |
| input  | 2023/1/11 下午 05:16 | 文字文件          | 14 KB |  |        |
| main   | 2023/1/11 下午 04:46 | C 來源檔案        | 2 KB  |  |        |
| main   | 2023/1/11 下午 06:09 | 應用程式          | 49 KB |  |        |
| output   | 2023/1/11 下午 06:10 | 文字文件          | 10 KB |  |        |
| README   | 2023/1/11 下午 04:19 | Markdown 來源   | 1 KB  |  |        |
| utils  | 2014/6/19 下午 06:19 | C 來源檔案        | 3 KB  |  |        |
| utils  | 2014/6/19 下午 06:19 | C Header File | 2 KB  |  |        |



### 3. 分析 編碼

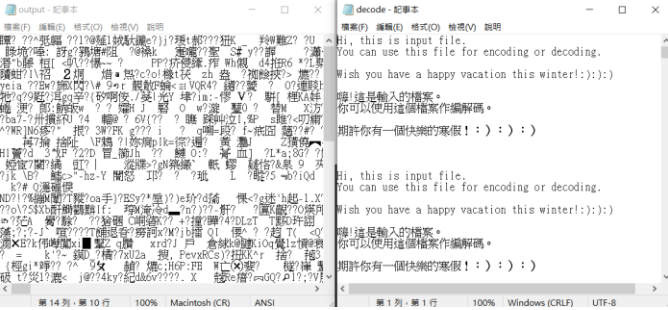
```
void encode(FILE *infile, FILE *outfile)
{
    char *curCode;
    int len, i;
    Tree *tree;

    fseek(infile, 0L, SEEK_END);
    len = ftell(infile);
    rewind(infile);
    tree = generateHuffmanTree(infile, len);
    rewind(infile);
    writeHeader(outfile, tree, len);
    for (i = 0; i < len; i++)
    {
        curCode = getCode(tree, (unsigned char) fgetc(infile));
        writeCode(outfile, convertCode(curCode), strlen(curCode));
    }
    writeCode(outfile, 0, 8 - getCount());
    destroyTree(tree);
}
```

由於 Huffman coding 在編碼時除了計算每個字元出現的次數亦須建立一個 HuffmanTree 導致在編碼時需花費較多的時間

可發現 decode.txt 與 input.txt 的檔案大小及內容相同

| 大四上 > 資料結構 > Arithmetic_coding > assignment_3 > src |                     |               |       |  | 搜尋 src |
|---|---------------------|---------------|-------|--|--------|
| 名稱  | 修改日期                | 類型            | 大小    |  |        |
| arth  | 2023/1/11 下午 03:55  | C 來源檔案        | 3 KB  |  |        |
| arth  | 2001/12/20 下午 10:35 | C Header File | 1 KB  |  |        |
| bitstream   | 2023/1/11 下午 03:52  | C 來源檔案        | 1 KB  |  |        |
| bitstream   | 2001/12/20 下午 04:56 | C Header File | 1 KB  |  |        |
| decode  | 2023/1/11 下午 06:06  | 文字文件          | 14 KB |  |        |
| input   | 2023/1/11 下午 05:16  | 文字文件          | 14 KB |  |        |
| main  | 2023/1/11 下午 05:06  | C 來源檔案        | 2 KB  |  |        |
| main  | 2023/1/11 下午 05:51  | 應用程式          | 46 KB |  |        |
| output  | 2023/1/11 下午 06:01  | 文字文件          | 10 KB |  |        |



### 3. 分析 編碼

```
void encode(int code)
{
    long range;
    int symbol;

    if(code<0) symbol = EOF_symbol;
    else symbol = c2i[code];

    range = (long) (high - low) + 1;
    high = low + (range*cum[symbol-1])/cum[0] - 1;
    low = low + (range*cum[symbol])/cum[0];

    while(1)
    {
        if(high<Half)
        {
            putbit0();
        }
        else if (low>=Half)
        {
            putbit1();
            low -= Half;
            high -= Half;
        }
        else if (low >= First_qtr && high<Third_qtr)
        {
            refine += 1;
            low -= First_qtr;
            high -= First_qtr;
        }
        else break;

        low <<= 1;
        high = (high<<1) + 1;
    }

    update_freq(symbol);
}
```

## 解碼

```
/* Decoding functions implementations */  
  
void decode(FILE *infile, FILE *outfile)  
{  
    int i, ch, length = 0;  
    Tree *pTree;  
  
    readHeader(infile, &pTree, &length);  
    setCount(0);  
    for (i = 0; i < length; i++)  
    {  
        ch = decodeByte(infile, pTree);  
        fputc(ch, outfile);  
    }  
    destroyTree(pTree);  
}
```

由於此次測試資料內容為一整串相同的文字重複出現，Huffman coding 所建立的 Huffman Tree 較廣而不深，解碼時速度較 Arithmetic coding 快

Arithmetic coding 在編碼時根據字元出現的機率及機率分布進行編碼，且編碼過程中不斷計算字串機率分布的區間，字元出現機率會受到之前出現的字元影響，機率分布區間亦隨之影響，直到整個字串演算編碼完成，算法較 Huffman 複雜，適用於出現機率平均的資料內容。

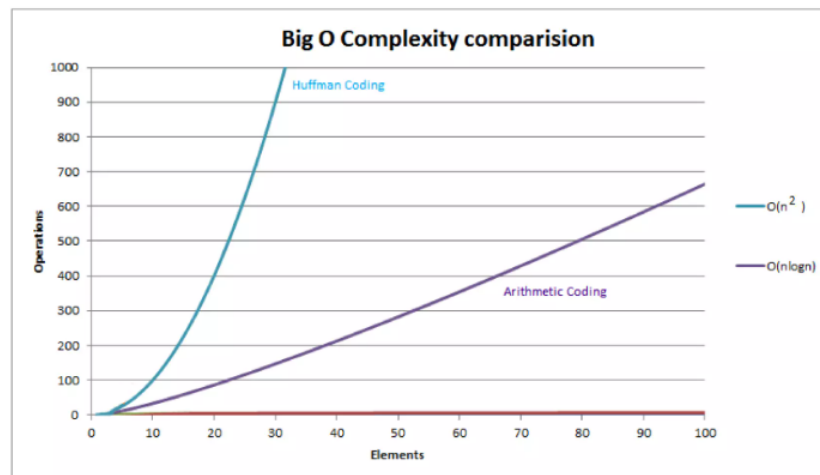
因此次測試資料內容為一整串相同的文字重複出現，每個字元出現機率平均，故 Arithmetic coding 的編碼速度較快

## 解碼

```
int decode(void)  
{  
    static bool first_time = 1;  
    int range;  
    int curr_cum, symbol;  
  
    if(first_time)  
    {  
        for(range=1; range<=Code_value_bits; range++)  
        {  
            value = (value<<1) + getbit();  
        }  
        first_time = 0;  
    }  
  
    range = (high - low) + 1;  
    curr_cum = (((value-low)+1)* int curr_cum e;  
    for(symbol = 1; cum[symbol]>curr_cum; symbol++);  
    high = low + (range*cum[symbol-1])/cum[0]-1;  
    low = low + (range*cum[symbol])/cum[0];  
  
    while(1)  
    {  
        if(high<Half)  
        {  
            value -- Half;  
            low -- Half;  
            high -- Half;  
        }  
        else if (low>=Half)  
        {  
            value -- First_qtr;  
            low -- First_qtr;  
            high -- First_qtr;  
        }  
        else if(low>=First_qtr && high<Third_qtr)  
        {  
            value -- First_qtr;  
            low -- First_qtr;  
            high -- First_qtr;  
        }  
        else break;  
        low <<= 1;  
        high = (high<<1)+1;  
        value = (value<<1)+getbit();  
    }  
  
    if(symbol==EOF_symbol) return -1;  
    update_freq(symbol);  
    return i2c[symbol];  
}
```

解碼與編碼時的速度差不多，但解碼較 Huffman coding 慢

## Complexity comparison



Arithmetic coding 的時間複雜度較 Huffman coding 佳

Huffman coding

Running time Function :  $T(n) = N[\log n + \log(2n-1)] + S_n$

Complexity :  $O(N^2)$

Arithmetic coding

Running time Function :  $T(n) = N[\log n + a] + S_n$

$N \Rightarrow$  Number of input symbols

$n \Rightarrow$  current number of unique symbols

$S \Rightarrow$  Time to maintain internal data structure

Complexity :  $O(N \log n)$

資料來源：<https://www.slideshare.net/ramakantsoni/performance-analysis-of-huffman-and-arithmetic-coding-compression>