JPEG Decoder

40647007s 朱健愷

[編寫在 Github]

編譯:

```
cd cmake-build-debug
make
```

這會在 cmake-build-debug/資料夾生成主程式 JPEG-Codec

使用方式:

```
JPEG-Codec -i [input file name] (-o output file name)
```

或

```
JPEG-Codec [input file name]
```

解碼出來的檔案會放在與 input file 同一個目錄下

環境:

Testing

- CPU: Intel Core i7-8750H CPU
- Operating System: Windows 10 64-bit
- Memory: 32GB

Build

- GCC Version: 6.3.0
- CMake Version: 3.16.5

額外加速:

原本的 idct:

$$\text{result[i][j]} = \frac{1}{4}\Sigma_{x=0}^{7}\Sigma_{y=0}^{7}C_xC_y\cos\left(\frac{(2i+1)x\pi}{16}\right)\cos\left(\frac{(2j+1)y\pi}{16}\right)table[x][y]$$

其中的 C 項定義為

$$C_0 = \frac{1}{\sqrt{2}}$$

$$C_i = 1 \quad for \quad 1 \le i \le 7$$

將原 idct 最內層的 x 項提取出來成為

$$result[i][j] = \frac{1}{4}\Sigma_{x=0}^{7} C_x \cos\left(\frac{(2i+1)x\pi}{16}\right)\Sigma_{y=0}^{7} C_y \cos\left(\frac{(2j+1)y\pi}{16}\right) table[x][y]$$

，可以觀察出在$j = 0 \ldots 7$時，$\Sigma_{y=0}^{7} C_y \cos\left(\frac{(2j+1)y\pi}{16}\right) table[x][y]$一直被重複計算，而且在$i = 0 \ldots 7$時會一直用到上述重複計算的值，因此可以將$\Sigma_{y=0}^{7} C_y \cos\left(\frac{(2j+1)y\pi}{16}\right) table[x][y]$視為一個以 j 與 x 為變數的函數，並做出以 j 與 x 為變數的快取，即可不必再重新算一遍重複的項。亦即把 IDCT 從$O(N^4)$優化到$O(N^3)$，不過因為即使如此還是要花兩次時間計算(一次是計算快取，另一次是利用快取計算出 idct)，但還是省了$8/2 \quad = 4$倍時間。

解 Zigzag 原本是用原本的 zigzag 表將舊的表換到新的在複製回舊的，後來用另外一個程式(在/resource 裡面的 zigzag_swap)計算出新的 zigzag table 讓解 Zigzag 可以用 in place swap 的方式轉回未 zigzag 前的表。

在範式霍夫曼編碼的部分，因為是把整個霍夫曼樹的每一層擠到右邊去，因此出現兩個性質:

1. 葉節點的編碼值可以從同一層前一個葉節點+1 取得
2. 每一層最左邊的葉節點可以從上一層最開頭的葉節點+上一層有的編碼數在乘以 2 得到

利用這兩個性質我們可以預先計算每一層最開頭葉節點的編碼值，在後續讀取 dc 或 ac 值，要解碼範式霍夫曼編碼值時可以利用編碼值減去該層最開頭葉節點的編碼值取得解碼的 offset，再從該層的解碼 table 加上解碼 offset 即可取得解碼值。