信息论第一次大作业实验报告

19373124 傅云濠

必做模组

分1MB进行分块压缩

使用方法见`README.md`

选做模组

缓冲大小为36kb

使用方法见`README.md`

实践问题

1 重复性的文件结构

```
root@DESKTOP-2AKSPCD:/mnt/c/Users/FYHSSGSS/Documents/University/2021spring/ITAC/tarz# ./tarz -l 1 lab1_data/testfile1  lab1_data/output
type 1
The file size is 65536bytes

The file size is 23055bytes

Bye!
root@DESKTOP-2AKSPCD:/mnt/c/Users/FYHSSGSS/Documents/University/2021spring/ITAC/tarz# ./tarz -g 1 lab1_data/testfile1  lab1_data/output
The file size is 65536bytes

The output file is 66821bytes
Bye!
```

可发现，当使用哈夫曼的时候，压缩率为$\frac{66821}{65536}=100.02\%$，甚至会出现越压缩越大的情况，而当我们使用LZ78的时候，压缩率为$\frac{23055}{65536}=35.18\%$

分析原因，是因为文件为256 字节 0x00，256 字节 0x01，256 字节 0x02，...，256 字节 0xff，256种字节出现的频率完全一样，若建出一棵哈夫曼树是一棵完全二叉树，每个节点的节点深度完全一样，哈夫曼编码也一样长，再加上我压缩程序前面表头会提前写入每个字符的词频，导致压缩结果大于之前，十分失败。

而对于LZ78编码而言，是针对新建trie树进行编码，若全出现相同字符c的次数为n，则会建出$\sqrt{n}$个节点，外加在写入时需要表上节点编号以及节点编号的字节数，导致最后压缩的结果为35%,尚可。

```
root@DESKTOP-2AKSPCD:/mnt/c/Users/FYHSSGSS/Documents/University/2021spring/ITAC/tarz# ./tarz -l 1 lab1_data/testfile1  lab1_data/output
type 1
The file size is 65536bytes

The file size is 23055bytes

Bye!
```

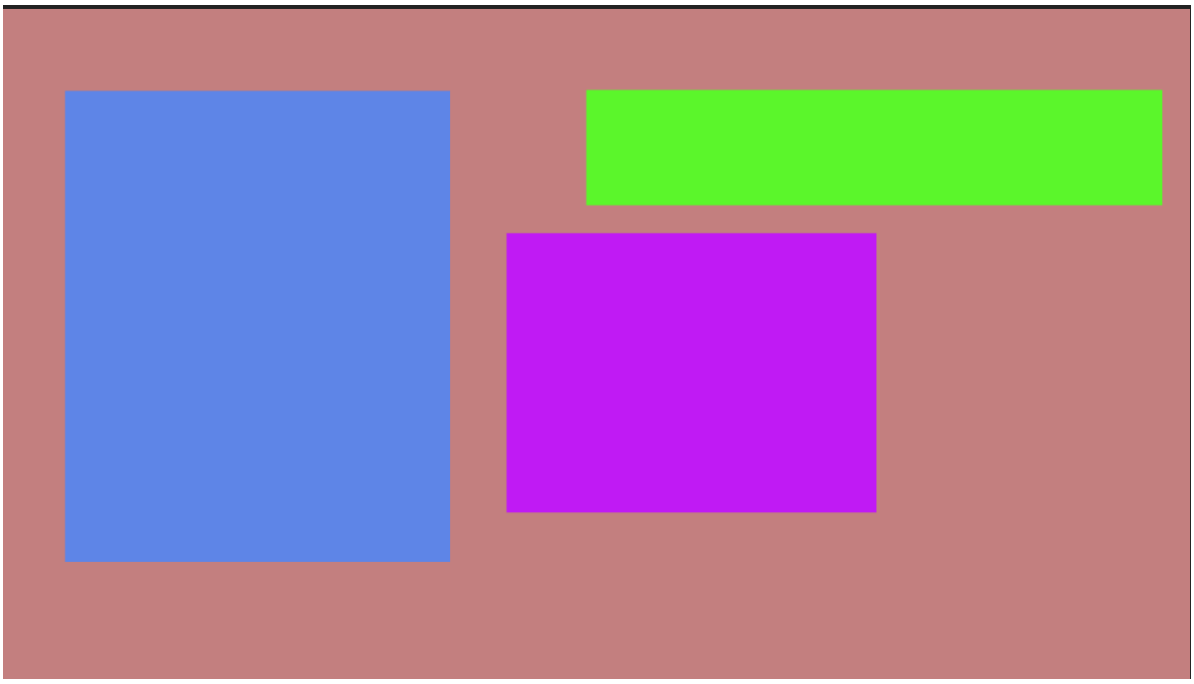
```
root@DESKTOP-2AKSPCD:/mnt/c/Users/FYHSSGSS/Documents/University/2021spring/ITAC/tarz# ./tarz -g 1 lab1_data/output  lab1_data/output2
The file size is 23055bytes

The output file is 19873bytes
Bye!
```

若我们对testfile1先进行lz78压缩，再进行哈夫曼压缩，最终压缩率为$\frac{19873}{65536}=30.32\%$

2. 不同格式的压缩

bmp

bmp图片如下（这个图片是直接管同学要的）



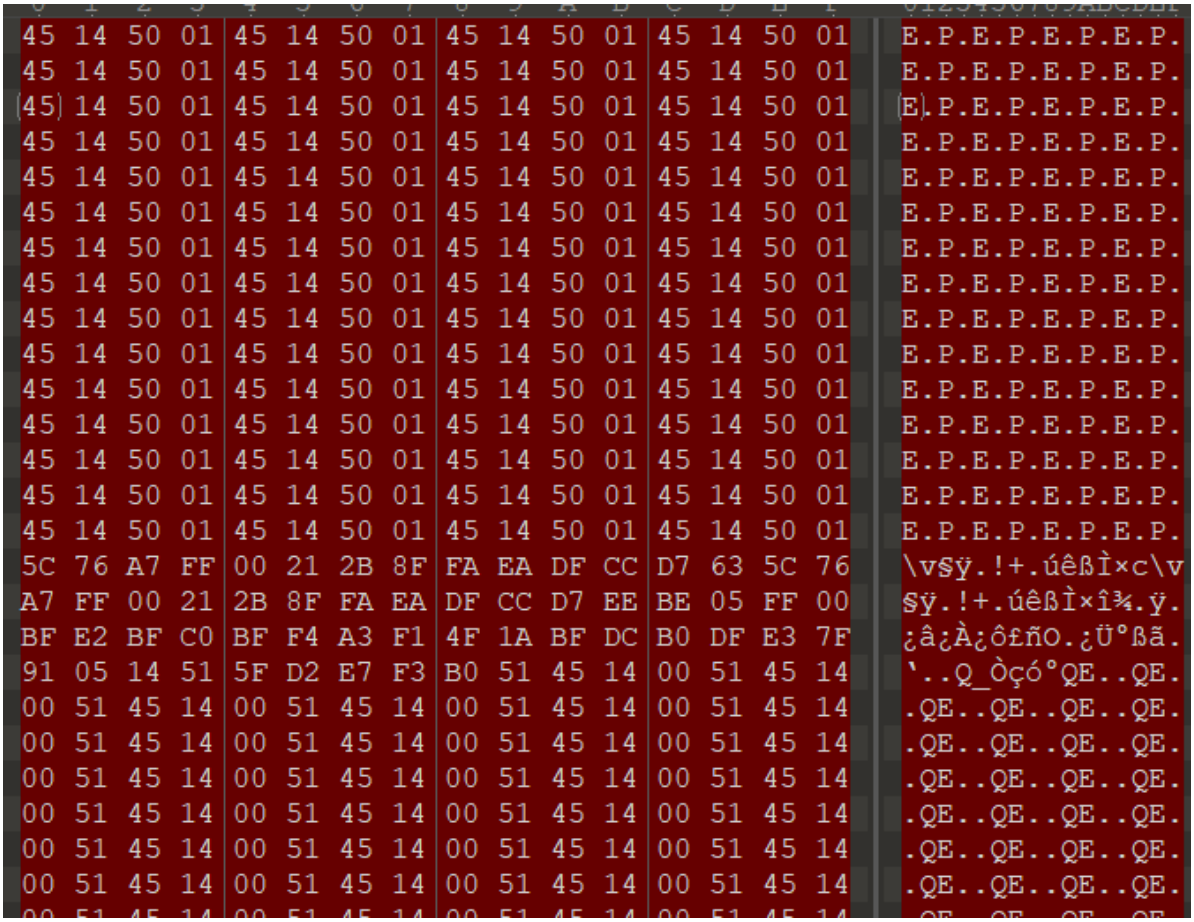
使用哈夫曼编码压缩bmp文件，压缩率$\frac{7277111}{24883338}=29.24\%$

```
root@DESKTOP-2AKSPCD:/mnt/c/Users/FYHSSGSS/Documents/University/2021spring/ITAC/tarz# ./tarz -g 1 lab1_data/draw.bmp  la
b1_data/output
The file size is 24883338bytes

```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
The output file is 7277111bytes
```

使用LZ78编码压缩bmp文件，压缩率$\frac{363841}{24883338}=1.46\%$
```

```
root@DESKTOP-2AKSPCD:/mnt/c/Users/FYHSSGSS/Documents/University/2021spring/ITAC/tarz# ./tarz -l 1 lab1_data/draw.bmp  la
b1_data/output
type 1
The file size is 24883338bytes

```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
```waiting```
The file size is 363841bytes
```

分析LZ编码为何在压缩位图方面如此优秀？

```
7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F  .Ã..Ã..Ã..Ã..Ã..
C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3  Ã..Ã..Ã..Ã..Ã..Ã
7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F  ..Ã..Ã..Ã..Ã..Ã.
7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F  .Ã..Ã..Ã..Ã..Ã..
C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3  Ã..Ã..Ã..Ã..Ã..Ã
7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F  ..Ã..Ã..Ã..Ã..Ã.
7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F  .Ã..Ã..Ã..Ã..Ã..
C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3  Ã..Ã..Ã..Ã..Ã..Ã
7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F  ..Ã..Ã..Ã..Ã..Ã.
7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F  .Ã..Ã..Ã..Ã..Ã..
C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3  Ã..Ã..Ã..Ã..Ã..Ã
7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F  ..Ã..Ã..Ã..Ã..Ã.
7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F  .Ã..Ã..Ã..Ã..Ã..
C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3  Ã..Ã..Ã..Ã..Ã..Ã
7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F  ..Ã..Ã..Ã..Ã..Ã.
7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F  .Ã..Ã..Ã..Ã..Ã..
C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3  Ã..Ã..Ã..Ã..Ã..Ã
7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F  ..Ã..Ã..Ã..Ã..Ã.
7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F  .Ã..Ã..Ã..Ã..Ã..
C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3  Ã..Ã..Ã..Ã..Ã..Ã
7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F 7F C3 7F  ..Ã..Ã..Ã..Ã..Ã.
```

分析位图的16进制，发现出现大量相同字符相间出现，设想一下巨大的trie树仅有两种字符交替出现，每次新开辟叶节点的深度为1，3，5，7，9，...,是$\sqrt{n}$级别的。故能压缩的十分优秀。

jpg

另存为.jpg格式，大小为176337字节。

使用哈夫曼编码压缩jpg文件，压缩率$\frac{117857}{176337}=66.84\%$

```
root@DESKTOP-2AKSPCD:/mnt/c/Users/FYHSSGSS/Documents/University/2021spring/ITAC/tarz# ./tarz -g 1 lab1_data/draw.jpg  la
b1_data/output
The file size is 176337bytes

The output file is 117857bytes
Bye!
```

使用LZ78编码压缩jpg文件，压缩率$\frac{56899}{176337}=32.27\%$
```



分析原因，jpg相较与bmp有很大压缩，也会出现像bmp一样很多的相同字符相间出现，但是这样的块很多，每块大小不是很大，导致LZ78的编码优势没有很好的体现出来。

exe

分别对exe进行Huf和lz78的压缩,压缩率分别为 $\frac{1488466}{2212731}=67.27\%, \frac{1412261}{2212731}=63.82\%$

```
ultron@DESKTOP-2AKSPCD:/mnt/c/Users/FYHSSGSS/Documents/University/2021spring/ITAC/tarz$ ./tarz -g 1 哈夫曼压缩.exe outpu
t
The file size is 2212731bytes

```waiting```
```waiting```
The output file is 1488466bytes
Bye!
ultron@DESKTOP-2AKSPCD:/mnt/c/Users/FYHSSGSS/Documents/University/2021spring/ITAC/tarz$ ./tarz -l 1 哈夫曼压缩.exe outpu
t
type 1
The file size is 2212731bytes

```waiting```
```waiting```
The file size is 1412261bytes

Bye!
```

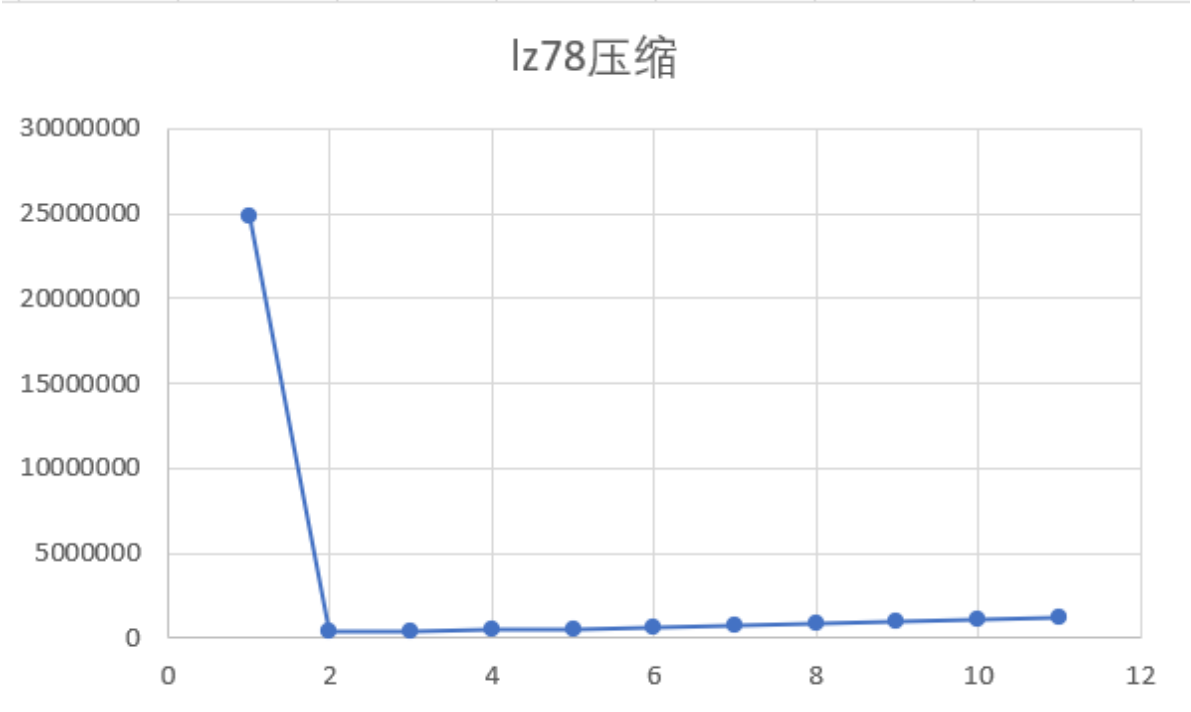
3. 黑洞!

压缩的是lab1_data/draw.bmp文件

lz编码

使用lz编码压缩10次，直接展示中间数据：

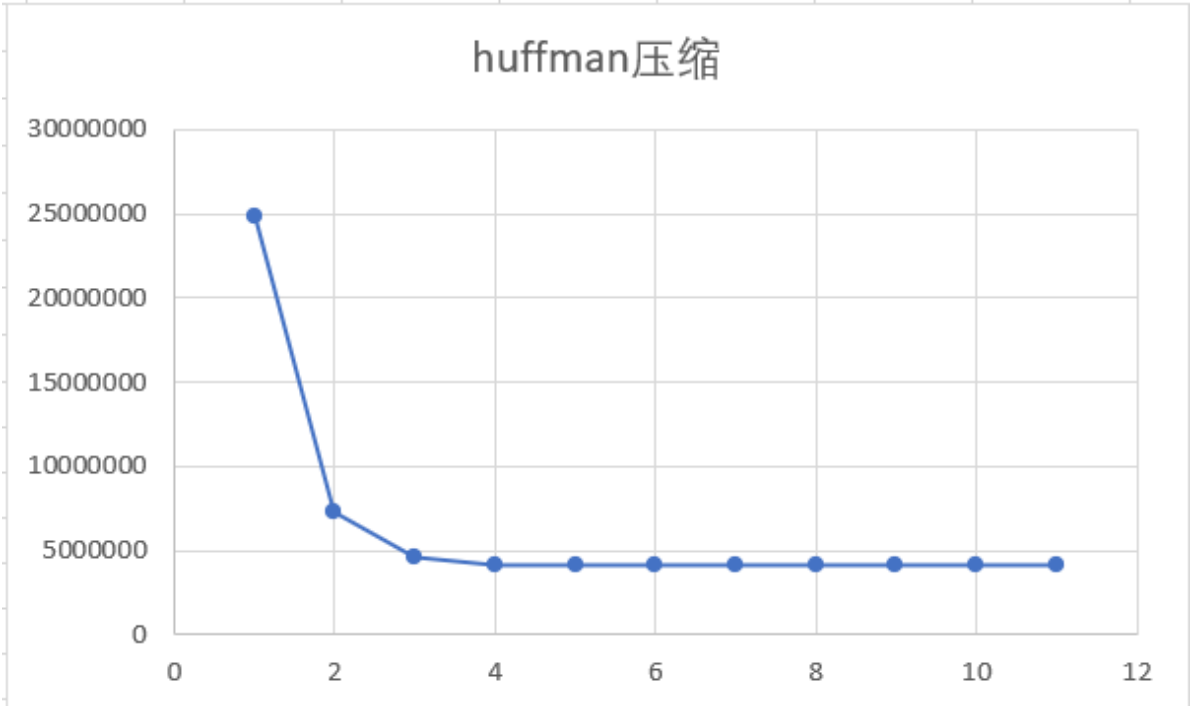
次数/times	字节数/bytes	压缩率
0	24883338	
1	363841	0.014621872676407
2	405348	1.11408
3	472451	1.165544
4	543765	1.150945
5	621741	1.1434
6	708447	1.1394567834516303,
7	805710	1.13729
8	919388	1.14109
9	1055528	1.14109
10	1221279	1.157031362502937



哈夫曼编码

使用哈夫曼编码压缩10次，中间数据为：

次数/times	字节数/bytes	压缩率
0	24883338	
1	7277111	0.2924491481006286
2	4574389	0.6285995912388859
3	4184077	0.9146745062564641
4	4160617	0.9943930286177812
5	4163328	1.000651586050819
6	4168425	1.0012242609758346,
7	4173565	1.0012330796403917
8	4178705	1.0012315610275628
9	4183845	1.0012300461506616
10	4188985	1.0012285349959187



首先越压越小是绝对不可能的，由于香农第一定理，平均码长B和信息量$H(X)$之间的关系为$\frac{B}{k}=H(x)$(K趋近无穷）,所以一定存在一个严格的下界是无法逾越的。之所以会出现后期越压越大，我猜测是因为当一个文件进行了一次（huf/lz78）的编码压缩后，剩余的编码再对于(huf/lz78)就一定不是最优编码了，字节的频率会发生改变，使得越压缩越大。