信息论与编码——信源编码实验指导

1 概要

本次实验的目的是让同学们熟悉信源编码的不同方法，将其应用于实际程序，从而更好地理解信源编码；此外，也锻炼一下同学们的编程能力。

本次实验的满分为 15 分。考虑到同学们的编程基础与空闲时间有所差别，因此实验提供了两个必做模块和若干个选做模块，且各模块的分数总和远高于实验的满分，因此你可以在完成必做模块的基础上，根据自己的能力和时间完成相应的选做模块，从而获得更多的分数，**当你实际获得的总分大于满分 15 分时，会记为 15 分**。

你可以使用你喜欢的语言实现功能，选择的语言不会影响你的分数。对核心功能以外的部分，你可以自由使用不同的库/包。

你可以参考不同来源的各种实现，但请务必确保代码由你自己编写。

我们会编译并运行你的代码，请确保你测试运行了你的代码。

2 计分规则

实验总计 15 分，由以下两部分相加得到：

（1）必做模块的完成：必做模块的满分为 10 分，你需要尽可能完成模块中的全部要求（当你在必做模块中的得分低于 5 分时，你将无法获得整个实验的满分，因为选做模块最多只能得到 10 分，反之同理）。

（2）选做模块的完成：选做模块包括 5 个实践问题，完成每个选做模块都能获得对应的分数，你可以根据自己的实际情况进行选择。选做模块的满分同样为 10 分，当你实际获得的分数大于 10 分时，会记为 10 分。

2.1 必做模块

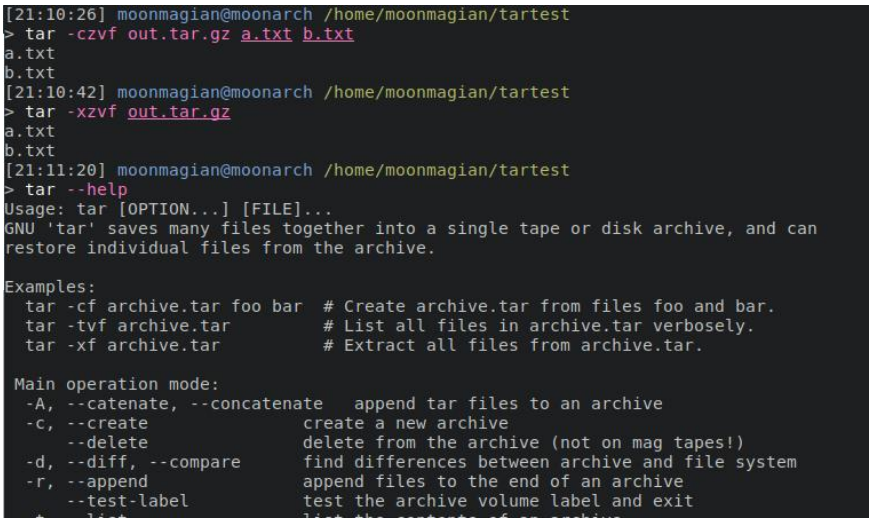
必做模块的分数细则如下：

模块	要求	分数
必做模块 1	Huffman 编码的功能实现	3 分
必做模块 2	LZ 系列编码的功能实现	3 分
	LZ 与 Huffman 编码结合的功能实现	1 分
共同要求	代码鲁棒性与安全性	1 分
	代码风格与用户交互	1 分
	实验报告的内容完整性	1 分

（1）功能实现：你的程序应该能够对任意文件进行编码，并对正确的编码文件进行解码，**你应该确保编码再解码文件后，其 SHA256 值与原文件相同，并在实验报告中展示这一点**。

（2）代码鲁棒性与安全性：你的程序应该考虑到一些边界情况和错误，例如对空文件的处理、对仅含有一种字节的文件的处理（在 Huffman 编码中需要特别考虑）、对解码输入错误的处理（欲解码的文件并非由该程序编码）等。对安全性和鲁棒性进行全盘考量是相当困难的，即使强如 OPENSSL 也有许多严重的安全漏洞，因此**你只需要按自己的理解尽可能使代码鲁棒，并且在实验报告中说明所做的工作**，便可以拿到这部分分数。

（3）代码风格与用户交互：代码风格决定了代码的可维护性，**你的代码应该有清楚的变量命名、恰当的注释和合理的结构设计**。此外，缺少用户交互的终端应用并不完整，因此这部分分数也来源于**设计一个可用的用户交互并在实验报告中对其进行展示**。我们不鼓励你设计 GUI（当然 GUI 也能拿到分数），而是鼓励你设计 CLI 交互，一个来自 TAR 的良好示例如图。



当然，TAR 是一个十分复杂而强大的工具，此处仅作为举例。**在实验过程中你只需要设计一个可用的、完整的用户交互接口便能拿到这部分分数**。根据语言的不同，你可以使用一些工具辅助你设计，这里举几个常见的库/包（当然，还有许多不同的选择）：

语言	库/包
Python	argparse（Built-in 包）
C	getopt（位于 unistd.h）
C++	Boost.Program_options （https://www.boost.org/doc/libs/1_63_0/doc/html/program_options.html）
Java	Commons CLI（http://commons.apache.org/proper/commons-cli/）
Rust	clap（https://github.com/clap-rs/clap）

2.2 选做模块

在选做模块中，你需要调用必做模块中的程序或编写新的程序来处理一些实际任务，并回答相关的问题。**正确地回答实践问题或完成代码实现**即可拿到相应模块的全部分数。即使是部分解答或实现，也能拿到部分分数，因此你应该**在代码中保留你的部分工作并在实验报告中说明**。

选做模块的分数细则如下：

模块	要求	分数
选做模块 1	回答有关“重复性文件结构”的实践问题	2 分
选做模块 2	回答有关“不同格式压缩”的实践问题	2 分
选做模块 3	回答有关“反复压缩”的实践问题	2 分
选做模块 4	算术编码的功能实现	6 分
选做模块 5	流压缩与解压缩方案的设计和实现	8 分

3 必做模块

3.1 必做模块 1——Huffman 编码

Huffman 编码是课程中重点学习的编码方式之一，在必做模块 1 中，你需要使用二元 Huffman 编码，以文件的每一个 byte 作为信源发出的一个符号，对其进行编码和解码。

你的解码操作应该仅依赖于编码后的文件，而不能依赖于内存中的任何数据，即不能将编码时在内存中构建的码树直接用于解码，而需要以恰当的方式将码树一起存储在编码后的文件中，并在解码时将其复原出来。

在实验报告中，你需要给出以下内容：

（1）运行源代码所需的环境依赖；

（2）展示你的程序对至少三种不同类型文件的编码、解码结果，包括源文件和解码后文件 SHA256 值的比较结果，以及你的用户交互设计；

（3）说明你是如何将码树存储在编码文件中并在解码时将其复原的；

（4）简述你的代码进行了哪些安全性和鲁棒性方面的考虑。

3.2 必做模块 2——LZ 系列编码

LZ77 与 LZ78 编码相比 Huffman 编码、Shannon-Fano 编码在实际的压缩、解压场景具有更为广泛的应用。在必做模块 2 中，你需要选择 LZ77 编码或 LZ78 编码实现其中之一。之后，你需要综合你选择的 LZ 系列编码与 Huffman 编码两种方式，实现对文件的编码与解码（编码时先使用 LZ 编码，再使用 Huffman 编码，解码时先使用 Huffman 解码，再使用 LZ 解码）。同样的，你的解码操作应该仅依赖于编码后的文件，而不依赖于内存中的数据。

在实验报告中，你需要给出以下内容：

（1）运行源代码所需的环境依赖；

（2）分别展示仅用 LZ 系列编码和结合 LZ、Huffman 两种编码，对文件进行编码、解码的结果（此处只要展示一种类型的文件即可，同样需要包含对 SHA256 值的比较和对用户交互设计的展示）。

（3）说明你所用的 LZ 系列编码的基本原理；

（4）简述你的代码进行了哪些安全性和鲁棒性方面的考虑。

4 选做模块

4.1 选做模块 1——重复性的文件结构（2 分）

分别使用你在必做模块中实现的 Huffman 编码和 LZ 系列编码，对文件 lab1_data/testfile 进行编码，lab1_data/testfile 的文件内容是：256 字节 0x00，256 字节 0x01，256 字节 0x02，...，256 字节 0xff，总计 64KB。

对于两种编码方式，分别观察文件编码前后的大小变化，并解释为什么会发生这种变化。

4.2 选做模块 2——不同格式的压缩（2 分）

使用画图或者其他工具进行一些简单的艺术创作（推荐使用三四种颜色，不要太多，尽量使用较大的分辨率(3840x2160)），将图片保存为 BMP 和 JPEG 两种格式，并独立使用你在必做模块中实现的 Huffman 编码与 LZ 系列编码，对两种格式的图片进行编码。

分别观察两种格式的图片在编码前后的大小变化，发现并解释其中存在的差异。

类似的，可以尝试编码一个可执行文件，解释文件体积的变化。更进一步，可以选择继续探究不同格式文件（.txt，.mp4，.avi，.zip 等），或者不同内容（中文文本和英文文本）的压缩效果。

4.3 选做模块 3——反复压缩（2 分）

“如果我将一个超大的文件压缩几百次，他就会变得越来越小，最后达到几 KB，这便是时间换空间!”这种说法正确吗？使用你在必做模块中实现的 Huffman 编码与 LZ 系列编码，对 BMP 格式的图片迭代编码 10 次。

观察每次编码前后文件大小的变化，并解释为什么会发生这种变化。

4.4 选做模块 4——算术编码（6 分）

算术编码有着不少的优秀性质，但它的计算机实现并非那么容易。在此选做模块中，你需要使用算术编码（或者一些实现变种，例如区间编码）来实现对文件的编码与解码。同样的，你的解码操作应该仅依赖于编码后的文件，而不依赖于内存中的数据。

作为选做模块，你可以主要聚焦于实现编码和解码的过程，此模块对鲁棒性和用户交互不做要求，**你在报告只需要中给出编码和解码函数的基本原理，以及对一种类型的文件进行编码和解码，展示其 SHA256 值的比较结果即可**。

4.5 选做模块 5——流压缩与解压缩（8 分）

设想一个情景：你的硬盘还剩 10GB 空间，而你想要立即下载一份 12GB 的数据保存起来（不会立即使用，仅仅是保存），你的硬盘中全是不能删除的重要文件，你的内存也只有 1GB，怎么办？

一个可行的方法是去借一块 16GB 的 U 盘，将数据下载到 U 盘后使用信源编码将其压缩，再转存到硬盘里。

设想另一个场景：你从服务器接收 100GB 的压缩文本，其中只有几行含有关键的字符串，需要从中找到这一行。你的硬盘只剩 1MB 可用，你的内存也相当有限，怎么办？

这个情景似乎比上一个情景更让人头大。

好在，这种情景在上世纪 90 年代过于常见，以至于操作系统都使用了管道和输入输出流来解决它。

想像一个未经过滤的水源，我们希望得到饮用水，一种做法是将所有水积蓄到一个大的蓄水池，使用不同的设施处理蓄水池中的水，一次性得到所有饮用水。

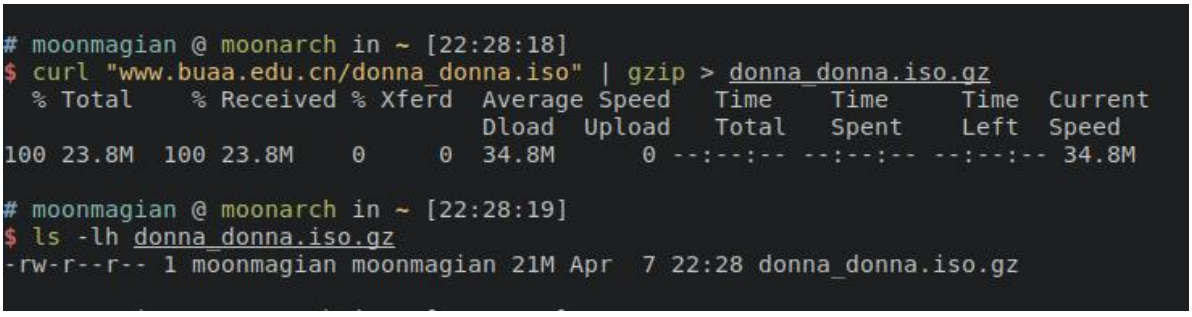
比起这种方式，一个更有效率的方式是从水源建立一个管道，一次只让数量不多的水流过各个设施，得到一部分饮用水，使用这种方式，我们不再需要建立一个极其庞大的蓄水池，而仅仅需要一些管道。

现在，将数据比喻成水流，程序比喻成各个设施，我们用一组管道连接各个程序，将上个程序的标准输出连接到下个程序的标准输入，之前的两个场景似乎可以解决了。

对第一个情景，我们连接这样一组程序：

下载程序（将下载的数据输出到标准输出流）→压缩程序→文件。

我们不借助额外的硬盘空间，在下载时便将数据压缩到文件。如图展示了使用 curl 和 gzip 建立一个这样的流（仅供演示使用）。



对第二个情景，我们连接这样一组程序：

下载程序→解压缩程序→按行检索数据的程序→输出。

我们一边下载一边检索信息，从而不再需要预先完全下载文件。如图展示了使用 curl、gzip 和 grep 建立一个这样的流（仅供演示使用）。

```
# moonmagian @ moonarch in /usr/share/nginx/html [22:35:33]
$ curl "www.buaa.edu.cn/never_gonna_give_you_up.txt.gz" | gunzip | grep -yn "never gonna"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   444  100   444    0     0   433k      0 --:--:-- --:--:-- --:--:--  433k
14:Never gonna give you up
15:Never gonna let you down
16:Never gonna run around and desert you
17:Never gonna make you cry
18:Never gonna say goodbye
19:Never gonna tell a lie and hurt you
31:Never gonna give you up
32:Never gonna let you down
33:Never gonna run around and desert you
34:Never gonna make you cry
35:Never gonna say goodbye
36:Never gonna tell a lie and hurt you
37:Never gonna give you up
38:Never gonna let you down
39:Never gonna run around and desert you
40:Never gonna make you cry
41:Never gonna say goodbye
42:Never gonna tell a lie and hurt you
48:Never gonna give, never gonna give (Give you up)
50:Never gonna give, never gonna give (Give you up)
62:Never gonna give you up
63:Never gonna let you down
64:Never gonna run around and desert you
65:Never gonna make you cry
66:Never gonna say goodbye
67:Never gonna tell a lie and hurt you
68:Never gonna give you up
69:Never gonna let you down
70:Never gonna run around and desert you
71:Never gonna make you cry
72:Never gonna say goodbye
73:Never gonna tell a lie and hurt you
74:Never gonna give you up
75:Never gonna let you down
76:Never gonna run around and desert you
77:Never gonna make you cry
78:Never gonna say goodbye
79:Never gonna tell a lie and hurt you
```

你的任务是修改你在必做模块中完成的程序（请做好备份），使其能接受标准输入，将结果输出到标准输出。

输出到标准输出是十分简单的，但是，处理输入并不那么容易。

对于标准输入：你无法通过 fseek 一类的函数重读以前的数据了（可以认为，从标准输入读入的数据被“消耗”了）；此外，你也无法用 ftell 和 fseek 拿到文件大小了（你会得到 0 或者-1）；你也不应该将标准输入读到一个超大的缓冲区再操作（这失去了使用流的意义）。

因此，你需要设立一个大小恰当的缓冲区（gzip 压缩时默认使用 36K 的输入缓冲和 8K 的输出缓冲，在这个实验你可以不考虑用于提高性能的输出缓冲），每当数据填满缓冲区，便单独地将当前块编码、解码，你还应该特别考虑流的总长度不是缓冲大小的整数倍的情况。

为了测试你的程序，在终端或 cmd 下分别输入：

```
your_program_in_encode_mode < some_file > some_file.enc
your_program_in_decode_mode < some_file.enc > some_file.dec
```

其中 some_file 是已经存在的一个文件。

之后，为了验证程序的正确性，对 macOS 或 Linux，在终端输入：

```
Openssl dgst -sha256 some_file some_file.dec
```

对 Windows，在 cmd 输入：

```
certUtil -hash filesome_file SHA256
certUtil -hash filesome_file.dec SHA256
```

两个文件的 SHA256 值应该相同。

5 实验提交

你需要提交包含实验报告、程序源代码和过程文件（用于测试你的程序的编码后、解码后的文件）的压缩包。

请使用 7z、rar 等格式，或在 utf-8locale 下使用 tar.gz，不要使用 zip。

实验报告并无特定的格式，但**需要转换成 pdf**。本指导书在 2-4 章中给出了各实验模块在实验报告中所需撰写的必要内容（选做模块 5 除外，该模块可以根据你的完成度自由发挥），只要实验报告包含所有必要的内容和解释，它便不会影响你的分数，字数多的实验报告不会给你带来任何加分。

你需要将压缩包命名为“2023 信息论与编码-实验 1-学号-姓名”，例如“2023 信息论与编码-实验 1-19371111-神秘人.7z”，你可以运行 lab1_data/check.py 来检查你提交的文件名是否合乎规范。在截止日期前，你可以无限次地修改并重新上传压缩包，只要保持文件名一样，便可覆盖上传，请不要覆盖别人（例如你的室友）的作业。

在确认无误后，将你的压缩包上传至：

https://bhpan.buaa.edu.cn:443/link/5B82251C6382AE74298D118181803848

你应该在北京时间 2023 年 4 月 30 日 23:59 前完成上述的提交。