МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М. В. ЛОМОНОСОВА
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ
КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Айрапетьянц Каринэ Арсеновна

Использование оптимальных кодов для сжатия данных: ZIPmeHuffman

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Москва, 2022

Оглавление

1. Вспомогательные бибилиотеки

Для корректной работы программы необходимы следующие пакеты:

```
collections
argparse
time
sys
pickle
```

Они содержатся в файле requirements.txt и при необходимости могут быть установлены с помощью следующей команды:

```
    pip install -r /path/to/requirements.txt
```

2. Использование программы:

```
python3 huffman.py [-h] --r {1,2} --f FILENAME

Welcome to ZipMeHuffman!

optional arguments:
  -h, --help    show this help message and exit
  --r {1,2}     1 - compress, 2-decompress
  --f FILENAME  path to file to compress/decompress
```

Поддерживается два режима работы, при этом выводится дополнительная информация:

- 1 - сжатие:

```
        python3 huffman.py --r 1 --f tocompress.txt
        compressing file tocompress.txt...
        original size (bites):  794
        size after compress (raw, bites):  401
        time in seconds:  0.0027205944061279297
```

На выходе формируется файл с названием исходного файла и расширением .zmh

- 2 - разжатие:

```
        python3 huffman.py --r 2 --f tocompress.zmh
        decompressing file tocompress.zmh...
        time in seconds:  0.0034983158111572266
```

На выходе формируется файл с названием исходного файла и добавкой _dec в конце в формате исходного файла.

3. Описание алгоритма

Был реализован класс Huffman с двумя основными методами:

```
Huffman.encode(self, enc_file_name)
Huffman.decode(self, dec_file_name)
```

3.1. Формат .zmh

Структура файла формата .zmh следующая:

1) Длина закодированного сообщения
2) Расширение файла до сжатия
3) Длина словаря
4) Словарь (pickle dumps)

3.2. Сжатие

Файл с названием enc_file_name считывается в режиме rb, затем считается частота встречания каждого байта и записывается в словарь, где ключ - байт, значение - частота (целое число). Далее, создаются узлы с помощью вспомогательного класса Node и помещаются в список, где сортируются по убыванию частоты. По алгоритму Хаффмана строится дерево: берутся два элемента с наименьшими частотами и „склеиваются“, новый элемент добавляется обратно в список, а два старых удаляются и становятся „детьми“ нового элемента.

По полученному дереву строится таблица: движение происходит от корня дерева вниз, до листьев и собирается код.

Затем производится кодирование: второй проход по входному файлу и замена байтов на соответствующие построенные коды. Полученная последовательность разбивается на куски по 8 бит и, так как последний байт может быть неполным, в выходной файл (формата .zmh) записывается длина закодированного сообщения для последующего корректного декодирования. Также записывается расширение исходного файла. Для более компактной записи словаря используется модуль pickle, поэтому для дальнейшего декодирования записывается и длина словаря (после применения pickle.dumps). В самом конце записывается закодированное сообщение.

3.3. Разжатие

На вход подается файл с расширением .zmh. Далее, первой строкой считывается длина закодированного сообщения, вотрой строкой - расширение. Затем последующие 4 байта отведены на длину словаря (len_dic), с 4 по len_dic считывается словарь, а остальное - закодированное сообщение.

При считывании кода байты преобразуются в двоичное представление и добиваются ведущими нулями до полных 8 бит, а последний байт обрабатывается отдельно - здесь количество нулей для добивки составляет: $code\_l - (code\_l//8) * 8$, где code_l - длина закодированного сообщения.

По считанному словарь восстанавливается дерево следующим образом: берется код (в данной реализации по порядку хранения в словаре) и посимвольно перебирается. Если стретился 0, то добавляется левый потомок (при отсутствии такового) и идет спуск к этому потомку. Аналогично и для 1, только здесь добавление и спуск к правому потомку. При просмотре следующего кода из словаря, обход дерева снова начинается с корня.

Теперь, имея закодированное сообщение в двоичном виде и построенное дерево, производится раскодирование с помощью посимвольного прохода по сообщению и одновременно по дереву до листьев. На листьях расположены искомые закодированные символы. После достижения листьев просмотр возобновляется с корня. Раскодированное сообщение записывается в результирующий массив. К имени файла добавляется окончание _dec и расширение.

4. Анализ

Время сжатия/разжатия указано в секундах.

Файл	Исх.	ZipMeHuffman			ZIP			7z		
	р-р	р-р сж.	вр. сж.	вр. раз.	р-р сж.	вр. сж.	вр. раз.	р-р сж.	вр. сж.	вр. раз.
txt	76,9kB	40,2kB	0.07	0.11	21,9kB	0.035	0.015	19,4kB	0.015	0.014
txt	333,3kB	172,9kB	0.11	0.26	97,2kB	0.12	0.022	80,2kB	0.15	0.027
pdf	5,4MB	5,4MB	2.04	16.02	5,3MB	0.27	0.15	5,2MB	0.6	0.29
pdf	81,1kB	84,4kB *	0.1	0.14	78,6kB	0.013	0.007	79,0kB	0.028	0.028
csv	486,3kB	231,1kB	0.15	0.34	222,3kB	0.14	0.021	181,3kB	0.19	0.055
csv	1,5MB	852,4kB	0.4	1.2	723,5kB	0.13	0.04	638,5kB	0.24	0.057
jpg	6,3MB	6,3MB	2.6	19.4	6,3MB	0.32	0.16	6,2MB	0.7	0.4
png	12,9kB	16,3kB*	0.04	0.01	12,9kB	0.005	0.002	12,9kB	0.018	0.012

ZMH сжимает текстовые данные в 2 раза с учетом накладных расходов, в то время как ZIP и 7z сжимают в 4 раза и делают это быстрее, чем ZMH (предположение: в ZIP и 7z уже встроена и используется таблица с оптимальными кодами для часто встречающихся алфавитов?). В случае с pdf файлами у ZMH сжатия не происходит либо доваляются накладные расходы (*- Было 81,1kB, стало 84,4kB, но до сжатия pdf файл имел размер 81100 бит, после чистый размер без словаря и т.д. составил 81033. Таким образом, из-за накладных расходов размер файла увеличился). CSV-файлы сжимаются в 2 раза с помощью ZMH и с чуть лучшим успехом у ZIP и 7z.

JPEG, PNG - файлы не сжимаются (так как JPEG и PNG итак сжаты в первом случае одноименным алгоритмом сжатия, а во втором - алгоритмом deflate-сжатия, который является улучшенной версией алгоритма сжатия Lempel-Ziv (LZ77), который используется в ZIP- и GZIP-файлах),снова размер файла может увеличиться из-за накладных расходов. (* - png было 12917 бит, стало 12906 „чистых“ бит).

Вывод

Данная реализация ZMH сжимает данные, но работает хуже и намного медленнее (декодирование) существующих решений (здесь декодирование осуществляется проходом по закодированному

сообщению и дереву, а в ZIP, например, происходит декодирование на основе поисковой таблицы,что быстрее). Также, накладные расходы нужно записывать еще компактнее, чем реализовано мной (с помощью pickle).