Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Дискретный анализ»
Тема: «Алгоритм LZW»

Студент: А. В. Тимофеев
Преподаватель: С. А. Сорокин
Группа: М8О-207Б-19

Дата:
Оценка:
Подпись:

Москва, 2021

Алгоритм LZW

Задача: Ваша программа должна читать входные данные из стандартного потока ввода и выводить ответ на стандартный поток вывода. Вам будут даны входные файлы двух типов.

Первый тип:

compress

<text>

Текст состоит только из малых латинских букв. В ответ на него вам нужно вывести коды, которыми будет закодирован данный текст.

Второй тип:

decompress

<codes>

Вам даны коды в которые был сжат текст из малых латинских букв, вам нужно его разжать.

Начальный словарь состоит выглядит следующим образом:

a -> 0

b -> 1

c -> 2

...

x -> 23

y -> 24

z -> 25

EOF -> 26

1 Описание

Процесс сжатия выглядит следующим образом: последовательно считываются символы входного потока и происходит проверка, существует ли в созданной таблице строк такая строка. Если такая строка существует, считывается следующий символ, а если строка не существует, в поток заносится код для предыдущей найденной строки, строка заносится в таблицу, а поиск начинается снова.

Например, если сжимают байтовые данные (текст), то строк в таблице окажется 256 (от "0"до "255"). Если используется 10-битный код, то под коды для строк остаются значения в диапазоне от 256 до 1023. Новые строки формируют таблицу последовательно, т. е. можно считать индекс строки ее кодом.

Для декодирования на вход подается только закодированный текст, поскольку алгоритм LZW может воссоздать соответствующую таблицу преобразования непосредственно по закодированному тексту. Алгоритм генерирует однозначно декодируемый код за счет того, что каждый раз, когда генерируется новый код, новая строка добавляется в таблицу строк. LZW постоянно проверяет, является ли строка уже известной, и, если так, выводит существующий код без генерации нового. Таким образом, каждая строка будет храниться в единственном экземпляре и иметь свой уникальный номер. Следовательно, при декодировании во время получения нового кода генерируется новая строка, а при получении уже известного, строка извлекается из словаря.

Кодирование

Начало.

Шаг 1. Все возможные символы заносятся в словарь. Во входную фразу X заносится первый символ сообщения.

Шаг 2. Считать очередной символ Y из сообщения.

Шаг 3. Если Y — это символ конца сообщения, то выдать код для X, иначе:

Если фраза XY уже имеется в словаре, то присвоить входной фразе значение XY и перейти к Шагу 2 ,

Иначе выдать код для входной фразы X, добавить XY в словарь и присвоить входной фразе значение Y. Перейти к Шагу 2.

Конец.

Декодирование

Начало.

Шаг 1. Все возможные символы заносятся в словарь. Во входную фразу X заносится первый код декодируемого сообщения.

Шаг 2. Считать очередной код Y из сообщения.
Шаг 3. Если Y — это конец сообщения, то выдать символ, соответствующий коду X, иначе:

Если фразы под кодом XY нет в словаре, вывести фразу, соответствующую коду X, а фразу с кодом XY занести в словарь.

Иначе присвоить входной фразе код XY и перейти к Шагу 2 .

Конец.

2 Исходный код

includes.h

```
#ifndef INCLUDES_H
#define INCLUDES_H

#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <vector>
#include <bitset>
#include <unordered_map>
#include <limits>

#include <cstdio>
#include <cstring>
#include <experimental/filesystem>

#include <ctime>

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>

#endif

}
```

lzw.h

В этом файле описан класс LZW и его методы по кодированию и декодированию. Данные методы работают на основе алгоритма описанного выше.

```

#include "includes.h"
#include <string>
#include <unordered_map>

using id = unsigned short int;
const id MAX_UNSIGNED_SHORT = std::numeric_limits<id>::max();

class LZW{

private:

    std::unordered_map<std::string, id> codeDict;
    std::unordered_map<id, std::string> decodeDict;
```

```
15      id eof = 26; // end of file
16      id next_code;
17      id max_code;
18
19  public:
20      LZW(){
21          max_code = MAX_UNSIGNED_SHORT;
22          next_code = eof + 1;
23      }
24      ~LZW(){
25
26      }
27      int code(){
28
29          std::string str;
30          std::string tmpStr;
31          this->initDict();
32          char letter;
33
34         while(std::cin >> letter){
35              tmpStr = str;
36              str += letter;
37
38              if (this->codeDict.find(str) == this->codeDict.end()){
39                  this->codeDict.insert(std::make_pair(str, this->next_code));
40                  ++(this->next_code);
41                  str = letter;
42                  std::cout.write((char*)&(this->codeDict.find(tmpStr)->second), sizeof (
                        this->codeDict.find(tmpStr)->second));
43                  tmpStr.clear();
44
45                  if (this->next_code == this->max_code){
46                      this->initDict();
47                      str.clear();
48                  }
49              }
50          }
51          if(!str.empty()){
52              std::cout.write((char*)&(this->codeDict.find(str)->second), sizeof this->
                    codeDict.find(str)->second);
53          }
54         return 0;
55      }
56
57      int decode(){
58          std::string str;
59          std::string tmpStr;
60          this->initDict();
61          id num = 0;
```

```
62          id tmpNum = 0;
63
64          if (!std::cin.read((char *)&tmpNum, sizeof(id))){
65              std::cout << "Empty file." << std::endl;
66              return 0;
67          }
68          std::cout << this->decodeDict.find(tmpNum)->second;
69           while(std::cin.read((char *)&num, sizeof(id))){
70              tmpStr = this->decodeDict.find(tmpNum)->second;
71              if(this->decodeDict.size() == num ){
72                 //std::cerr << "if(this->decodeDict.size() == num ){ " << std::endl;
73                  tmpStr += tmpStr.front();
74                  this->decodeDict.insert(std::make_pair(this->next_code, tmpStr));
75                  ++(this->next_code);
76                  str = this->decodeDict.find(num)->second;
77                  std::cout << str;
78              }else if((this->decodeDict.size() < num)&&(num != 0)){
79                      std::cerr << "ERROR: id does not exist in the dictionary." << std::
                            endl;
80                      break;
81              }else{
82                  str = this->decodeDict.find(num)->second;
83                  std::cout << str;
84                  tmpStr += str.front();
85                  this->decodeDict.insert(std::make_pair(this->next_code, tmpStr));
86                  ++(this->next_code);
87              }
88              if(this->next_code == this->max_code - 1){
89                  if (!std::cin.read((char *)&tmpNum, sizeof(id))){
90                      break;
91                  }
92                  this->initDict();
93                  if (this->decodeDict.find(tmpNum) != this->decodeDict.end()){
94                      std::cout << this->decodeDict.find(tmpNum)->second;
95                      continue;
96                  }else{
97                      std::cerr << "Dictionary initialization error." << std::endl;
98                  }
99              }
100             tmpNum = num;
101         }
102         return 0;
103     }
104     void initDict(){
105         next_code = eof + 1;
106         codeDict.clear();
107         decodeDict.clear();
108
109         for (id ch = 'a'; ch <= 'z'; ch++) {
```

```
110             std::string str(1,static_cast<char>(ch));
111             codeDict.insert(std::make_pair(str, ch - 'a'));
112             decodeDict.insert(std::make_pair(ch - 'a', str));
113         }
114         codeDict.insert(std::make_pair("", eof));
115         decodeDict.insert(std::make_pair(eof, ""));
116         }
117 };
```

main.cpp

В данном файле содержится код, отвечающий за замену дескрипторов стандартного ввода и вывода на дескрипторы файлов входного и выходного соответственно, также происходит вызов функции сжатия и декодирование файла. Предполагалось еще два case в структуре switch, в которых происходил бы ввод со стандартного потока и вывод в стандартный поток вывода, но в дальнейшем я отказался от этой идеи.

```
1
2  #include "includes.h"
3  #include "lzw.h"
4
5  void Menu(){
6      std::cout << "write 0 for exit" << std::endl;
7      std::cout << "write 1 for use in_file and out_file for code" << std::endl;
8      std::cout << "write 2 for use in_file and out_file for decode" << std::endl;
9  }
10
11 int main (){
12
13     std::ofstream out;
14     std::ofstream in;
15
16     LZW zip;
17     Menu();
18
19     unsigned short int act = 5;
20     std::string tmpfn1;
21     std::string fn1;
22     std::string fn2;
23     std::string tmp;
24
25     while (act != 0){
26         std::cin >> act;
27         switch (act){
28         case 1:{
29             std::cout << "Enter the file name to compress." << std::endl;
30             std::cin >> fn1;
31
32             std::ofstream out(fn1 + ".z",std::ios::binary);
33             std::streambuf *coutbuf = std::cout.rdbuf(); //save old buf
```

```
34              std::cout.rdbuf(out.rdbuf()); //redirect std::cout to out.txt!
35
36              std::ifstream in(fn1);//std::ios_base::in|std::ios_base::binary
37              std::streambuf *cinbuf = std::cin.rdbuf(); //save old buf
38              std::cin.rdbuf(in.rdbuf()); //redirect std::cin to in.txt!
39
40              zip.code();
41
42              std::cin.rdbuf(cinbuf); //reset to standard input again
43              std::cout.rdbuf(coutbuf); //reset to standard output again
44              break;
45          }
46          case 2:{ //decode infile -> outfile
47              std::cout << "Enter the file name to decompress." << std::endl;
48              std::cin >> fn1;
49
50              if (fn1.back() != 'z' || fn1[fn1.length() - 2] != '.'){
51                  std::cout << "Bad file " << std::endl;
52                  return 0;
53              }
54
55              fn2 = fn1;
56              fn2.erase(fn2.length() - 2, fn2.length() - 1);
57
58              std::ofstream out(fn2 + ".txt");
59              std::streambuf *coutbuf = std::cout.rdbuf(); //save old buf
60              std::cout.rdbuf(out.rdbuf()); //redirect std::cout to out.txt!
61
62              std::ifstream in(fn1,std::ios_base::binary);
63              std::streambuf *cinbuf = std::cin.rdbuf(); //save old buf
64              std::cin.rdbuf(in.rdbuf()); //redirect std::cin to in.txt!
65
66              zip.decode();
67
68              std::cin.rdbuf(cinbuf); //reset to standard input again
69              std::cout.rdbuf(coutbuf); //reset to standard output again
70              break;
71          }
72          default:
73              return 0;
74              break;
75          }
76          Menu();
77      }
78      return 0;
79  }
```

3 Консоль

```
dude@DESKTOP-545VSUH:/mnt/d/education/education/DA/DA_KP/KP_DA_Timofeev_207$
ls
a.out  acaga  includes.h  lzw.h  main.cpp  makefile
dude@DESKTOP-545VSUH:/mnt/d/education/education/DA/DA_KP/KP_DA_Timofeev_207$
./a.out
write 0 for exit
write 1 for use in_file and out_file for code
write 2 for use in_file and out_file for decode
1
Enter the file name to compress.
acaga
write 0 for exit
write 1 for use in_file and out_file for code
write 2 for use in_file and out_file for decode
0
dude@DESKTOP-545VSUH:/mnt/d/education/education/DA/DA_KP/KP_DA_Timofeev_207$
ls
a.out  acaga  acaga.z  includes.h  lzw.h  main.cpp  makefile
dude@DESKTOP-545VSUH:/mnt/d/education/education/DA/DA_KP/KP_DA_Timofeev_207$
./a.out
write 0 for exit
write 1 for use in_file and out_file for code
write 2 for use in_file and out_file for decode
2
Enter the file name to decompress.
acaga.z
write 0 for exit
write 1 for use in_file and out_file for code
write 2 for use in_file and out_file for decode
0
dude@DESKTOP-545VSUH:/mnt/d/education/education/DA/DA_KP/KP_DA_Timofeev_207$
ls
a.out  acaga  acaga.txt  acaga.z  includes.h  lzw.h  main.cpp  makefile
dude@DESKTOP-545VSUH:/mnt/d/education/education/DA/DA_KP/KP_DA_Timofeev_207$
cat acaga
acagaatagagaacagaatagaga
dude@DESKTOP-545VSUH:/mnt/d/education/education/DA/DA_KP/KP_DA_Timofeev_207$
cat acaga.txt
acagaatagagaacagaatagaga
```

4 Выводы

Алгоритм LZW позволяет достичь одну из наилучших степеней сжатия среди других существующих методов сжатия графических данных, при полном отсутствии потерь или искажений в исходных файлах. В настоящее время используется в файлах формата TIFF, PDF, GIF, PostScript и других, а также отчасти во многих популярных программах сжатия данных (ZIP, ARJ, LHA).

К преимуществам алгоритма LZW можно отнести:
LZW является однопроходным.
Для декомпрессии не надо сохранять таблицу строк в файл для распаковки. Алгоритм построен таким образом, что мы в состоянии восстановить таблицу строк, пользуясь только потоком кодов.

К недостаткам алгоритма LZW относится:
Алгоритм не проводит анализ входных данных.

Выполнив курсовой проект, я научился реализовывать алгоритм LZW, разобрался в основах кодирования и декодирования данных.

Список литературы

[1] *Алгоритмы LZW, LZ77 и LZ78.*
URL: `https://habr.com/ru/post/132683/` (дата обращения: 20.06.2021).

[2] *Алгоритм LZW.*
URL: `https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_LZW` (дата обращения: 19.06.2021).

[3] *Алгоритм Лемпеля — Зива — Велча.*
URL: `https://ru.wikipedia.org/wiki/Алгоритм_Лемпеля_-_Зива_-_Велча` (дата обращения: 20.06.2021).