Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект
По курсу «Дискретный анализ»
4 семестр

Студент: В. С. Епанешников
Преподаватель: С. А. Сорокин
Группа: М8О-306Б-19

Дата:
Оценка:
Подпись:

Москва, 2021

1 Постановка задачи

Задача: Необходимо реализовать программу, которая сможет сжимать и расжимать файлы, используяю алгоритм LZ78.
Ключи:

1. -h справочная информация
2. -c filename (закодировать файл filename)
3. -d filename (декодировать файл filename)

2 Описание

Требуется написать реализацию алгоритма LZ-78 для кодирования текста.
Алгоритм LZ-78 в явном виде использует словарный подход, в отличие от алгоритма LZ-77, генерируя временный словарь во время кодирования и декодирования.

Изначально словарь пуст, а алгоритм пытается закодировать первый символ. На каждой итерации мы пытаемся увеличить кодируемый префикс, пока такой префикс есть в словаре. Кодовые слова такого алгоритма будут состоять из двух частей — номера в словаре самого длинного найденного префикса (pos) и символа, который идет за этим префиксом ($next$). При этом после кодирования такой пары префикс с приписанным символом добавляется в словарь, а алгоритм продолжает кодирование со следующего символа. Пара хранится в виде структуры *TNode*. В качестве словаря будем используется обычный map. При кодировании и декодировании Выводится информация о старом и новой файле (имя и размер)

3 Исходный код

LZ78.hpp

```
#ifndef LZ78_HPP
#define LZ78_HPP

#include <iostream>
#include <vector>
#include <fstream>
#include <map>

const uint64_t SIZE_OF_PIECE = (1ll << 10) * 32; // 32KB
const uint64_t DICT_SIZE = 1ll << 16;

struct TNode {
    uint16_t pos;
    char next;

    TNode ();
    TNode (uint16_t pos, char next);
    ~TNode ();

    static uint64_t SizeOfNode ();

    friend std::ifstream &operator>> (std::ifstream &input, TNode &node);
    friend std::ofstream &operator<< (std::ofstream &output, const TNode &node);

};

void Compress (const std::string &filename);
void Decompress (const std::string &filename);

#endif
```

LZ78.cpp

```
#include "LZ78.hpp"

TNode::TNode() : pos(0), next('\0') {}

TNode::TNode(uint16_t pos, char next) : pos(pos), next(next) {}

TNode::~TNode() {}

uint64_t TNode::SizeOfNode() {
    return sizeof(pos) + sizeof(next);
}

std::ifstream &operator>>(std::ifstream &input, TNode &node) {
```

```
14      input.read(reinterpret_cast<char*>(&node.pos), sizeof(node.pos));
15      input.read(reinterpret_cast<char*>(&node.next), sizeof(node.next));
16      return input;
17  }
18
19  std::ofstream &operator<<(std::ofstream &output, const TNode &node) {
20      output.write(reinterpret_cast<const char*>(&node.pos), sizeof(node.pos));
21      output.write(reinterpret_cast<const char*>(&node.next), sizeof(node.next));
22      return output;
23  }
24
25  void Compress(const std::string &filename) {
26      std::ifstream file(filename);
27      if (!file) {
28          std::cout << "Can't open file \"" << filename << "\"" << std::endl;
29          return;
30      }
31
32      std::ofstream output(filename + ".comp", std::ios::binary);
33      uint64_t oldSize = 0, newSize = 0, toRead = 0;
34
35      file.seekg(0, std::ios::end);
36      oldSize = file.tellg();
37      file.seekg(0);
38
39      std::string str, buffer;
40      str.resize(SIZE_OF_PIECE);
41
42      char last_char;
43      std::map<std::string, uint64_t> dict;
44
45      toRead = oldSize > SIZE_OF_PIECE ? SIZE_OF_PIECE : oldSize;
46
47      file.read(&str[0], toRead);
48      uint64_t j = 0;
49      for (uint64_t i = 0; i < oldSize; ++i) {
50          if (j == str.size()) {
51              j = 0;
52              toRead = oldSize - i > SIZE_OF_PIECE ? SIZE_OF_PIECE : oldSize - i;
53              file.read(&str[0], toRead);
54          }
55          if (dict.count(buffer + str[j]) && i != oldSize - 1) {
56              buffer += str[j];
57          } else {
58              uint16_t pos = dict[buffer] == 0 ? 0 : dict[buffer] - 1;
59              output << TNode(pos, str[j]);
60              if (dict.size() != DICT_SIZE) {
61                  dict[buffer + str[j]] = dict.size();
62              }
```

```
63              buffer.clear();
64              newSize += TNode::SizeOfNode();
65          }
66          ++j;
67      }
68      if (buffer != "") {
69          last_char = buffer.back();
70          buffer.pop_back();
71          output << TNode(dict[buffer], last_char);
72          newSize += TNode::SizeOfNode();
73      }
74
75      std::cout << filename << " --> " << filename + ".comp\n";
76      std::cout << oldSize << " bytes --> " << newSize << " bytes\n";
77      file.close();
78      output.close();
79  }
80
81
82  void Decompress(const std::string &filename) {
83      std::ifstream file(filename, std::ios::binary);
84      if (!file) {
85          std::cout << "Can't open file \"" << filename << "\"" << std::endl;
86          return;
87      }
88      std::ofstream output(filename + ".deco");
89      TNode node;
90      std::string ans, word;
91      std::vector<std::string> dict;
92      uint64_t inputSize = 0, outputSize = 0;
93      dict.push_back("");
94
95      file.seekg(0, std::ios::end);
96      inputSize = file.tellg();
97      file.seekg(0);
98
99      while (file >> node) {
100         word = dict[node.pos] + node.next;
101         ans += word;
102         if (dict.size() != DICT_SIZE) {
103             dict.push_back(word);
104         }
105         if (ans.size() > SIZE_OF_PIECE) {
106             outputSize += ans.size();
107             output << ans;
108             ans.clear();
109         }
110     }
111     if (ans.size() > 0) {
```

```
        outputSize += ans.size();
        output << ans;
    }

    std::cout << filename << " --> " << filename + ".deco\n";
    std::cout << inputSize << " bytes --> " << outputSize << " bytes\n";

    file.close();
    output.close();
}
```

main.cpp

```
#include <iostream>
#include "LZ78.hpp"

void help () {
    std::cout << "help: -h\ncompress: -c filename\ndecompress: -d filename\n";
}

int main (int argc, char *argv[]) {
    if (argc < 2 || argc > 3) {
        std::cout << "Incorrect count of arguments. Try \"" << std::string(argv[0]) <<
            " -h\"" << std::endl;
        return 1;
    }
    std::string command(argv[1]);
    if (command == "-h") {
        help();
    } else if (command == "-c") {
        Compress(argv[2]);
    } else if (command == "-d") {
        Decompress(argv[2]);
    } else {
        std::cout << "Invalid command. Try \"" << std::string(argv[0]) << " -h\"" <<
            std::endl;
    }
    return 0;
}
```

4 Консоль

Кодирование, декодирование и сравнение файла.

```
vladislove@macbook:~/DA/KP$ ./a.out -c test1.txt
test1.txt --> test1.txt.comp
4642 bytes --> 3519 bytes

vladislove@macbook:~/DA/KP$ ./a.out -d test1.txt.comp
test1.txt.comp --> test1.txt.comp.deco
3519 bytes --> 4642 bytes
vladislove@macbook:~/DA/KP$ diff test1.txt test1.txt.comp.deco
vladislove@macbook:~/DA/KP$
```

Кодирование текста, состоящего из многократно повторяющегося сивола

```
vladislove@macbook:~/DA/KP$ ./a.out -c test2.txt
test2.txt --> test2.txt.comp
3705 bytes --> 258 bytes

vladislove@macbook:~/DA/KP$ ./kp -d test3.txt.comp
test2.txt.comp --> test2.txt.comp.deco
258 bytes --> 3705 bytes
vladislove@macbook:~/DA/KP$ diff test2.txt test2.txt.comp.deco
vladislove@macbook:~/DA/KP$
```

5 Выводы

В ходе выполнения курсового проекта я познакомился с алгоритмами сжатия данных семейства LZ. Был создан архиватор, использующий алгоритм LZ-78.

Так же была замечена низкая эффективность алгоритма на маленьких объёмах данных.

Список литературы

[1] Алгоритмы LZ77 и LZ78 - neerc
URL: https://neerc.ifmo.ru/wiki/index.php?title=Алгоритмы_LZ77_и_LZ78#LZ78

[2] Алгоритмы LZW, LZ77 и LZ78 - habr
URL: https://habr.com/ru/post/132683/