Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

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

Студент: Р. С. Лисин
Преподаватель: С. А. Сорокин
Группа: М8О-306Б-20

Дата:
Оценка:
Подпись:

Москва, 2022

Задание

Задача: Реализуйте алгоритм LZW.

Начальный словарь выглядит следующим образом: a -> 0 b -> 1 c -> 2 ... x -> 23 y -> 24 z -> 25 EOF -> 26.

Формат входных данных: Вам будут даны тесты 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.

Формат результата: В ответ на тест первого типа вам нужно вывести коды, которыми будет закодирован данный текст через пробел.

В ответ на тест второго типа выведите разжатый текст.

1 Описание

Согласно [1] непосредственным предшественником LZW является алгоритм LZ78, опубликованный Абрахамом Лемпелем (Abraham Lempel) и Якобом Зивом (Jacob Ziv) в 1978 г. Этот алгоритм воспринимался как математическая абстракция до 1984 г., когда Терри Уэлч (Terry A. Welch) опубликовал свою работу с модифицированным алгоритмом, получившим в дальнейшем название LZW (Lempel-Ziv-Welch).

Кодирование.

В методе сжатия LZW используется начальный словарь всех различных символов кодируемого текста. Он может строиться путем анализа всего текста.

- 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.

Нужно иметь такой же словарь всех символов текста. Только в этот раз ключами должны быть коды символов, а значениями сами символы.

- 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.
- 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.

Сложность алгоритмов кодирования и декодирования: $O(n)$, где n - количество символов в тексте.

2 Исходный код

```
#include <iostream>
#include <vector>
#include <unordered_map>

using namespace std;

unordered_map<string, unsigned long long> dict = {{"a", 0}, {"b", 1}, {"c", 2}, {"d", 3}, {"e", 4}, {"f", 5}, {"g", 6}, {"h", 7}, {"i", 8}, {"j", 9}, {"k", 10}, {"l", 11}, {"m", 12}, {"n", 13}, {"o", 14}, {"p", 15}, {"q", 16}, {"r", 17}, {"s", 18}, {"t", 19}, {"u", 20}, {"v", 21}, {"w", 22}, {"x", 23}, {"y", 24}, {"z", 25}};
unordered_map<unsigned long long, string> reversed_dict = {{0, "a"}, {1, "b"}, {2, "c"}, {3, "d"}, {4, "e"}, {5, "f"}, {6, "g"}, {7, "h"}, {8, "i"}, {9, "j"}, {10, "k"}, {11, "l"}, {12, "m"}, {13, "n"}, {14, "o"}, {15, "p"}, {16, "q"}, {17, "r"}, {18, "s"}, {19, "t"}, {20, "u"}, {21, "v"}, {22, "w"}, {23, "x"}, {24, "y"}, {25, "z"}};


void LZW_encode(string s) {
   if (!s.size()) {
      cout << 26 << '\n';
      return;
   }

   unsigned long long index = 27;
   string prev_word = "";
   prev_word += s[0];
   bool flag = false;

   for (const auto& c: s) {

      if (!flag) {
         flag = true;
         continue;
      }

      if (dict.count(prev_word + c) == 0) {
         cout << dict[prev_word] << " ";
         dict[prev_word + c] = index++;
         prev_word = c;
      }

      else {
         prev_word += c;
      }
   }
   cout << dict[prev_word] << " ";
   cout << 26 << " " << '\n';
```

```
}


void LZW_decode() {
   unsigned long long index = 27;
   unsigned long long cur_code, prev_code;
   cin >> prev_code;
   if (prev_code == 26) {
    cout << '\n';
      return;
   }
   cout << reversed_dict[prev_code];
   while (cin >> cur_code) {

      if (cur_code == 26) {
         break;
      }

      string cur_word = "";
      if (reversed_dict.count(cur_code) == 0) {
         cur_word += reversed_dict[prev_code] + reversed_dict[prev_code][0];
      }
      else {
         cur_word += reversed_dict[cur_code];
      }
      cout << cur_word;

      reversed_dict[index++] = reversed_dict[prev_code] + cur_word[0];
      prev_code = cur_code;
   }
   cout << '\n';
}


int main() {
   string task, s;
   cin >> task;
   if (task == "compress") {
      cin >> s;
      LZW_encode(s);
   }
   else if (task == "decompress"){
      LZW_decode();
   }
   return 0;
}
```

3 Примеры работы

```
roma@DESKTOP-JD58QU2:~/Diskran/cp$ g++ main.cpp
roma@DESKTOP-JD58QU2:~/Diskran/cp$ ./a.out
compress
abracadabra
0 1 17 0 2 0 3 27 29 26
roma@DESKTOP-JD58QU2:~/Diskran/cp$ g++ main.cpp
roma@DESKTOP-JD58QU2:~/Diskran/cp$ ./a.out
decompress
0 1 17 0 2 0 3 27 29 26
abracadabra
roma@DESKTOP-JD58QU2:~/Diskran/cp$ g++ main.cpp
roma@DESKTOP-JD58QU2:~/Diskran/cp$ ./a.out
compress
abacabadabacabae
0 1 0 2 27 0 3 31 30 28 4 26
roma@DESKTOP-JD58QU2:~/Diskran/cp$ g++ main.cpp
roma@DESKTOP-JD58QU2:~/Diskran/cp$ ./a.out
decompress
0 1 0 2 27 0 3 31 30 28 4 26
abacabadabacabae
```

4 Выводы

Выполнив курсовой проект по курсу «Дискретный анализ», я изучил алгоритм LZW, который является одним из самых популярных алгоритмов кодирования. Это помогло мне лучше понять процесс и смысл кодирования.

Список литературы

[1] *Алгоритм LZW.*
URL: `https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_LZW` (дата обращения: 24.12.2022).