Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Дискретный анализ»
«Алгоритм LZ77»

Студент: Д. М. Чистяков
Преподаватель: С. А. Сорокин
Группа: М8О-306Б-20
Дата:
Оценка:
Подпись:

Москва, 2023

1 Условие

Задача:

1. **Вариант LZ77** Реализуйте алгоритм LZ77. Без окна, поиск производится по всему просмотренному ранее тексту.
2. Первый тип: текст состоит только из малых латинских букв. В ответ на него вам нужно вывести тройки, которыми будет закодирован данный текст.
3. Второй тип: вам даны тройки (<offset, len, char>) в которые был сжат текст из малых латинских букв, вам нужно его разжать.

2 Метод решения

Был создан класс *LZ77*, в котором реализованы методы для кодирования и декодирования текста. Метод *LZ77::Encode* принимает на вход строку и возвращает вектор троек, алгоритм использует наивный поиск строки в буфере и имеет итоговую сложность $O(n^3)$. Метод *LZ77::Decode* принимает на вход вектор троек и возвращает декодированную строку, работает за $O(n)$.

3 Исходный код

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

class LZ77 {
  public:
  struct Node {
    int offset;
    int size;
    char symbol;
  };

  static inline const char EOM = ' ';

  static void Encode(std::string& input, std::vector<LZ77::Node>& output) {
    int pos = 0;
    while (pos < input.size()) {
      LZ77::Node n = {0, 0, input[pos]};
      for (int i = 0; i < pos; i++) {
        if (input[i] != input[pos]) {
          continue;
        }
        int size = 1;
        while (pos + size < input.size() && input[i + size] == input[pos + size]) {
          size++;
        }
        if (size >= n.size) {
          n.size = size;
          n.offset = pos - i;
        }
      }
      if (pos + n.size >= input.size()) {
        n.symbol = LZ77::EOM;
      }
      else {
        n.symbol = input[pos + n.size];
      }
      output.push_back(n);
      pos += n.size + 1;
    }
  }

  static void Decode(std::vector<LZ77::Node>& input, std::string& output) {
    for (LZ77::Node& n : input) {
      if (n.offset != 0) {
        int pos = output.size();
        for(int i = 0; i < n.size; i++) {
          output.push_back(output[pos - n.offset + i]);
```

```
48        }
49      }
50
51      if (n.symbol != EOM) {
52        output.push_back(n.symbol);
53      }
54    }
55  }
56};
57
58int main() {
59  std::ios::sync_with_stdio(false);
60  std::cin.tie(0);
61  std::cout.tie(0);
62  std::string cmd;
63  std::cin >> cmd;
64
65  if (cmd == "compress") {
66    std::string text;
67    std::cin >> text;
68    std::vector<LZ77::Node> res;
69    LZ77::Encode(text, res);
70    for (LZ77::Node& n : res) {
71      std::cout << n.offset << " " << n.size << " " << n.symbol << "\n";
72    }
73  }
74
75  if (cmd == "decompress") {
76    std::vector<LZ77::Node> code;
77    LZ77::Node n;
78    while (std::cin >> n.offset >> n.size) {
79      if (!(std::cin >> n.symbol)) {
80        n.symbol = LZ77::EOM;
81      }
82      code.push_back(n);
83    }
84
85    std::string res;
86    LZ77::Decode(code, res);
87
88    std::cout << res << "\n";
89  }
90  return 0;
91}
```

4 Консоль

```
den@vbox:~/Документы/DA/CP$ g++ cp.cpp
den@vbox:~/Документы/DA/CP$ ./a.out
compress
abracadabra
0 0 a
0 0 b
0 0 r
3 1 c
2 1 d
7 4
den@vbox:~/Документы/DA/CP$ ./a.out
decompress
0 0 a
0 0 b
0 0 r
3 1 c
2 1 d
7 4
abracadabra
```

5 Выводы

В результате выполнения курсового проэкта я реализовал алгоритм LZ77. Моя реализация простая и использует наивный алгоритм поиска подстроки, но алгоритм можно улучшить используя суффиксное дерево, тогда сложность кодирования станет $O(n^2)$.