ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

CHALLENGE 3: NÉN DỮ LIỆU

GVLT: ThS VĂN CHÍ NAM

GVTH: BÙI HUY THÔNG

DANH SÁCH THÀNH VIÊN:

20120352 – VŨ HOÀNG PHÚC

20120373 – LÊ TRƯƠNG KINH THÀNH

20120482 - NGUYỄN TẠ HUY HOÀNG

THÀNH PHỐ HỒ CHÍ MINH – 2021

Mục lục

I. **THUẬT TOÁN HUFFMAN TĨNH** 2
1. Khái niệm: **2**
2. Pha nén **2**
- Ý tưởng: 2
- Các bước thực hiện: 2
- Phát sinh mã bit: 2
- Ví dụ 2
3. Pha giải nén **4**
- Ý tưởng 4
- Các bước thực hiện 5
- Ví dụ 5
4. Ưu điểm và nhược điểm: **5**
- Ưu điểm: 5
- Nhược điểm: 5
5. Ứng dụng thực tế và phiên bản cải tiến **5**
II. THUẬT TOÁN LEMPEL-ZIV-WELCH (LZW) 6
1. Khái niệm: **6**
2. Pha nén: **7**
- Ý tưởng: 7
- Chức năng của các hàm: 7
3. Pha giải nén: **9**
- Ý tưởng: **9**
4. Trường hợp đặc biệt **9**
5. Ưu điểm và nhược điểm: **9**
- Ưu điểm: 9
- Nhược điểm: 10
 Tốn nhiều bộ nhớ. **10**
6. Ứng dụng thực tế và phiên bản cải tiến: **10**

I. THUẬT TOÁN HUFFMAN TĨNH

1. Khái niệm:

- Nén Huffman là 1 thuật toán mã hoá thông tin dùng để nén dữ liệu dựa trên việc tối ưu hoá việc mã hoá các kí tự trong chuỗi ban đầu bằng việc xây dựng bộ mã nhị phân đại diện cho từng ký tự trong đó.

- Thuật toán có mục tiêu sẽ xây dựng được bảng mã nhị phân đại diện cho từng kí tự sao cho những kí tự có tần suất xuất hiện nhiều sẽ có mã nhị phân đại diện cho nó ngắn và ngược lại.

2. Pha nén

- **Ý tưởng:**
  + Phương pháp cũ: dùng 1 dãy bit cố định để biểu diễn 1 ký tự
  + David Huffman(1952): tìm ra phương pháp xác định mã tối ưu trên dữ liệu tĩnh :
    - Sử dụng vài bit để biểu diễn 1 ký tự (gọi là "mã bit"–bit code)
    - Độ dài "mã bit" cho các ký tự không giống nhau:
    - Ký tự xuất hiện nhiều lần: biểu diễn bằng mã ngắn;
    - Ký tự xuất hiện ít : biểu diễn bằng mã dài

  ⇨ Mã hóa bằng mã có độ dài thay đổi (Variable Length Encoding)

  Tổng quan của bài toán là việc thay vì biểu diễn từng ký tự trong 1 chuỗi bằng dãy nhị phân 8 bit sẽ gây dư thừa những đoạn nhị phân không cần thiết. Thuật toán sẽ cố gắng xây dựng một bộ ánh xạ từ kí tự trong chuỗi sang 1 chuỗi nhị phân mới sao cho những ký tự xuất hiện càng nhiều sẽ có chuỗi nhị phân kích thước càng ngắn. Qua đó sẽ giúp biểu diễn lại chuỗi ký tự đó thành 1 chuỗi nhị phân có kích thước ngắn hơn.

- **Các bước thực hiện:**

[B1]: Duyệt tập tin-> Lập bảng thống kê tần số xuất hiện của các ký tự trong chuỗi
[B2]: Xây dựng cây Huffman dựa vào bảng thống kê tần số xuất hiện
[B3]: Phát sinh bảng mã bit cho từng ký tự tương ứng
[B4]: Duyệt tập tin-> Thay thế các ký tự trong tập tin bằng mã bit tương ứng.
[B5]: Lưu lại thông tin của cây Huffman cho giải nén

- **Phát sinh mã bit:** Mã bit của từng ký tự: đường đi từ node gốc của cây Huffman đến node lá của ký tự đó:
  - **Cách thức:**
    - Bit 0 được tạo ra khi đi qua nhánh trái
    - Bit 1 được tạo ra khi đi qua nhánh phải

- **Ví dụ**

Cho đoạn txt: ABBACCDADCBADCBABCDABCDACDBABAAB

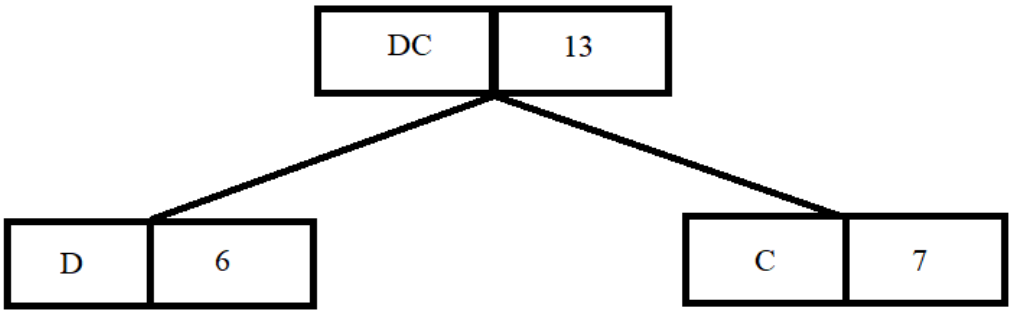
- Bảng thống kê:

A	B	C	D
10	9	7	6

- Bảng thống kê đã được sắp xếp

D	C	B	A
6	7	9	10

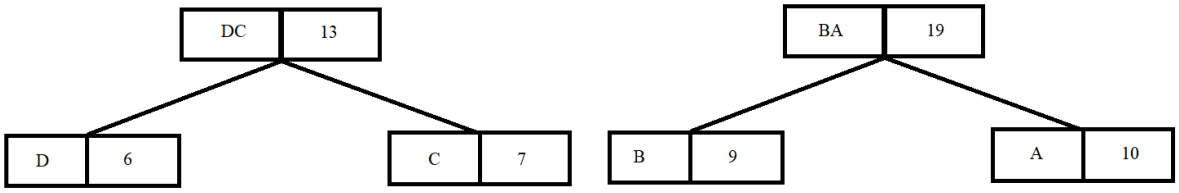
[B1]: C và D là 2 node có trọng số nhỏ nhất



- Xóa C và D ra khỏi bảng số liệu. Thêm DC vào bảng số liệu và sắp xếp:

B	A	DC
9	10	13

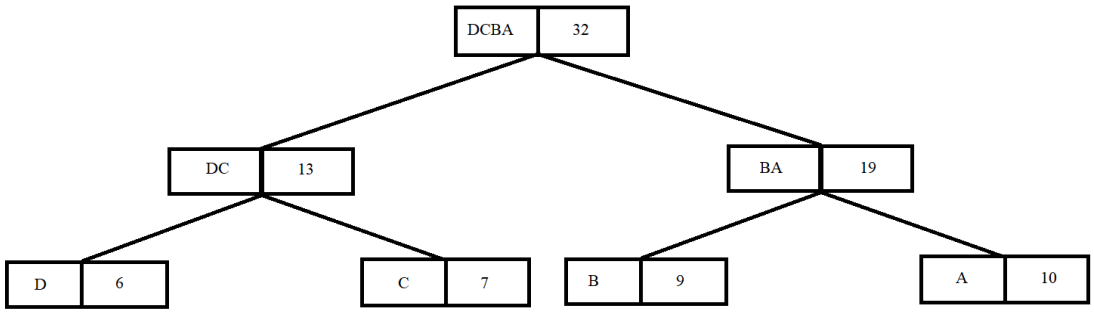
[B2]: A và B là 2 node có trọng số nhỏ nhất



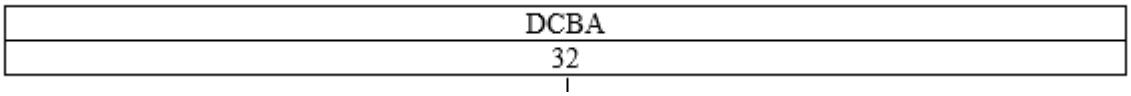
Xóa A và B ra khỏi bảng số liệu. Thêm BA vào bảng số liệu và sắp xếp:

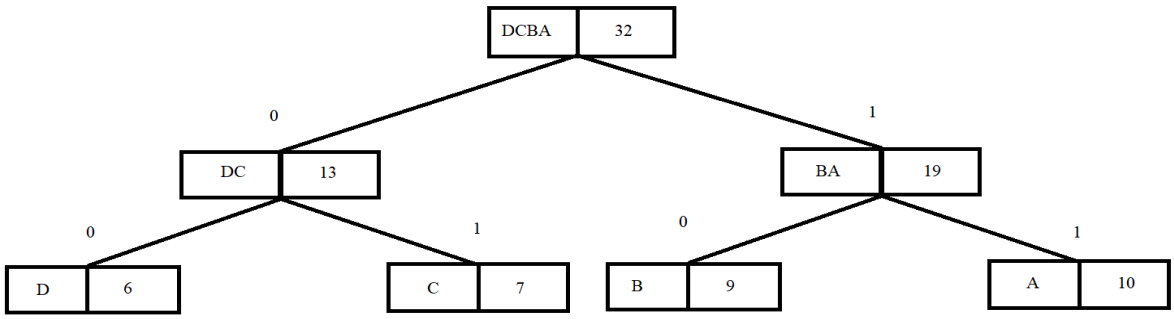
DC	BA
13	19

[B3]: DC và BA là 2 node có trọng số nhỏ nhất.



Xóa DC và BA ra khỏi bảng số liệu. Thêm DCBA vào bảng số liệu và sắp xếp lại:





***** Qua quá trình phát sinh mã bit ta được:**

A	B	C	D
11	10	01	00

- **Chuỗi ban đầu:** ABBACCDADCBADCBABCDABCDACDBABAAB
- **Chuỗi sau khi mã hóa:**
  11 10 10 11 01 01 00 11 00 01 10 11 00 01 10 11 10 01 00 11 10 01 00 11 01 00 10 11 10 11 11 10

3. Pha giải nén

Đầu vào: Chuỗi nhị phân đã mã hoá C của 1 tệp xác định, cây nhị phân Huffman. T
Đầu ra: Chuỗi dữ liệu ban đầu lúc chưa mã hoá.O

- **Ý tưởng:** là việc đọc từng bit của chuỗi C và tiến hành duyệt cây Huffman đến khi đạt đến nút lá thì ngắt phần chuỗi nhị phân đã đọc ra và thay bằng ký tự đại diện của node lá.

- **Các bước thực hiện:**
  Để giải quyết bài toán này, mình dùng 1 biến mảng A để lưu các kí tự của bản rõ O, dùng 1 vòng lặp for duyệt trên chuỗi C và đọc từ bit. Đặt 1 con trỏ P ở node gốc cây T, ứng với từng bit, đọc được, mình sẽ di chuyển con trỏ P sang node con bên trái nếu đọc được bit 0 và sang phải nếu đọc được bit 1. Và kiểm tra vị trí hiện hành của con trỏ P có phải ở node lá không. Nếu chưa thì đọc tiếp bit kế và thực hiện lại các bước trên. Nếu đúng thì đưa con trỏ P vè node gốc của cây T và xuất ra ký tự tìm được vào A.

- **Ví dụ:**
  + Từ ví dụ của phần nén ta được dãy bit: 11 10 10 11 01 01 00 11 00 01 10 11 00 01 10 11 10 01 00 11 10 01 00 11 01 00 10 11 10 11 11 10
  + Duyệt theo cách đã nói ở trên để lần lượt được từng ký tự từ dãy bit:
  11 10 10 11 01 01 00 11 00 …
  A | B | B | A | C | C | D | A | D …

4. Ưu điểm và nhược điểm:

- **Ưu điểm:**
  - Hệ số nén tương đối cao.
  - Phương pháp thực hiện tương đối đơn giản.
  - Đòi hỏi ít bộ nhớ.
- **Nhược điểm:**
  - Cần 2 lần duyệt file: lần 1 lập bảng tần suất, lần 2 để thay thế các ký tự bảng mã Huffman tương ứng
  - Phải lưu lại thông tin để giải nén: cây nhị phân Huffman hoặc mã Huffman hoặc bảng tần suất
  - Dữ liệu tĩnh, có sẵn và hoàn chỉnh: để có thể lập được bảng tần suất

5. Ứng dụng thực tế và phiên bản cải tiến

- Trong khoa học máy tính và lí thuyết công nghệ thông tin, nén dữ liệu là quá trình mã hóa thông tin dùng ít bit hơn so với thông tin dữ liệu gốc, làm giảm lượng thông tin " dư thừa " trong dữ liệu gốc.
- Dựa vào nguyên tắc này giúp tránh hiện tượng kênh truyền bị quá tải và truyền tin trở nên dễ dàng hơn. Ngoài ra nó giảm được nguồn tài nguyên cũng như dung lượng lưu trữ hay băng thông đường truyền. Tuy nhiên, vì dữ liệu nén cần được giải nén nên sẽ đòi hỏi nhiều phần cứng và xử lý
- Vì thuật toán nén của ta sử dụng ở trên là Huffman tĩnh nên phiên bản cải tiến của thuật toán trên là thuật toán Huffman động

II. THUẬT TOÁN LEMPEL-ZIV-WELCH (LZW)

Khái niệm nén từ điển được Jacob Lampel và Abraham Ziv đưa ra lần đầu tiên vào năm 1977. Sau đó phát triển thành một họ giải thuật nén từ điển LZ. Năm 1984 Terry Welch đã cải tiến thuật giải LZ thành một họ giải thuật mới hiệu quả hơn và đặt tên là LZW.

1. Khái niệm:

- Phương pháp LZW dựa trên việc xây dựng từ điển cho các "chuỗi ký tự" đã từng xuất hiện trong văn bản, những "chuỗi ký tự" xuất hiện sau đó sẽ được thay thế bằng mã của nó trong bảng từ điển.
- Giải thuật LZW được sử dụng cho tất cả các loại file nhị phân. Nó thường được dùng để nén các loại văn bản, ảnh đen trắng, ảnh màu … và là chuẩn nén cho các dạng ảnh GIF, TIFF… Mức độ hiệu quả của LZW không phụ thuộc vào số bit màu của ảnh.

Phương pháp nén LZW

- Phương pháp LZW hoạt động theo nguyên tắc là tạo ra một từ điển động theo dữ liệu của file ảnh. Từ điển là tập hợp những cặp *Khoá* và *nghĩa* của nó. Trong đó *khoá* được sắp xếp theo thứ tự nhất định, *nghĩa* là một chuỗi con trong dữ liệu ảnh.
- Từ điển được xây dựng đồng thời với quá trình đọc dữ liệu. Sự có mặt của một chuỗi con trong từ điển khẳng định rằng chuỗi đó đã từng xuất hiện trong phần dữ liệu đã đọc. Thuật toán liên tục tra cứu và cập nhật từ điển sau mỗi lần đọc một ký tự ở dữ liệu đầu vào.
- Do kích thước bộ nhớ không phải là vô hạn và để đảm bảo tốc độ tìm kiếm, người ta thường dùng từ điển với kích thước 4096 (2^{12}) phần tử.

Cấu trúc từ điển có dạng như sau:

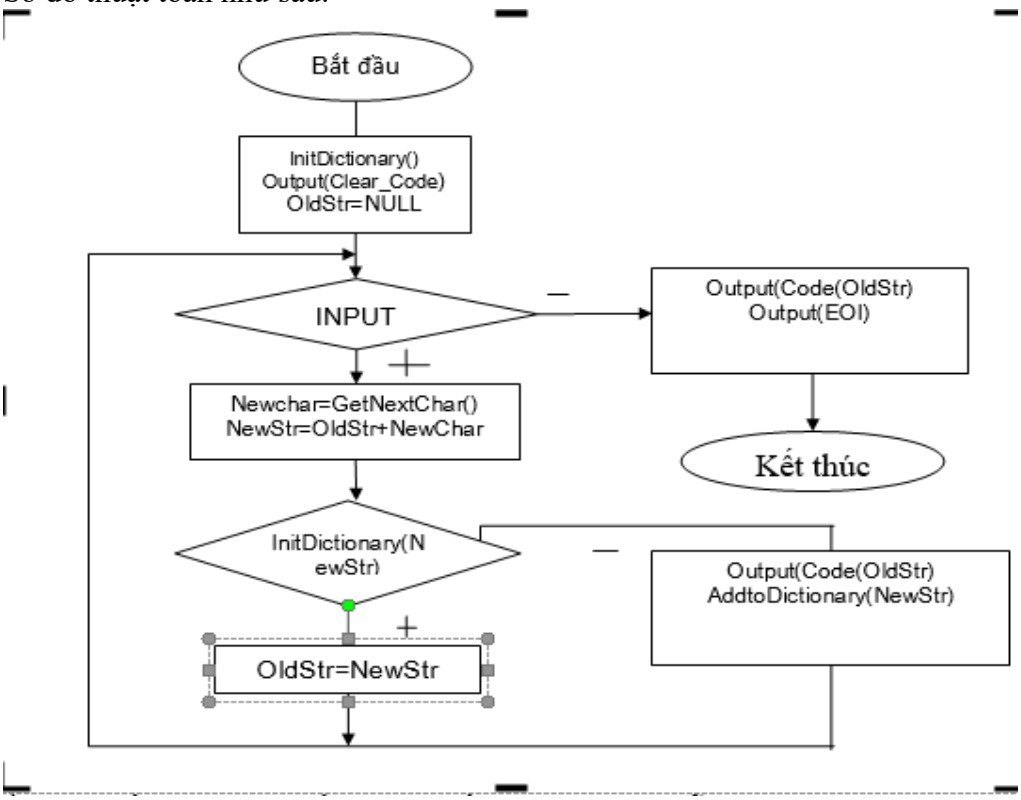
Khoá	Ý nghĩa	Ghi chú
0	0	
1	1	
...	...	
255	255	
256	256	*(Clear Code)*
257	257	*(End Of Information)*
258	Chuỗi	
259	Chuỗi	
...	...	
4095	Chuỗi	

- 256 từ mã đầu tiên có khoá từ 0 .. 255 chứa 256 ký tự của bảng mã ASCII).
- Từ mã thứ 256 chứa một mã đặc biệt là "mã xóa" (CC - Clear Code). Mục đích việc dùng mã xóa nhằm khắc phục tình trạng số mẫu lặp trong ảnh lớn hơn 4096. Khi đó một ảnh được quan niệm là nhiều mảnh ảnh, và từ điển là một bộ từ điển gồm nhiều từ điển con. Cứ hết một mảnh ảnh người ta lại gửi một mã xóa để báo hiệu kết thúc mảnh ảnh cũ, bắt đầu mảnh ảnh mới đồng thời khởi tạo lại từ điển cho mảnh ảnh mới.
- Từ mã thứ 257 chứa mã kết thúc thông tin (EOI - End Of Information). Một file ảnh có thể chứa nhiều ảnh (ví dụ ảnh GIF), khi đó mỗi ảnh sẽ được mã hóa riêng và mã EOI dùng để xác định điểm kết thúc thông tin của 1 ảnh.

- Các từ mã tiếp theo (từ 258 trở đi) chứa các mẫu lặp lại trong ảnh. Các từ mã nà được sinh ra trong quá trình mã hoá.

2. Pha nén:

Sơ đồ thuật toán như sau:



- **Ý tưởng:**

  Nếu còn dữ liệu đầu vào thì tiếp tục đọc.Một chuỗi mới sẽ được tạo ra từ một chuỗi cũ(chuỗi này ban đầu rỗng,chuỗi này phải là chuỗi đã tồn tại trong từ điển) và kí tự vừa đọc vào.Sau đó kiểm tra xem chuỗi mới đã có trong từ điển hay chưa.Mục đích của công việc này là hy vọng tìm được chuỗi có số ký tự lớn nhất đã tồn tại trong từ điển.Nếu tồn tại ta lại tiếp tục đọc một ký tự tiếp theo và lặp lại công việc.Nếu chưa có trong từ điển,thì gửi chuỗi cũ ra ngoài và thêm chuỗi mới vào từ điển.

- **Chức năng của các hàm:**

  - Giá trị cờ INPUT = true khi vẫn còn dữ liệu đầu vào và ngược lại
  - InitDictionary(): Hàm này có chức năng khởi tạo từ điển. Đặt giá trị cho 255 phần đầu tiên.Gán mã xóa(Clear Code) cho phần tử thứ 256 và mã kết thúc thông tin (End of Information) cho phần tử thứ 257. Xóa giá trị tất cả các phần tử còn lại
  - Hàm Output(): gửi chuỗi bit ra file. Chuỗi bit này có độ dài là 9,10,11, hoặc 12 tùy thuộc vào vị trí trong từ điển của từ mã gửi ra. Các chuỗi bít này được nối tiếp vào với nhau.
  - Hàm GetNextChar(): Trả về một ký tự đầu vào. Hàm này cập nhật giá trị của cờ INPUT xác định xem còn dữ liệu đầu vào nữa hay không?

- Hàm AddtoDictionary() sẽ được gọi khi có một mẫu mới xuất hiện. Hàm này sẽ cập nhật mẫu này vào phần tử tiếp theo trong từ điển. Nếu từ điển đã đầy nó sẽ gửi ra từ mã xóa(Clear Code) và gọi đến hàm InitDictionary() để khởi tạo lại từ điển.
- Hàm Code(): Trả về mã tương ứng với chuỗi

- **Ví dụ:** Sử dụng thuật toán LZW để nén chuỗi : **THISISTHE**

Các bước liên quan được thể hiện qua bảng sau:

Current	Code	Next	Code	Output	Code	Add to dictionary	Code
T	116	H	104	T	116	TH	256
H	104	I	105	H	104	HI	257
I	105	S	115	I	105	IS	258
S	115	I	105	S	115	SI	259
I	105	S	115	"IS" is in the dictionary, check "IST"			
IS	258	T	116	IS	258	IST	260
T	116	H	104	"TH" is in the dictionary, check "THE"			
TH	256	E	101	TH	256	THE	261
E	101	_		E	101	_	

Khi đó ra có **Output:** 116 104 105 115 258 256 101

Dưới dạng nhị phân(bit):

001110100 001101000 001101001 001110011 100000010 100000000 001100101

Current	Next	Output	Add to dictionary	Code
116	104	116	116 104	256
104	105	104	104 105	257
105	115	105	105 115	258
115	258	115	115 105 115	259
258	256	105 115	105 115 116	260
256	101	116 104	116 104 101	261
101	_	101	_	_

Ta thu được chuỗi output:

116 104 105 115 105 115 116 104 101

Hay: **"THISISTHE"**

3. Pha giải nén:

- Ý tưởng:

Giải thuật giải nén gần như ngược với giải thuật nén.Với giải thuật nén,một từ mã ứng với một chuỗi sẽ được ghi ra tệp khi chuỗi ghép bởi chuỗi trên với ký tự vừa đọc chưa có trong từ điển.Người ta cũng cập nhật ngay vào từ điển từ mã ứng với chuỗi tạo bởi chuỗi cũ với ký tự vừa đọc.Ký tự này đồng thời là ký tự đầu tiên trong chuỗi ứng với từ mã sẽ được ghi ra tiếp theo.Đây là điểm mấu chốt cho phép xây dựng thuật toán giải nén.

- Ví dụ:

Sử dụng LZW để giải nén chuỗi : 116 104 105 115 258 256 101

4. Trường hợp đặc biệt

- Giải thuật LZW như trên có thể dẫn đến một tình huống là quá trình nén thì thực hiện được, còn quá trình giải nén thì "không" thực hiện được.

- Tình huống đó sảy ra như sau: giả sử c là một kí tự, S là một chuỗi (có độ dài lớn hơn 0); mã k của từ điển chứa giá trị là cS. Nếu chuỗi vào tiếp theo là cScSc, khi đọc đến cSc chương trình sẽ tạo mã k' chứa cSc (vì cS đã có mã là k), theo thuật toán nén ký tự c được giữ lại và chương trình đọc tiếp xâu S, ký tự c để hình thành từ cSc, khi đó mã k' sẽ được dùng thay cho cSc.

- Trong chương trình giải nén, k' sẽ xuất hiện trước khi nó được định nghĩa. Rất may là từ mã vừa đọc trong trường hợp này bao giờ cũng có nội dung trùng với tổ hợp của từ mã cũ với kí tự đầu tiên của nó. Điều này giúp cho quá trình cài đặt chương trình khắc phục được trường hợp ngoại lệ một cách dễ dàng.

5. Ưu điểm và nhược điểm:

- Ưu điểm:

- Thuật toán LZW đặc biệt có hiệu quả khi sử dụng để nén file văn bản vì độ lặp lại của ký tự là lớn.
- Có thể nén dữ liệu đầu vào trong 1 lần truyền
- Không yêu cầu thông tin trước về inout
- Tỉ lệ nén: 2÷5
- Độ phức tạp: Đơn giản
- Tốc độ nén: Bình thường
- Hệ số nén tương đối cao, trong tập tin nén không cần phải chứa bảng mã.

- **Nhược điểm:**
  - Thuật toán Shanon có hệ số nén khá thấp và yêu cầu khá phức tạp nên hiếm khi được sử dụng
  - Tốn nhiều bộ nhớ.

6. Ứng dụng thực tế và phiên bản cải tiến:

Áp dụng cho tất cả các file nhị phân. Thường dùng để nén các loại văn bản,ảnh đen trắng,ảnh màu,ảnh đa mức xám...và là chuẩn nén cho các định dạng ảnh GIF và TIFF.Mức độ hiệu quả của LZW không phụ thuộc vào số bít màu của ảnh. Được sử dụng trong hệ điều hành UNIX để nén các dữ liệu nhất định.

III)LẬP TRÌNH:

1)Các kiểu dữ liệu sử dụng:

-Vector

-String

2)Các hàm quan trọng trong Project:

-Nén:

```
vector<int> encoding(string s1)
{
   unordered_map<string, int> table;
   for (int i = 0; i <= 255; i++) {
      string ch = "";
      ch += char(i);
      table[ch] = i;
   }

   string p = "", c = "";
   p += s1[0];
   int code = 256;
   vector<int> output_code;
   for (int i = 0; i < s1.length(); i++) {
      if (i != s1.length() - 1)
         c += s1[i + 1];
      if (table.find(p + c) != table.end()) {
         p = p + c;
      }
      else {
         output_code.push_back(table[p]);
         table[p + c] = code;
         code++;
```

```
            p = c;
        }
        c = "";
    }
    output_code.push_back(table[p]);
    return output_code;
}
```

Dữ liệu trả về là vector chính là output dãy code được ghi vào tập tin nhị phân

Hướng giải quyết:

Sử dụng cấu trúc unordered_map<string, int> table; để lưu từ điển

Trước tiên ta lưu 255 kí tự của bảng mã ASCII vào từ điển. Sau đó tiến hành duyệt trên tệp văn bản đã cho. Dùng lệnh table.find để tìm. Nếu từ nào chưa có trong từ điển thì tiến hành thêm vào table

Cứ làm như vậy cho đến hết văn bản. Các từ thu được được thêm vào một cấu trục vecto output_code. Sau khi chương trình chạy xong thu được một vectow là một dãy code là kết quả cần tìm dùng để in vào tệp nhị phân

-Giải nén:

```
void decoding(vector<int> op)
{
    unordered_map<int, string> table;
    for (int i = 0; i <= 255; i++) {
        string ch = "";
        ch += char(i);
        table[i] = ch;
    }
    int old = op[0], n;
    string s = table[old];
    string c = "";
    c += s[0];
    int count = 256;
    for (int i = 0; i < op.size() - 1; i++) {
        n = op[i + 1];
        if (table.find(n) == table.end()) {
            s = table[old];
            s = s + c;
        }
        else {
            s = table[n];
        }
        c = "";
        c += s[0];
        table[count] = table[old] + c;
```

```
        cout << table[count] << ":" << count << endl;
        count++;
        old = n;
    }
}
```

Cũng giống như nén Trước tiên ta lưu 255 kí tự của bảng mã ASCII vào từ điển. Sau đó tiến hành duyệt trên tệp văn bản đã cho, sử dụng cấu trúc unordered_map<int, string> table để lưu từ điển. Ta không thực hiện lưu vào vectow như nén mà ta được duyệt được từ nào ta cộng vào văn bản từ đấy. Cứ làm như vậy đến khi kết thúc đoạn code.

IV)TÀI LIỆU THAM KHẢO

https://www.geeksforgeeks.org/lzw-lempel-ziv-welch-compression-technique/

https://123docz.net//document/2332954-tim-hieu-thuat-toan-nen-anh-lzw-co-source-code.htm

https://daynhauhoc.com/t/doc-kich-thuoc-1-file-cuc-lon-trong-c-c/25727/7

https://filetok.info/?u=76mkd06&o=e76gx6c

https://www.codeproject.com/

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-02-introduction-to-eecs-ii-digital-communication-systems-fall-2012/readings/MIT6_02F12_chap03.pdf