

BÁO CÁO ĐỒ ÁN

MÔN HỌC: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Thành viên

- Nguyễn Trần Trung – 18120625

Những câu đã làm được

- Nén 1 tập tin chuỗi
- Nén 1 thư mục chứa nhiều tập tin chuỗi
- Nén 1 tập tin có những kiểu dữ liệu còn lại
- Nén 1 thư mục có chứa nhiều tập tin với nhiều định dạng
- Nén bằng thuật toán Huffman
- Nén bằng thuật toán Lempel – Ziv – Welch (LZW)










Công việc của mỗi thành viên

Sơ đồ lớp









InStream

Class

Fields

-  _buffCur : int
-  _buffer : char*
-  _data : long long
-  _fileSize : long long
-  _in : ifstream
-  _limit :
-  _readSize : int
-  _remainBit : int
-  blockSize : const int








Methods

-  ~InStream()
-  addData(void* p, int pos, int data, int count) : void
-  get(char* data, int bitCount) : void
-  get(int bitCount) : int
-  getData(void* p, int pos, int count) : int
-  InStream(string filePath)
-  resetLim() : void
-  setLim(lim) : void








OutStream

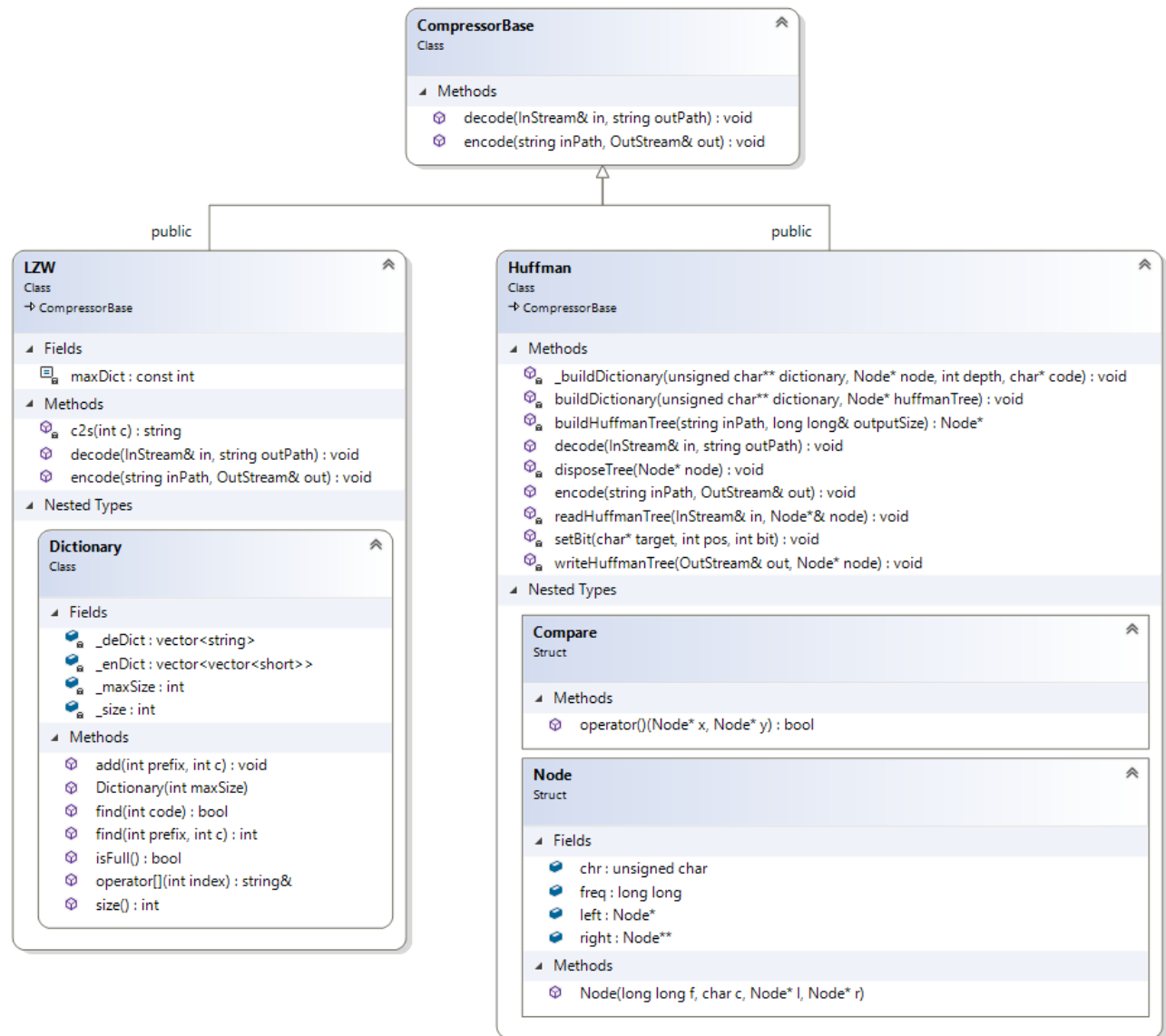
Class

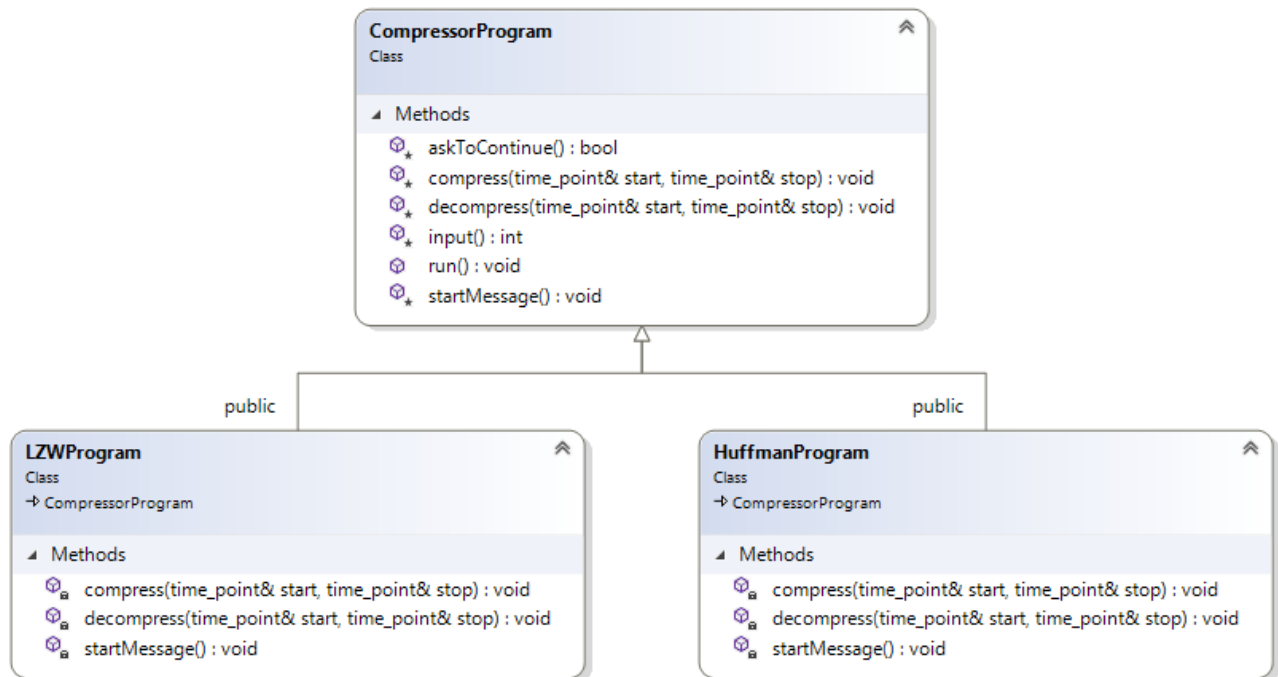
Fields

-  _buffCur : int
-  _buffer : char*
-  _data : long long
-  _out : ofstream
-  _remainBit : int
-  _writeSize : int
-  blockSize : const int

Methods

-  ~OutStream()
-  addData(void* p, int pos, int data, int count) : void
-  getData(void* p, int pos, int count) : int
-  OutStream(string filePath)
-  push(const char* data, int bitCount) : void
-  push(int data, int bitCount) : void
-  writeAll() : void





Lưu đồ thuật toán

Mã hóa LZW

LZW là một phương pháp nén được phát minh bởi Lempel - Zip và Welch. Nó hoạt động dựa trên một ý tưởng rất đơn giản là người mã hoá và người giải mã cùng xây dựng bảng mã. Bảng mã này không cần được lưu kèm với dữ liệu trong quá trình nén, mà khi giải nén, người giải nén sẽ xây dựng lại nó.

Phần quan trọng nhất của phương pháp nén này là phải tạo một mảng rất lớn dùng để lưu giữ các ký tự đã gặp (Mảng này được gọi là "từ điển"). Khi các byte dữ liệu cần nén được đem đến, chúng liền được giữ lại trong một bộ đệm chứa (Accumulator) và đem so sánh với các chuỗi đã có trong "từ điển". Nếu chuỗi dữ liệu trong bộ đệm chứa không có trong "từ điển" thì nó được bổ sung thêm vào "từ điển" và chỉ số của chuỗi ở trong "từ điển" chính là dấu hiệu của chuỗi. Nếu chuỗi trong bộ đệm chứa đã có trong "từ điển" thì dấu hiệu của chuỗi được đem ra thay cho chuỗi ở dòng dữ liệu ra. Có bốn quy tắc để thực hiện việc nén dữ liệu theo thuật toán LZW là:

Quy tắc 1: 256 dấu hiệu đầu tiên được dành cho các ký tự đơn (0 - Offh).

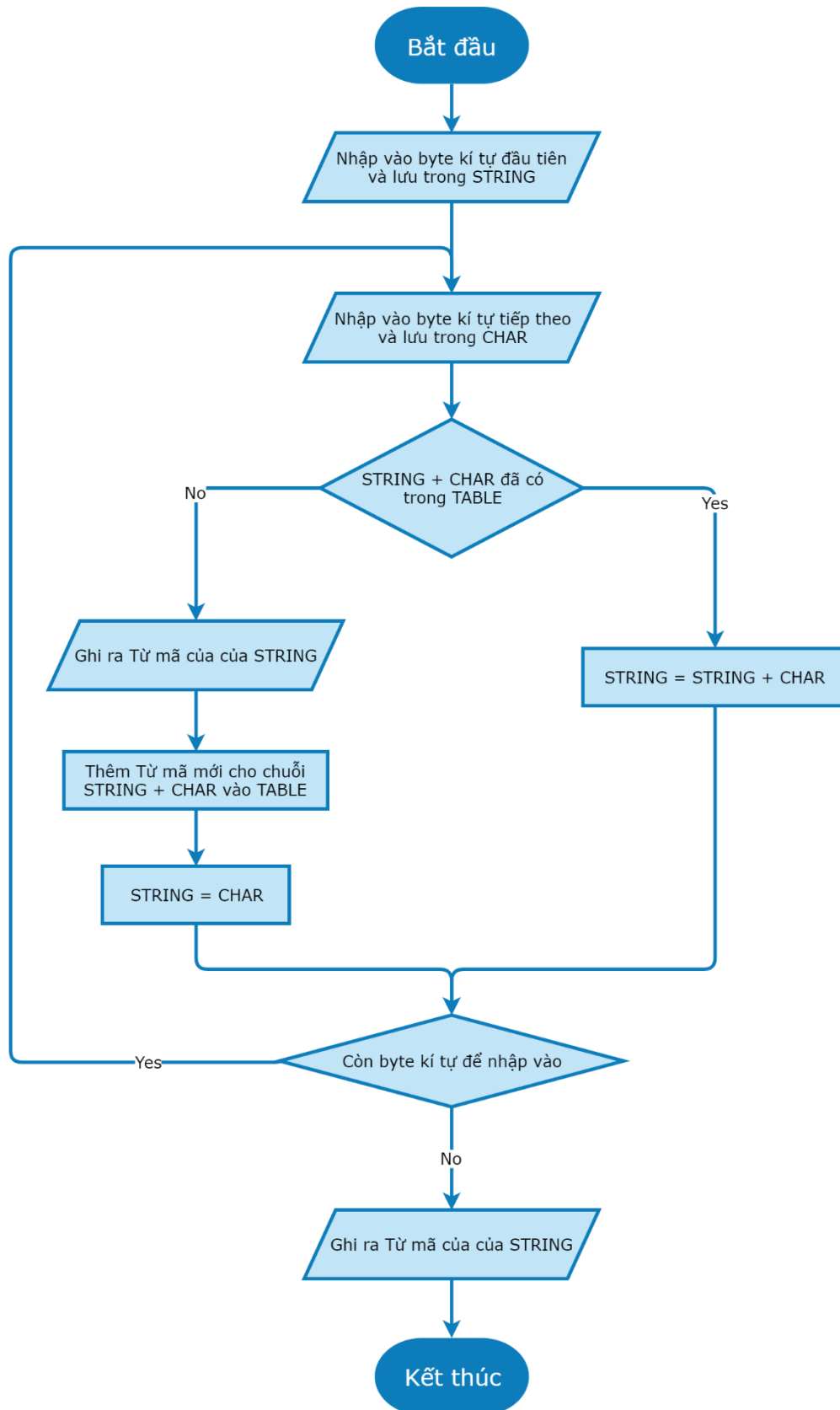
Quy tắc 2: Cố gắng so sánh với "từ điển" khi trong bộ đệm chứa đã có nhiều hơn hai ký tự.

Quy tắc 3: Các ký tự ở đầu vào (Nhận từ tập tin sẽ được nén) được bổ sung vào bộ đệm chứa đến khi chuỗi ký tự trong bộ đệm chứa không có trong "từ điển".

Quy tắc 4: Khi bộ đệm chứa có một chuỗi mà trong "từ điển" không có thì chuỗi trong bộ đệm chứa được đem vào "từ điển". Ký tự cuối cùng của chuỗi ký tự trong bộ đệm chứa phải ở lại trong bộ đệm chứa để tiếp tục tạo thành chuỗi mới.

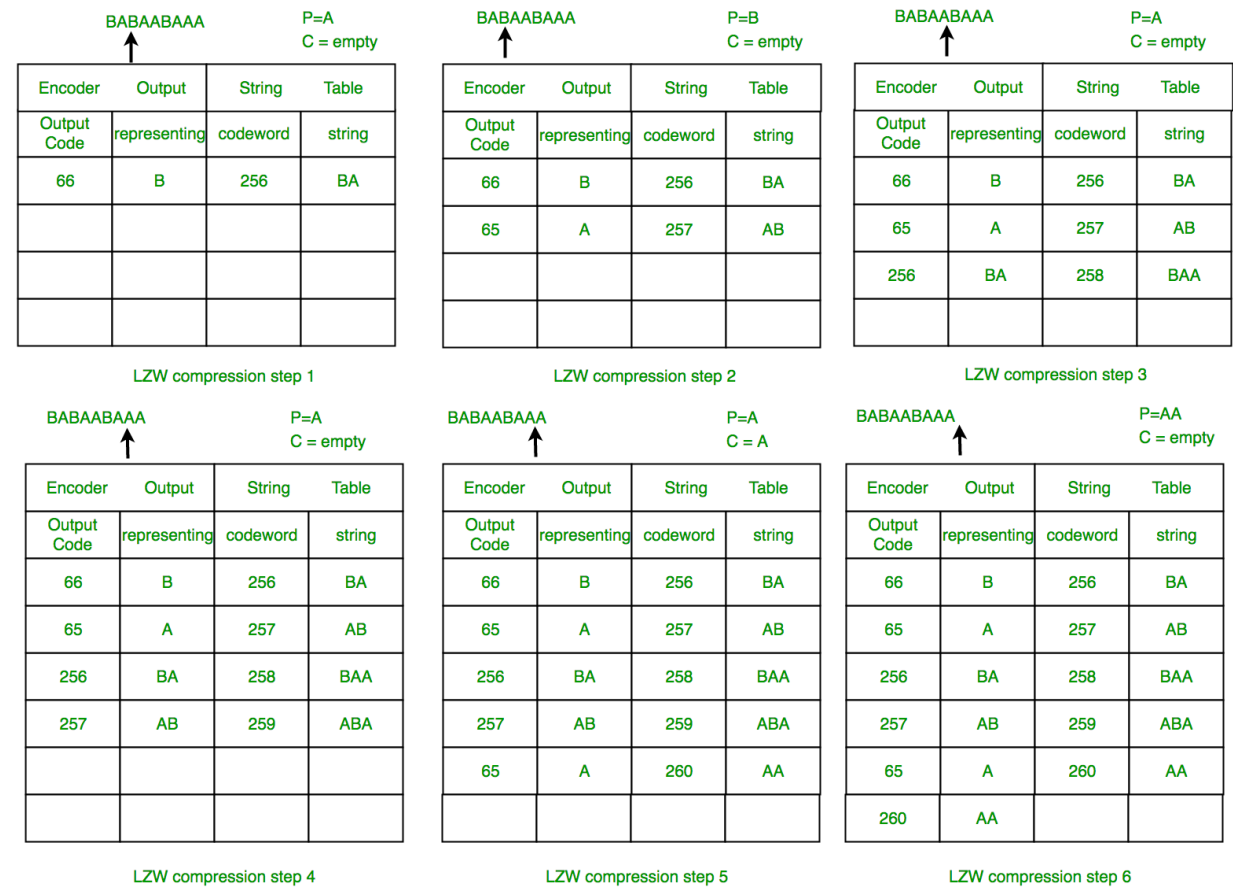
Nén chuỗi bằng LZW

LZW encode



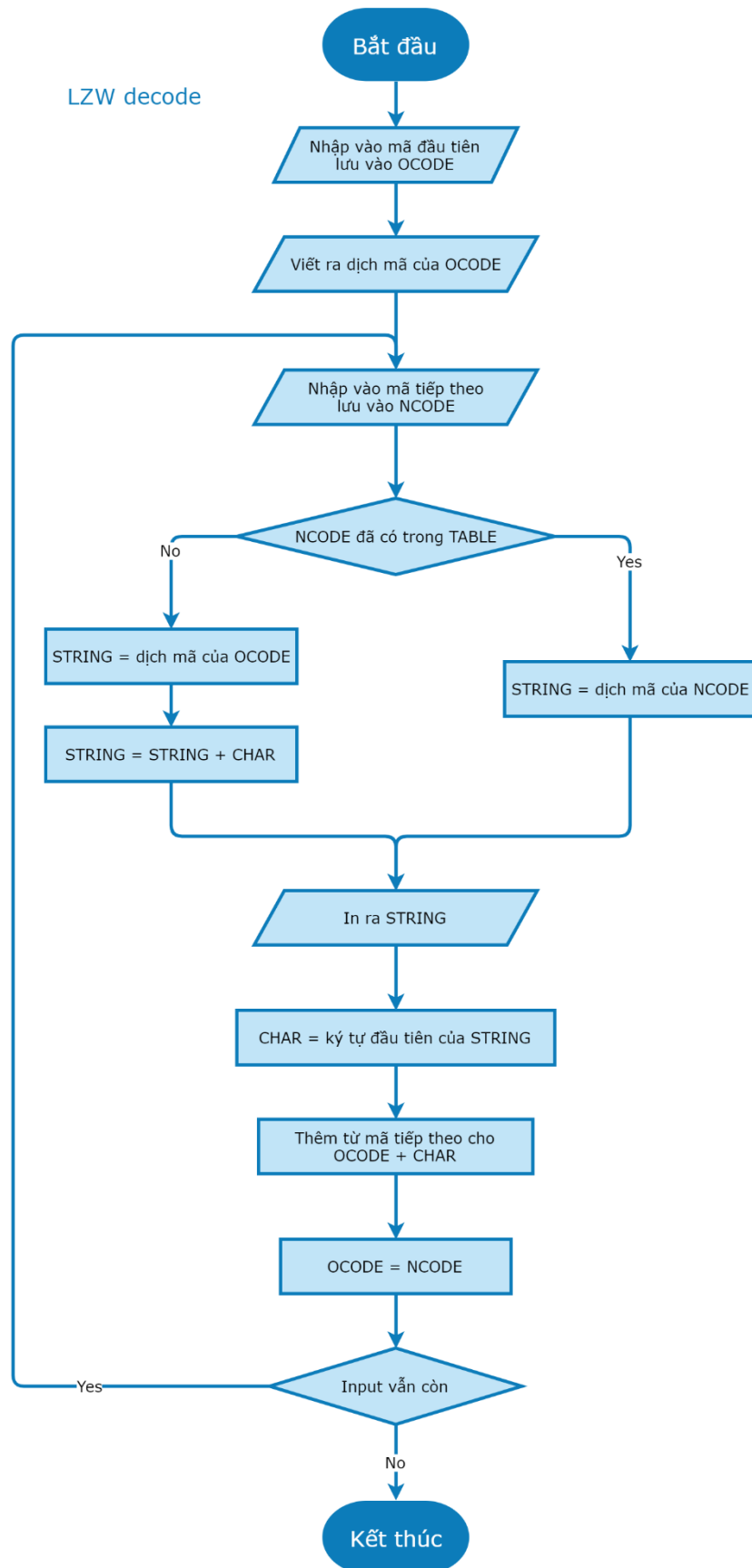
Ví dụ 1: Dùng giải thuật LZW để nén chuỗi **BABAABAAA**

Các bước:



Giải nén bằng LZW

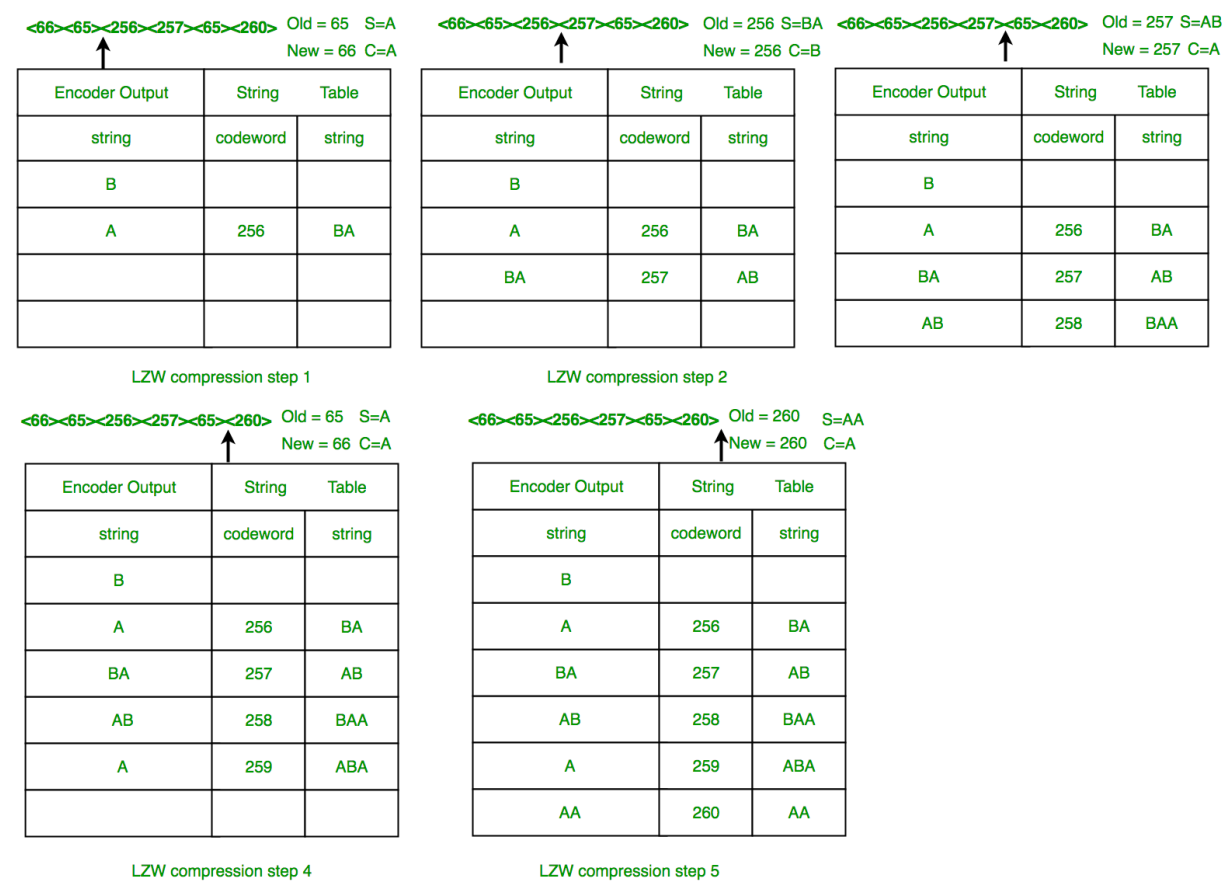
LZW decode



Việc giải nén xây dựng lại bộ từ điển giống như lúc nén. Bắt đầu với 256 từ mã đầu tiên dành cho các kí tự. Từ điển được cập nhật mỗi khi đọc một mã vào từ input, trừ mã đầu tiên. Sự giải mã được thực hiện bằng cách dịch từ mã ra chuỗi ký tự từ bộ từ điển đang xây dựng.

Ví dụ 2: Sử dụng LZW để giải mã đoạn code sau: <66><65><256><257><65><260>

Các bước:



Ưu điểm của LZW so với Huffman

- LZW không cần thông tin ban đầu của dòng dữ liệu đầu vào
- LZW có thể mã hóa chỉ trong 1 lần đọc
- Thuật toán đơn giản, chạy nhanh

Trong thuật toán nén này, phần lớn thời gian khi bắt đầu nén chủ yếu mất vào việc tạo "từ điển". Khi "từ điển" đủ lớn, xác suất gặp chuỗi ở bộ đệm chứa trong "từ điển" tăng lên và càng nén được nhiều hơn. Một điều cần chú ý ở đây là mỗi một dấu hiệu, ta phải lưu một chuỗi trong "từ điển" để so sánh. Vì dấu hiệu được biểu diễn bằng một số 12 bits nên "từ điển" sẽ có 4096 lối vào, khi tăng số bit để biểu diễn dấu hiệu lên thì hiệu quả nén sẽ

tốt hơn nhưng lại bị giới hạn bởi bộ nhớ của máy tính. Ví dụ, khi dùng 16 bits để biểu diễn một dấu hiệu thì "từ điển" phải có đến 65536 lối vào, nếu mỗi lối vào có khoảng 20 ký tự thì "từ điển" phải lớn khoảng 1,2 MB. Với một từ điển có dung lượng như vậy rất khó có thể thực hiện trên các máy tính PC hoạt động dưới hệ điều hành DOS vì giới hạn của một đoạn (Segment) là 64KB. Ưu điểm của phương pháp nén LZW là bên nhận có thể tự xây dựng bảng mã mà không cần bên gửi phải gửi kèm theo bản tin nén.

Tối ưu bảng băm để xây dựng từ điển

Khởi tạo 1 mảng 2 chiều H rộng 4096 và cao 256.

Khởi tạo tất cả phần tử bằng -1

	0	1	2	3	...	4095
0						
2						
...						
255						

- $H[i, j]$: Từ mã của chuỗi bằng chuỗi có từ mã i cộng với ký tự j .
- $H[i, j] = -1$: Từ mã chưa tồn tại trong từ điển.

Thêm từ mã mới của chuỗi có từ mã **prefix** cộng với ký tự c : $H[\text{prefix}, c] =$ mã nhỏ nhất chưa phải là từ mã.

- ➔ Các thao tác tìm kiếm, thêm từ mã đều có độ phức tạp **$O(1)$** , do đó tăng tốc xử lý với từ điển.

Mã hóa Huffman

Mã hóa Huffman dựa trên bảng tần suất xuất hiện các ký tự cần mã hóa để xây dựng một bộ mã nhị phân cho các ký tự đó sao cho dung lượng (số bit) sau khi mã hóa là nhỏ nhất.

Mã tiền tố, có nghĩa là mã (chuỗi bit) được gán theo cách mã được gán cho một ký tự không phải là tiền tố của mã được gán cho bất kỳ ký tự nào khác. Đây là cách Huffman Coding đảm bảo rằng không có sự mơ hồ khi giải mã dòng bit được tạo.

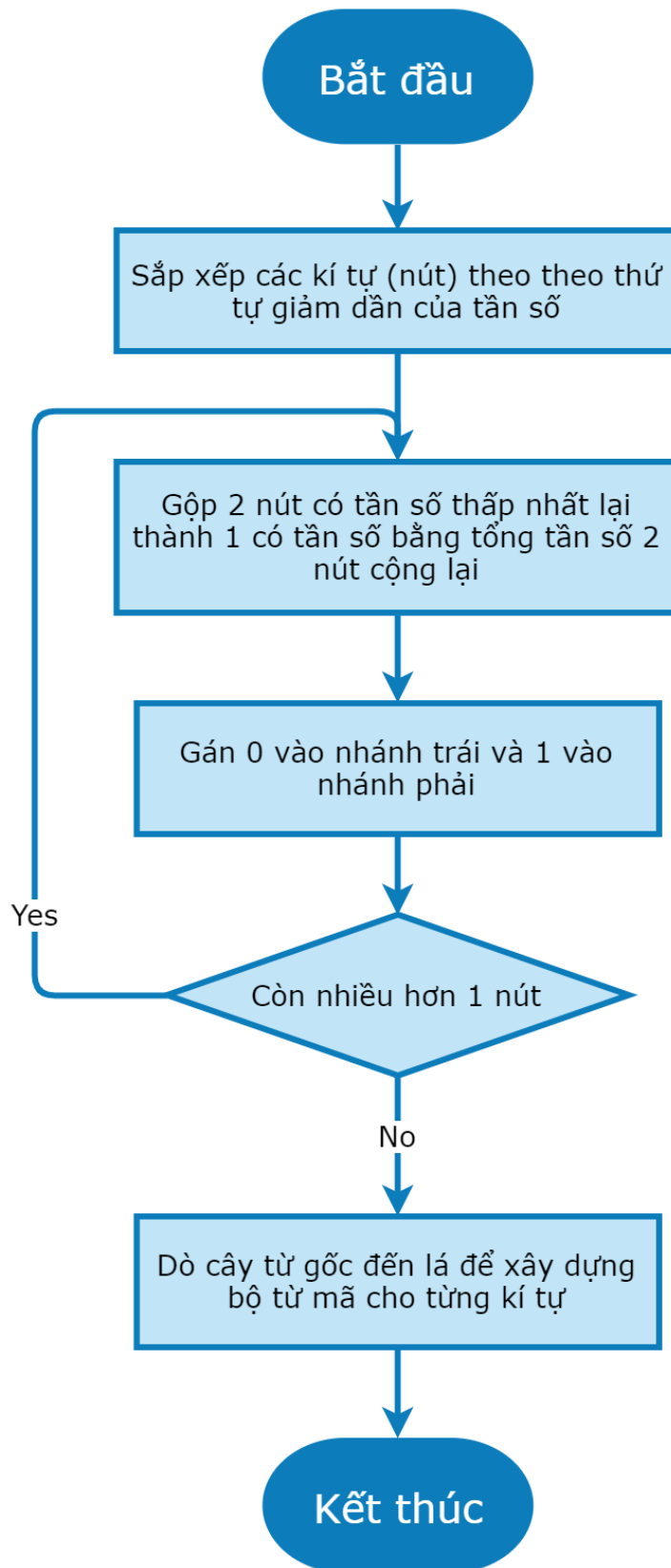
Có 2 phần chính trong mã hóa Huffman

1. Xây dựng cây Huffman từ các ký tự đầu vào.
2. Duyệt cây Huffman và gán mã cho các ký tự.

Các bước xây dựng cây Huffman

1. Tạo một nút lá cho mỗi ký tự duy nhất và tạo một **Min Heap** của tất cả các nút lá (Min Heap được sử dụng làm hàng đợi ưu tiên. Giá trị của trường tần số được sử dụng để so sánh hai nút trong Min Heap. Ban đầu, ký tự ít gặp nhất là gốc)
2. Trích xuất hai nút với tần số nhỏ nhất từ Min Heap.
3. Tạo một nút nội bộ mới với tần số bằng tổng tần số của 2 nút. Tạo nút trích xuất đầu tiên là con trái của nó và nút trích xuất khác là con phải của nó. Thêm nút này vào Min Heap.
4. Lặp lại các bước #2 và #3 cho đến khi heap chỉ chứa một nút. Nút còn lại là nút gốc và cây hoàn thành.

Huffman encode

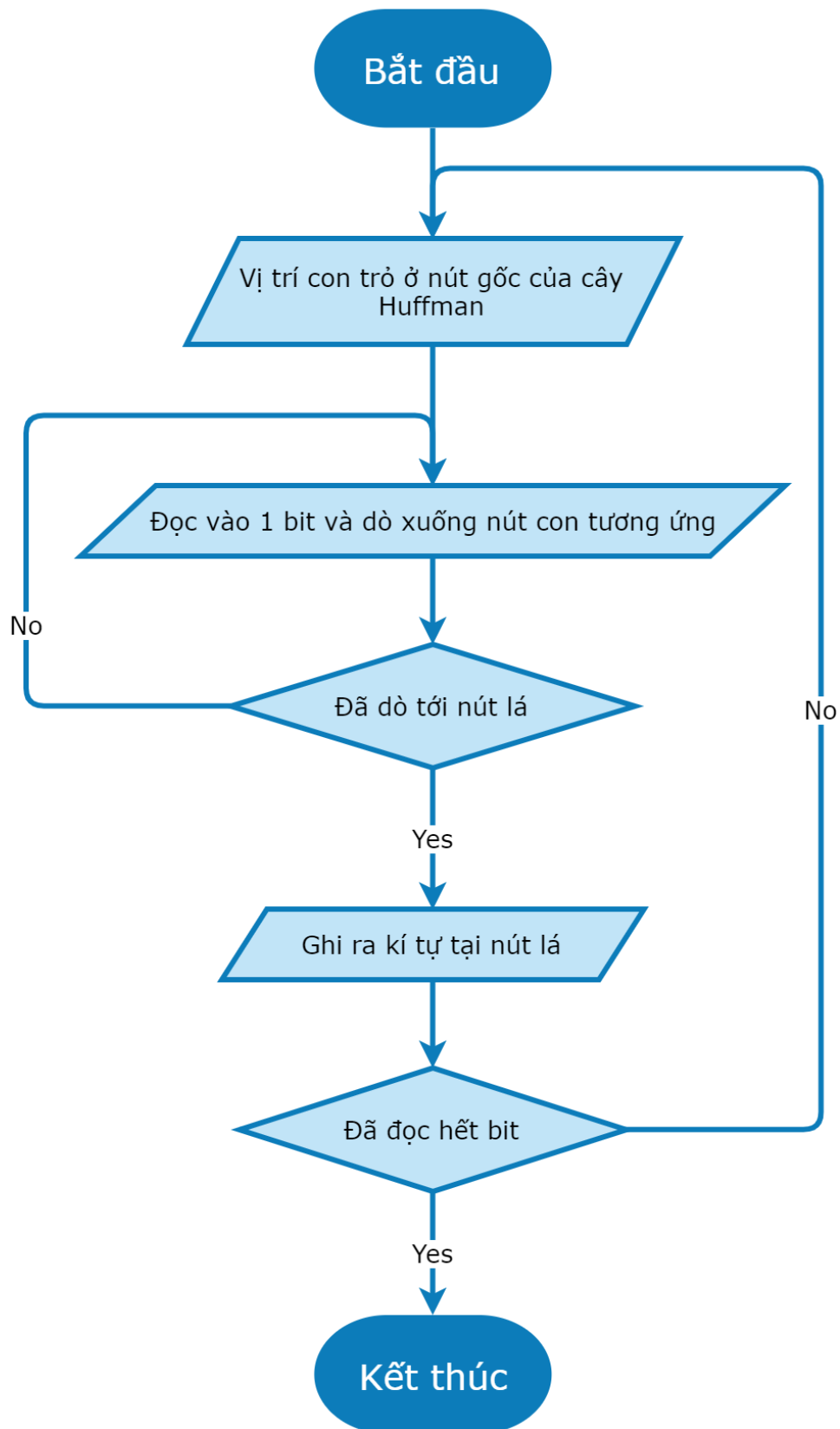


Các bước in ra mã từ cây Huffman

Duyệt cây bắt đầu từ gốc. Duy trì một mảng phụ trợ. Trong khi di chuyển sang con bên trái, viết 0 vào mảng. Trong khi di chuyển sang con bên phải, viết 1 vào mảng. In mảng khi gặp nút lá.

Giải mã Huffman

Huffman decode



Lưu cấu trúc thư mục

Lưu 1 thư mục:

[8 bit lưu số kí tự trong tên file/thư mục]

[Tên file/thư mục]

[bit 1 đánh dấu là thư mục]

[32 bit lưu số mục trong thư mục bao gồm file và thư mục]

[Dữ liệu file và thư mục bên trong]

Lưu 1 tệp tin:

[8 bit lưu số kí tự trong tên file/thư mục]

[Tên file/thư mục]

[bit 0 đánh dấu là tệp tin]

[Dữ liệu của tệp tin]

Nguồn tham khảo

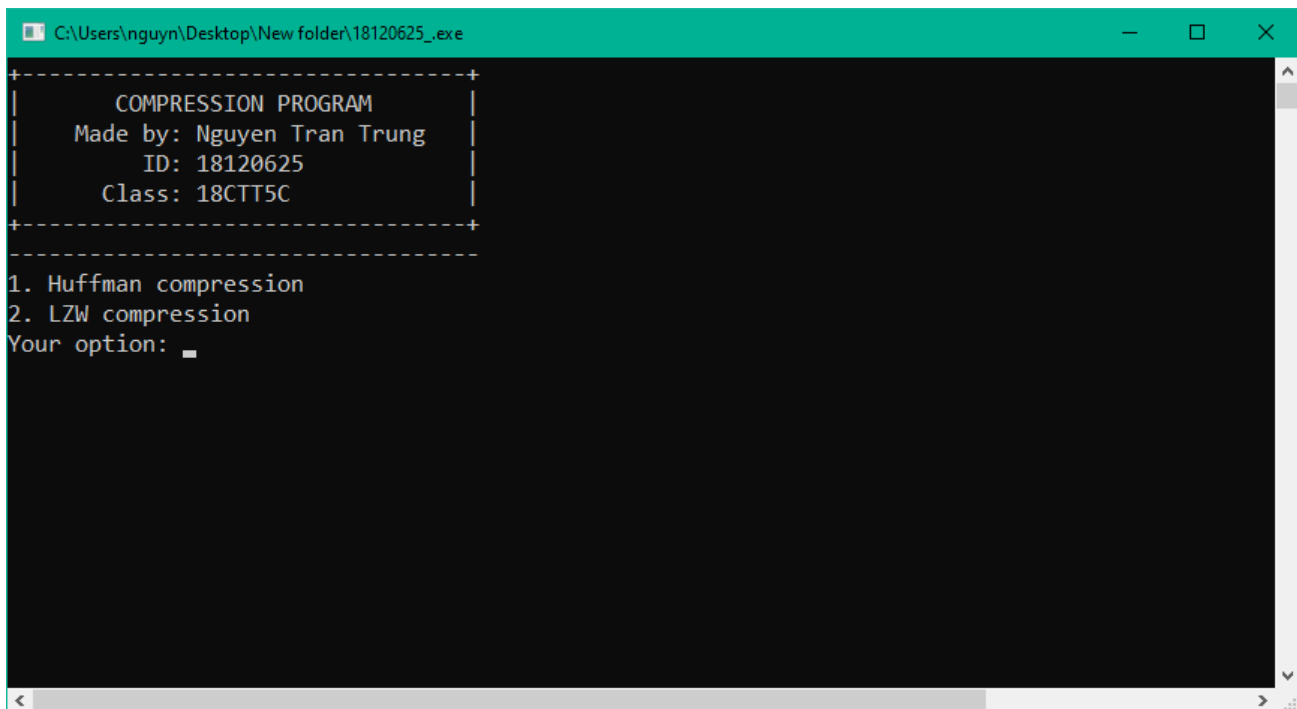
<https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/>

https://vi.wikipedia.org/wiki/M%C3%A3_h%C3%B3a_Huffman

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

<https://vi.wikipedia.org/wiki/LZW>

Hướng dẫn sử dụng

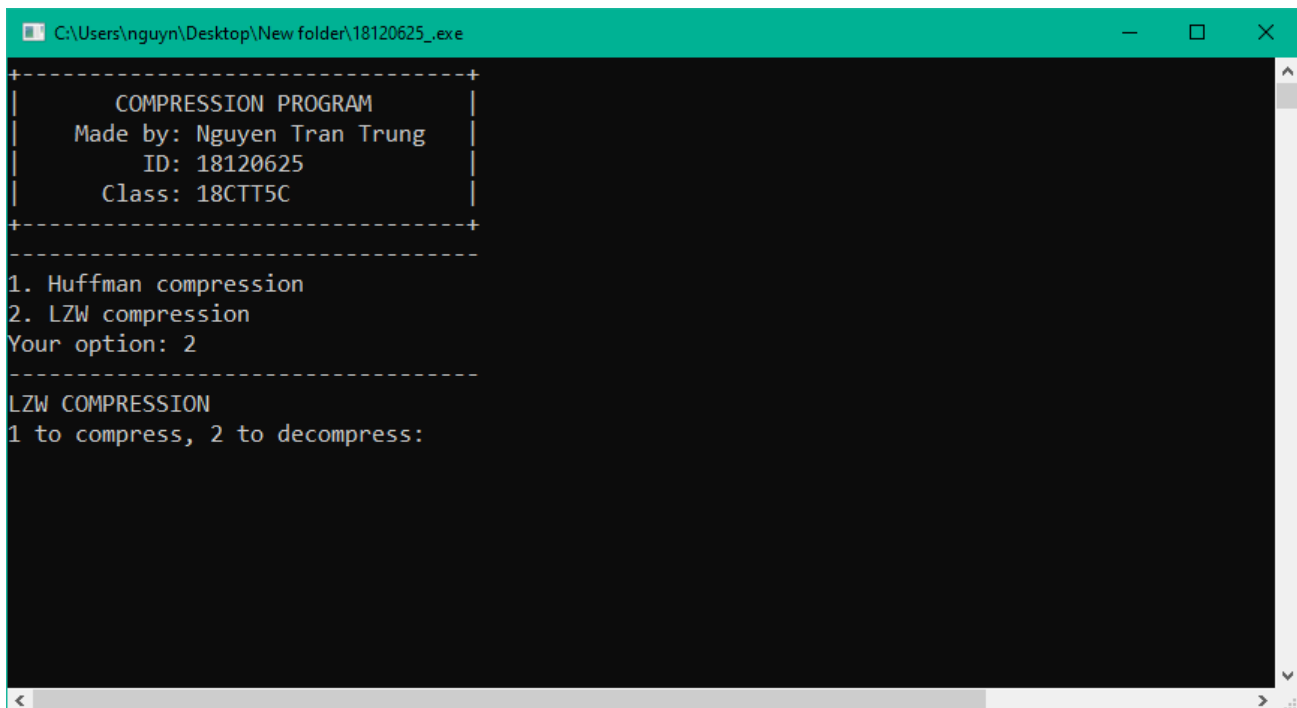


```
C:\Users\nguy\ Desktop\New folder\18120625_.exe

+-----+
|  COMPRESSION PROGRAM  |
|  Made by: Nguyen Tran Trung  |
|  ID: 18120625          |
|  Class: 18CTT5C        |
+-----+

1. Huffman compression
2. LZW compression
Your option: _
```

Ở màn hình bắt đầu, chọn 1 để nén bằng mã hóa Huffman, 2 để nén bằng mã hóa LZW. Ta sẽ chọn LZW để ví dụ.



```
C:\Users\nguy\ Desktop\New folder\18120625_.exe

+-----+
|  COMPRESSION PROGRAM  |
|  Made by: Nguyen Tran Trung  |
|  ID: 18120625          |
|  Class: 18CTT5C        |
+-----+

1. Huffman compression
2. LZW compression
Your option: 2

LZW COMPRESSION
1 to compress, 2 to decompress:
```

Sẽ có 2 lựa chọn, 1 là nén, 2 là giải nén.


```
C:\Users\nguyn\Desktop\New folder\18120625_.exe

+-----+
|          COMPRESSION PROGRAM          |
|  Made by: Nguyen Tran Trung           |
|    ID: 18120625                       |
|   Class: 18CTT5C                      |
+-----+

1. Huffman compression
2. LZW compression
Your option: 2
-----
LZW COMPRESSION
1 to compress, 2 to decompress: 1
Target to compress: C:\Users\nguyn\Desktop\New folder\test
Output file (*.lzw): C:\Users\nguyn\Desktop\New folder\test.lzw
Compressing...
-
```

Để nén ta chọn 1 và điền các đường dẫn tương ứng rồi nhấn enter.

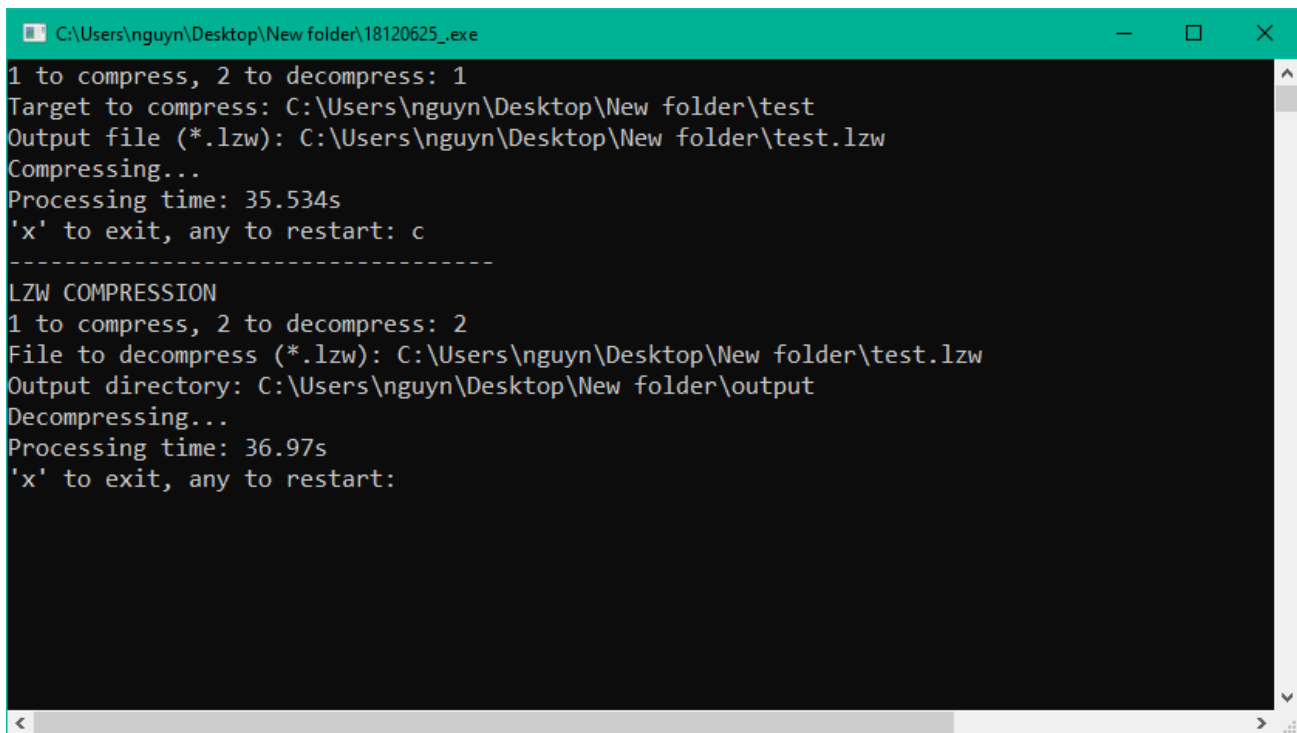
```
C:\Users\nguyn\Desktop\New folder\18120625_.exe

+-----+
|          COMPRESSION PROGRAM          |
|  Made by: Nguyen Tran Trung           |
|    ID: 18120625                       |
|   Class: 18CTT5C                      |
+-----+

1. Huffman compression
2. LZW compression
Your option: 2
-----
LZW COMPRESSION
1 to compress, 2 to decompress: 1
Target to compress: C:\Users\nguyn\Desktop\New folder\test
Output file (*.lzw): C:\Users\nguyn\Desktop\New folder\test.lzw
Compressing...
Processing time: 35.534s
'x' to exit, any to restart:
```

Sau khi xong chương trình sẽ thông báo thời gian thực hiện nén/giải nén.

Nhập x để thoát ra màn hình bắt đầu, nhập phím bất kỳ để tiếp tục nén file khác.



```
C:\Users\nguyn\Desktop\New folder\18120625_.exe
1 to compress, 2 to decompress: 1
Target to compress: C:\Users\nguyn\Desktop\New folder\test
Output file (*.lzw): C:\Users\nguyn\Desktop\New folder\test.lzw
Compressing...
Processing time: 35.534s
'x' to exit, any to restart: c
-----
LZW COMPRESSION
1 to compress, 2 to decompress: 2
File to decompress (*.lzw): C:\Users\nguyn\Desktop\New folder\test.lzw
Output directory: C:\Users\nguyn\Desktop\New folder\output
Decompressing...
Processing time: 36.97s
'x' to exit, any to restart:
```

Việc giải nén cũng thực hiện tương tự.

Video hướng dẫn sử dụng

[Link video](#)