**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN LẬP TRÌNH TÍNH TOÁN**

**TÊN ĐỀ TÀI :Lập trình nén và giải nén dữ liệu tiếng Việt sử dụng mã hóa Huffman**

Người hướng dẫn**: PHẠM MINH TUẤN**

Nhóm sinh viên thực hiện**:**

**Phan Đức Thịnh LỚP: 19TCLC\_NHAT1 NHÓM: YYY**

**Nguyễn Minh Quân LỚP: 19TCLC\_NHAT1 NHÓM: YYY**

**Đà Nẵng, 12/2020**

MỤC LỤC

[MỤC LỤC i](#_Toc59624950)

[DANH MỤC HÌNH VẼ iii](#_Toc59624951)

[LỜI MỞ ĐẦU 1](#_Toc59624952)

[BẢNG PHÂN CÔNG NHIỆM VỤ 2](#_Toc59624953)

[1. TỔNG QUAN ĐỀ TÀI 3](#_Toc59624954)

[2. CƠ SỞ LÝ THUYẾT 3](#_Toc59624955)

[2.1. Ý tưởng 3](#_Toc59624956)

[2.2. Cơ sở lý thuyết[5] 3](#_Toc59624957)

[2.2.1. Mã tiền tố 3](#_Toc59624958)

[2.2.2. Dữ liệu 4](#_Toc59624959)

[2.2.3. Hàng đợi ưu tiên 4](#_Toc59624960)

[3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN 4](#_Toc59624961)

[3.1. Phát biểu bài toán 4](#_Toc59624962)

[3.2. Cấu trúc dữ liệu 5](#_Toc59624963)

[3.2.1. File[4] 5](#_Toc59624964)

[3.2.2. Struct[3] 5](#_Toc59624965)

[3.2.3. Containers[1] 5](#_Toc59624966)

[3.3. Thuật toán 6](#_Toc59624967)

[3.3.1. Thuật toán nén Huffman:[2] 6](#_Toc59624968)

[3.3.2. Giải nén Huffman: 6](#_Toc59624969)

[4. CHƯƠNG TRÌNH VÀ KẾT QUẢ 7](#_Toc59624970)

[4.1. Tổ chức chương trình 7](#_Toc59624971)

[4.1.1. Nén:[2] 7](#_Toc59624972)

[4.1.2. Giải nén: 7](#_Toc59624973)

[4.2. Ngôn ngữ cài đặt 7](#_Toc59624974)

[4.3. Kết quả 8](#_Toc59624975)

[4.3.1. Giao diện chính của chương trình 8](#_Toc59624976)

[4.3.2. Kết quả thực thi của chương trình 8](#_Toc59624977)

[4.3.3. Nhận xét đánh giá 9](#_Toc59624978)

[5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 9](#_Toc59624979)

[5.1. Kết luận 9](#_Toc59624980)

[5.2. Hướng phát triển 10](#_Toc59624981)

[TÀI LIỆU THAM KHẢO 11](#_Toc59624982)

[PHỤ LỤC 12](#_Toc59624983)

DANH MỤC HÌNH VẼ

[Hình 1: Giao diện chính của chương trình 8](#_Toc59621775)

[Hình 2: Kết quả của nén 8](#_Toc59621776)

[Hình 3: Kết quả của giải nén 9](#_Toc59621777)

LỜI MỞ ĐẦU

Trong phần này, cần trình bày về: Mục đích thực hiện đề tài, mục tiêu đề tài, phạm vi và đối tượng nghiên cứu, phương pháp nghiên cứu, cấu trúc của đồ môn học}

Mạng internet là xu hướng phát triển thế giới ngày nay. Hiện nay, Internet đã trở nên rất phổ biến trên toàn thế giới. Thông qua mạng Internet mọi người có thể trao đổi thông tin với một người hay một tổ chức n đó thông qua mạng máy tính thì yêu cầu quan trọng là làm sao để đảm bảo thông tin không bị sai lệch hoặc bị lộ do sự xâm nhập của kẻ thứ ba.

Trước các yêu cầu cần thiết đó, một số giải thuật mã hóa đã được xây dựng nhằm đảm bảo tính an toàn dữ liệu tại nơi lưu trữ cũng như khi dữ liệu được truyền trên mạng như giải thuật Huffman. Việc tìm hiểu và xây dựng chương trình giải thuật này cũng không nằm ngoài mục đích của đồ án môn học này. Đồ án có nhiệm vụ tìm hiểu lý thuyết về mã hóa thông tin.

Thuật toán Huffman, đây được coi là thuật toán mã hóa cơ bản nhất, từ đó chúng ta sẽ có cách nhìn về các phương pháp mã hóa dữ liệu và các ứng dụng của nó

BẢNG PHÂN CÔNG NHIỆM VỤ

|  |  |
| --- | --- |
| Phan Đức Thịnh | Đọc tài liệu, cài đặt phần nén Huffman, viết báo cáo, làm silde. |
| Nguyễn Minh Quân | Đọc tài liệu, viết báo cáo cài đặt về thuật toán giải nén Huffman và thuật toán chuyển đổi tiếng việt. |

# TỔNG QUAN ĐỀ TÀI

Trong thời buổi công nghệ phát triển như hiện nay, thông tin là phương tiện vô cùng quan trọng trong đời sống chúng ta. Và càng nhiều thông tin chúng ta càng tốn nhiều dữ liệu để lưu trữ thông tin đó. Vì vậy, thuật toán Huffman là một thuật toán mã hóa thông tin dùng để nén dữ liệu dựa trên việc tối ưu hóa việc mã hóa các ký tự trong chuỗi ban đầu bằng việc xây dựng bộ mã nhị phân đại diện cho từng ký tự. Thuật toán có mục tiêu sẽ xây dựng được bảng mã nhị phân đại diện cho từng ký tự sao cho những ký tự có tần suất xuất hiện nhiều sẽ có mã nhị phân đại diện ngắn nhất và ngược lại.

Huffman được sử dụng rộng rãi trong tất cả các định dạng nén chính mà bạn có thể gặp ví dụ như: GZIP, PKZIP (winzip, ...) và BZIP2, hoặc đến các định dạng hình ảnh như JPEG và PNG...

Giới thiệu chung về bài toán, kèm theo liệt kê một vài lĩnh vực ứng dụng cụ thể.

# CƠ SỞ LÝ THUYẾT

## Ý tưởng

Ý tưởng chính của bài toán là việc thay vì biểu diễn từng ký tự trong một 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 ký tự trong chuỗi sang một 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 một chuỗi nhị phân có kích thước ngắn hơn

## Cơ sở lý thuyết[5]

### Mã tiền tố

Mã tiền tố là bộ các từ mã của một tập hợp các ký hiệu sao cho từ mã của mỗi ký hiệu không là tiền tố (phần đầu) của từ mã một ký hiệu khác trong bộ mã ấy.

Đương nhiên mã hóa với độ dài không đổi là mã tiền tố.

Ví dụ: Giả sử mã hóa từ "ARRAY", tập các ký hiệu cần mã hóa gồm 3 chữ cái "A","R","Y".

Nếu mã hóa bằng các từ mã có độ dài bằng nhau ta dùng ít nhất 2 bit cho một chữ cái chẳng hạn "A"=00, "R"=01, "Y"=10. Khi đó mã hóa của cả từ là 0001010010. Để giải mã ta đọc hai bit một và đối chiếu với bảng mã.

Nếu mã hóa "A"=0, "R"=01, "Y"=11 thì bộ từ mã này không là mã tiền tố vì từ mã của "A" là tiền tố của từ mã của "R". Để mã hóa cả từ ARRAY phải đặt dấu ngăn cách vào giữa các từ mã 0,01,01,0,11

Nếu mã hóa "A"=0, "R"=10, "Y"=11 thì bộ mã này là mã tiền tố. Với bộ mã tiền tố này khi mã hóa xâu "ARRAY" ta có 01010011.

### Dữ liệu

* Input:

Bảng n chữ cái A = {a1,a2, ..., an}

Tập các trọng số tương ứng W = {w1, w2, ..., wn}

* Output:

Đặt L(C) =

### Hàng đợi ưu tiên

Trong mỗi bước của thuật toán xây dựng cây Huffman, ta luôn phải chọn ra hai gốc có trọng số nhỏ nhất. Để làm việc này ta sắp xếp các gốc vào một hàng đợi ưu tiên theo tiêu chuẩn trọng số nhỏ nhất. Một trong các cấu trúc dữ liệu thuận lợi cho tiêu chuẩn này là cấu trúc đống (minheap) (với phần tử có trọng số nhỏ nhất nằm trên đỉnh của đống).

# TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

## Phát biểu bài toán

**Bài toán nén :**

* Vấn đề: Cần nén lại dữ liệu gốc thành dãy nhị phân dựa trên bảng tần suất xuất hiện các kí tự cần mã hóa để xây dựng một bộ mã nhị phân cho các kí tự đó sao cho dung lượng (số bit) sau khi mã hóa là nhỏ nhất.
* Đầu vào: File dữ liệu gốc.
* Đầu ra: Dữ liệu đã được nén(encode string).
* Yêu cầu: dữ liệu đã nén giống và có dung lượng (số bit) ít hơn dữ liệu gốc ban đầu.

**Bài toán giải nén :**

* Vấn đề: Cần khôi phục lại dữ liệu gốc từ dãy nhị phân đã được nén dựa trên mã bit của từng ký tự có trong văn bản(kể cả viết hoa viết thường). Dữ liệu đã được giải nén ghi vào file ouput.
* Đầu vào: Dãy nhị phân đã mã hoá của 1 tệp xác định (encode string), mã của từng ký tự được nén Huffman(bitcode).
* Đầu ra: Dữ liệu đã được giải nén(decode string).
* Yêu cầu: dữ liệu đã giải nén giống với dữ liệu ban đầu chưa giải nén.

## Cấu trúc dữ liệu

### File[4]

* File là kiểu đối tượng, nó xác định một stream và chứa các thông tin cần thiết để điều khiển, bao gồm một con trỏ trỏ đến buffer của nó, các chỉ mục và trạng thái của nó.
* Các bạn có thể hiểu File (trong ngôn ngữ lập trình C/C++) là một kiểu đối tượng mà thông qua nó chúng ta có thể thao tác với dữ liệu được lưu trữ bên trong File (chứ không phải là một File trên máy tính).

Để các bạn không bị nhầm lẫn, mình đang nói về kiểu dữ liệu FILE được định nghĩa trong thư viện cstdio (hay stdio.h) mà có thể các bạn đã từng học trong ngôn ngữ C. Chúng ta sẽ học cách sử dụng các Stream để thao tác với file thay vì sử dụng kiểu dữ liệu FILE trong các bài học sau, nhưng mình nghĩ kiểu dữ liệu FILE trong thư viện cstdio cũng có những ưu điểm riêng của nó nên mình không bỏ qua bài học này.

### Struct[3]

* Các mảng trong C/C++ cho phép bạn định nghĩa một vài loại biến có thể giữ giá trị của một vài thành viên cùng kiểu dữ liêu. Nhưng **structure - cấu trúc** là một loại dữ liệu khác trong ngôn ngữ lập trình C/C++, cho phép bạn kết hợp các dữ liệu khác kiểu nhau.

### Containers[1]

* Một container là một đối tượng cụ thể lưu trữ một tập các đối tượng khác (các phần tử của nó). Nó được thực hiện như các lớp mẫu (class templates)
* Containers quản lý không gian lưu trữ cho các phần tử của nó và cung cấp các hàm thành viên để truy cập tới chúng hoặc trực tiếp hoặc thông qua các biến lặp (interator – giống như con trỏ).
* Container xây dựng các cấu trúc thuờng sử dụng trong lập trình như: mảng động - dynamic arrays (vector), hàng đợi – queues (queue), hàng đợi ưu tiên – heaps (priority queue), danh sách kiên kết – linked list (list), cây – trees (set), mảng ánh xạ - associative arrays (map),...
* Nhiều container chứa một số hàm thành viên giống nhau. Quyết định sử dụng loại container nào cho nhu cầu cụ thể nói chung không chỉ phụ thuộc vào các hàm được cung cấp mà còn phải dựa vào hiệu quả của các hàm thành viên của nó (độ phức tạp (từ giờ mình sẽ viết tắt là ĐPT) của các hàm). Điều này đặc biệt đúng với container dãy (sequence containers), mà trong đó có sự khác nhau về độ phức tạp đối với các thao tác chèn/xóa phần tử hay truy cập vào phần tử.

### String[6]

* String là kiểu dữ liệu mới được định nghĩa sẵn trong C++, nó có nhiều ưu điểm vượt trội và giúp tránh được những phiền phức so với chuỗi kiểu char\* của C...

## Thuật toán

### Thuật toán Dijkstra:[2]

Mô tả về giải thuật Dijkstra:

* **Bước 1**: Chọn S = {} là tập các soure\_node bao gồm current\_node và passed\_node . Với current\_node là node đang được xét đến, passed\_node là các node đã được xét. current\_node đầu tiên sẽ là node đích của bài toán tìm đường đi ngắn nhất.
* **Bước 2**: Khởi tạo giải thuật với current\_node là node đích và cost(N) là giá trị của đường đi ngắn nhất từ N đến node đích.
* **Bước 3**: Xét các node kề N với current\_node . Gọi d(current\_node,N) là khoảng cách giữa node kề N và current\_node . Với p = d(current\_node,N) + cost (current\_node). Nếu p < cost(N) thì cost(N) = p . Nếu không thì cost(N) giữ nguyên giá trị .
* **Bước 4**: Sau khi xét hết các node kề N, đánh dấu current\_node thành passed\_node .
* **Bước 5**: Tìm current\_node mới với 2 điều kiện: không phải passed\_node và cost(current\_node) là nhỏ nhất
* **Bước 6**: Nếu tập S = {} chứa đủ các node của đồ thị thì dừng thuật toán. Nếu không thì quay trở lại **Bước 3** .

### Giải nén Huffman:

* Xây dựng Huffman ngược*:* đọc từng ký tự và mã của ký tự đó sau đó xây dựng cây từ rễ(root) nếu bit là 0 thì sang trái còn 1 thì sang phải , nếu không còn bit nào trong mã thì đánh dấu lá và gán ký tự cho nội dung của lá
* Giải nén Huffman*:* đọc từng bit của dãy nhị phân đã mã hóa và tiến hành duyệt cây đến khi đạt nút lá thì ngắt và thêm kí tự tại nút lá vào dãy dữ liệu đã giải nén.

# CHƯƠNG TRÌNH VÀ KẾT QUẢ

## Tổ chức chương trình

### Nén:[2]

**B1:** Duyệt file, thống kê tần suất xuất hiện của từng ký tự.(void calcFreq(string str,int n))

**B2:** Xây dựng cây Huffman dựa vào bảng thống kê. (void HuffmanCodes(int size))

**B3:** Sinh mã Huffman cho mỗi ký tự dựa vào cây Huffman. (void printCodes(struct MinHeapNode\* root, string str); void storeCodes(struct MinHeapNode\* root, string str))

**B4:** Duyệt file, thay toàn bộ ký tự bằng mã Huffman tương ứng.

**B5:** Lưu lại cây Huffman (bảng mã) dùng cho việc giải nén. Xuất file đã nén.

### Giải nén:

Giải nén:dùng 1 chuỗi để lưu kết quả giải nén, dùng 1 vòng lặp để duyệt dãy nhị phân và đọc từng bit, đặt 1 con trỏ Curr ở gốc cây Huffman, ứng với mỗi bit 0 t di chuyển con trỏ sang trái và bit 1 thì di chuyển sang phải và kiểm tra xem con trỏ Curr có đang trỏ đến nút lá không. Nếu chưa thì lặp lại các bước trên, nếu là nút lá thì đưa con trỏ Curr về lại gốc và xuất ra ký tự tìm được vào chuỗi lưu kết quả.

Chuyển đổi từ chuỗi ký tự sang tiếng việt:(Chưa có cách sửa được lỗi font chữ): Tìm kiếm trên chuỗi kết quả những chuỗi con đặc biệt(af,ax,…), những chuỗi này là những chuỗi vô nghĩa trong tiếng việt, đổi sang những ký tự chứa dấu thông qua class trung gian là “tranfer.h”(\*) với phương thức inverse giúp chuyển đổi đầu vào: chuỗi kết quả (string) thành đầu ra: chuỗi có dấu (string).

(\*)Class “tranfer.h”: chứa 2 thuộc tính là Sign Vietnamese và Unsigned Vietnamese chứa các chuỗi có dấu và không dấu tiếng việt và dựa vào 2 hàm có sẵn trong thư viện string<String.h> cơ bản của c++ là replace() và find() để tìm và thay thế những ký tự có dấu sang chuỗi không có dấu theo kiểu telex(Unikey).

## Ngôn ngữ cài đặt

C++

## Kết quả

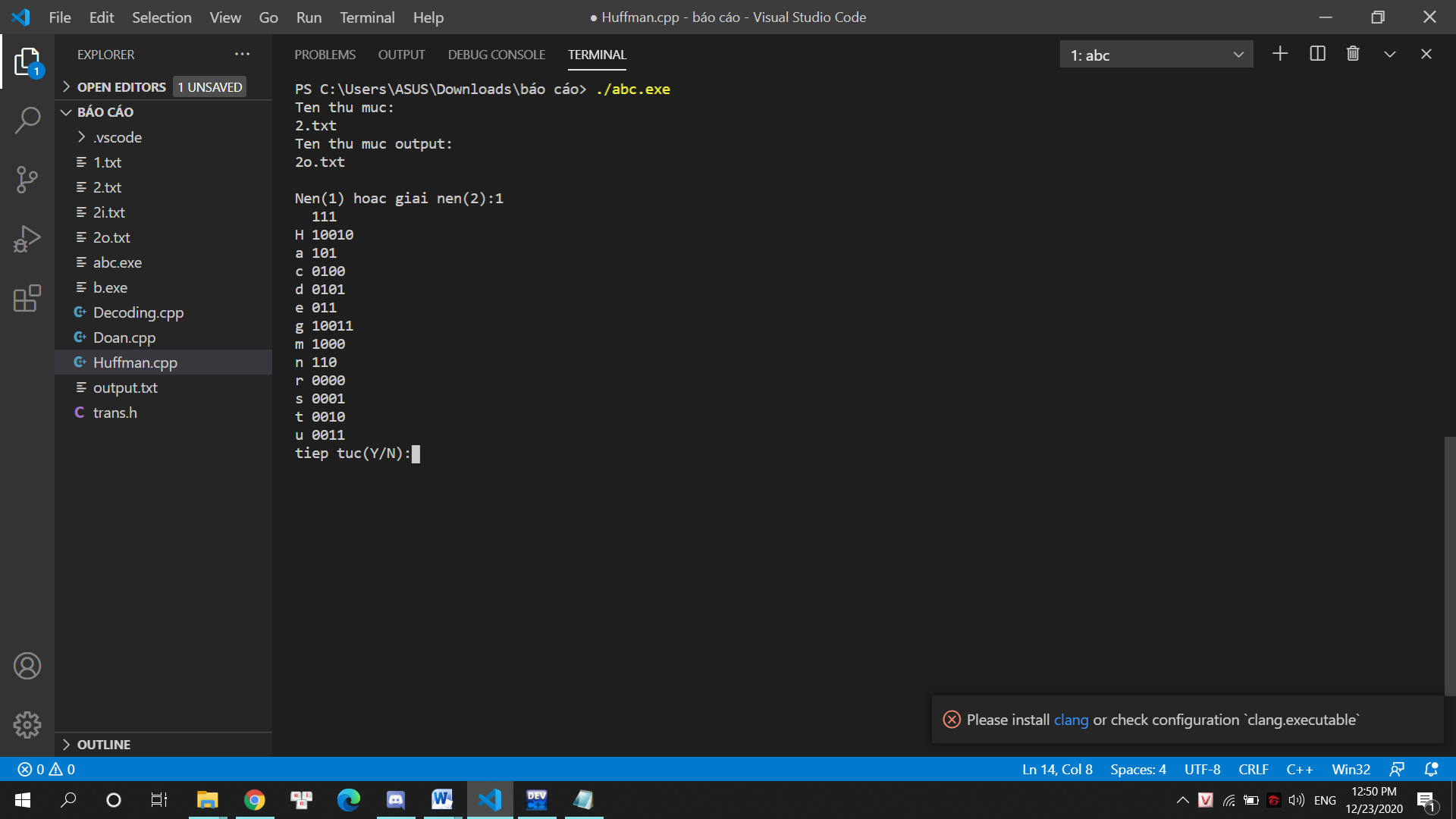
### Giao diện chính của chương trình



Hình : Giao diện chính của chương trình

### Kết quả thực thi của chương trình

#### Nén:



Hình : Kết quả của nén

#### Giải nén

#### 

Hình : Kết quả của giải nén

### Nhận xét đánh giá

Tuy chương trình đã phân biệt được tiếng việt và cho ra kết quả đúng với chuỗi ban đầu nhưng vẫn còn hạn chế với những văn bản quá dài việc lưu dữ liệu sẽ bị hạn chế và quả tải nên chương trình sẽ tự động ngắt và không thực hiện.

Giải nén: Kết quả đúng với chuỗi không dấu chưa được nén, tuy nhiên việc chuyển đổi sang tiếng việt có dấu vẫn còn trục trặc vì C++ hiện vẫn chưa cung cấp đầy đủ phương thức lưu trữ và chuyển đổi những ký tự có dấu. Nếu có thể phát triển thêm về mặt lưu trữ thì có thể cho ra kết quả tốt hơn. (Hoặc có thể sử dụng những tiện ích bên ngoài C++ để hỗ trợ cho việc chuyển đổi tiếng việt).

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Nhìn chung chương đã thực hiện được đầy đủ được các yêu cầu của đề tài tuy nhiên vẫn tồn tại một số lỗi và vẫn chưa thực hiện được ngoài các yêu cầu đề tài như vẫn chưa giải nén được về tiếng việt một cách hoàn thiện.

## Hướng phát triển

Trong tương lai sẽ tiếp tục phát triển chương trình có thể nén và giải nén một văn bản dài hơn, bổ sung việc giải nén tiếng việt.

Tiếp tục sửa lỗi và hoàn thiện chương trình với thời gian ngắn hơn và hiệu quả hơn.

TÀI LIỆU THAM KHẢO

[1] <http://vnoi.info/library/56/4958/>

[2] <http://dulieu.tailieuhoctap.vn/books/cong-nghe-thong-tin/co-so-du-lieu/file_goc_768105.pdf>

[3] , <https://vietjack.com/cplusplus/struct_trong_cplusplus.jsp>

[4], <https://daynhauhoc.com/t/file-va-cac-thao-tac-co-ban-voi-file-trong-c/35276>

[5], <https://vi.wikipedia.org/wiki/M%C3%A3_h%C3%B3a_Huffman>

[6], http://cunglaptrinh.blogspot.com/2015/03/tong-quan-ve-kieu-string-trong-cpp.html

PHỤ LỤC

#include <bits/stdc++.h>

#include"trans.h"

#define MAX\_TREE\_HT 256

using namespace std;

map<char, string> codes;

map<char, int> freq;

struct Tree

{

char data;

int freq;

Tree \*left, \*right;

Tree()

{

left = right = nullptr;

}

**//Hàm gán giá trị cho con trỏ ký tự và tần số có trong file.**

**Tham số đầu vào: ký tự , tần số**

Tree(char data, int freq)

{

left = right = NULL;

this->data = data;

this->freq = freq;

}

};

**// Hàm trả về cây nhị phân với độ ưu tiên từ trái sang phải)**

struct compare

{

bool operator()(Tree\* l, Tree\* r)

{

return (l->freq > r->freq);

}

};

**//Hàm in ra chuỗi đã mã hóa**

**Tham số đầu vào: con trỏ hiện tại đang trỏ đến của cây nhị phân, mảng nhị phân sẽ in ra**

void printCodes(struct Tree\* root, string str)

{

if (!root)

return;

if (root->data != '$')

cout << root->data << ": " << str << "\n";

printCodes(root->left, str + "0");

printCodes(root->right, str + "1");

}

**//Hàm lưu trữ giá trị của cây nhị phân.**

**Tham số đầu vào: con trỏ hiện tại đang trỏ đến của cây nhị phân, mảng nhị phân sẽ in ra**

void storeCodes(struct Tree\* root, string str)

{

if (root==NULL)

return;

if (root->data != '$')

codes[root->data]=str;

storeCodes(root->left, str + "0");

storeCodes(root->right, str + "1");

}

**// Tao cây theo minHeap hay cây huffman với độ ưu tiên là bên trái(0) trước bên phải(1) sau**

priority\_queue<Tree\*, vector<Tree\*>, compare> minHeap;

**// Hàm xây dựng cây nhị phân và lưu trữ minHeap: tham số đầu vao: kích thước của file**

void HuffmanCodes(int size)

{

struct Tree \*left, \*right, \*top; //Tao ngan xep

for (map<char, int>::iterator v=freq.begin(); v!=freq.end(); v++) //Khai bao bien lap interator v de lay tan so ky tu x

minHeap.push(new Tree(v->first, v->second)); // Tao kich thuoc cho size minHeap theo so ky tu co trong file

while (minHeap.size() != 1)

{

left = minHeap.top(); //phan tu co do uu tien lon nhat voi do uu tien tu trai sang phai va tu duoi len

minHeap.pop(); //Xoa ky tu cua minHeap cho toi khi = 1 hay noi cach khac la xoa cay da duyet

right = minHeap.top();

minHeap.pop();

top = new Tree('$', left->freq + right->freq);

top->left = left;

top->right = right;

minHeap.push(top); //day gia tri top len 1

}

storeCodes(minHeap.top(), "");

}

**//Hàm đếm tần số xuất hiện các ký tự trong file:**

**Tham số đầu vào: chuỗi ký tự trong file, số nguyên n**

void calcFreq(string str, int n)

{

for (int i=0; i<str.size(); i++)

freq[str[i]]++;

}

Tree Htree;

Hàm thêm lá vào cây HuffMan ngược:

Tham số vào: con trỏ cây Huffman, ký tự cần thêm vào, mã của ký tự của lá.

void buildHuffman(struct Tree\* root,char bit, string bitcode)

{

Tree\* curr=root;

for(int i=0;i<(bitcode.size()-1);i++)

{

if (bitcode[i]=='0')

{

if ((curr->left)==nullptr)

{

curr->left=new struct Tree;

}

curr=curr->left;

}

else

{

if (curr->right==NULL) curr->right=new struct Tree;

curr=curr->right;

}

}

curr->data=bit;

}

//Hàm giải nén Huffman

Tham số vào: cây Huffman, chuỗi nhị phân; Giá trị trả về: chuỗi lưu dữ liệu đã được giải nén.

string decoding(struct Tree\* root, string s)

{

string ans = "";

struct Tree\* curr = root;

for (int i=0;i<s.size();i++)

{

if (s[i] == '0')

curr = curr->left;

else

curr = curr->right;

if (curr->left==NULL and curr->right==NULL)

{

ans += curr->data;

curr = root;

}

}

return ans+'\0';

}

int main()

{

int compress;

char filename[20],oname[20];

FILE \*input;

fstream f;

int co=0;

do

{

do

{

cout<<"Ten thu muc: "<<endl;

scanf("%s",filename);

input = fopen(filename, "r");

if(input==NULL) cout << "khong ton tai file yeu cau" << endl;

}while (input==NULL);

cout<<"Ten thu muc output: "<<endl;

scanf("%s",oname);

f.open(oname, ios::out);

printf("\nNen(1) hoac giai nen(2):");

scanf("%d",&compress);

string encodedString, decodedString;

if (compress==1)

{

char\* tmp;

fgets(tmp,1000000000, input);

string str = string(tmp);

trans a(str);

str=a.transfer(str);

calcFreq(str, str.length());

HuffmanCodes(str.length());

for (auto v=codes.begin(); v!=codes.end(); v++)

{

cout << v->first <<' ' << v->second << endl;

f << v->first <<' ' << v->second << endl;

}

for (auto i: str)

encodedString+=codes[i];

f <<encodedString;

f.close();

}

cout<<"tiep tuc(0/1):";cin>>co;

if (compress==2)

{

char bit[256];

string bitcode[256];

char c[10000];

int i=0;

do

{

fgets(c,100000000,input);

if((((c[0]=='0')||(c[0]=='1')))and(c[1]!=' ')) break;

bit[i]=c[0];

for(int j = 2;j<strlen(c);j++) bitcode[i]=bitcode[i]+c[j];

i++;

}while(1);

for(int j = 0;j<strlen(c);j++) encodedString=encodedString+c[j];

for(int j = 0;j<i;j++)

{

buildHuffman(&Htree,bit[j],bitcode[j]);

}

decodedString = decoding(&Htree, encodedString);

cout << "\nDu lieu da duoc giai nen:\n" << decodedString << endl;

trans a(decodedString);

decodedString=a.inverse(decodedString);

f <<decodedString;

f.close();

//cout<<"tiep tuc(Y/N):";cin>>co;

}

cout<<"tiep tuc(0/1):";cin>>co;

}while(co==0);

return 0;

}