TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THỒNG TIN

---------------o0o---------------

Ảnh có chứa biểu tượng, Nhãn hiệu, Phông chữ, vòng tròn

Mô tả được tạo tự động

**Bài tập lớn môn học**

**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

Giảng viên hướng dẫn: TS. Hoàng Văn Thông

Sinh viên thực hiện: Hoàng Mạnh Khiêm

Lớp: CNTT VA 1 – K63

Hà Nội, tháng 11 năm 2023

# **MỤC LỤC**

MỤC LỤC 2

I.BÀI 1 – BÀI SỐ 29 3

**1. Đề Bài** 3

2. Phân tích bài toán 3

a. Yêu cầu 3

b. Cài đặt các lớp và hàm main bằng C++ 3

b.1 HuffmanTree.h 3

b.2 main.cpp 5

c. Phân tích thời gian chạy có trong phương thức của các lớp 6

c.1 HuffmanTree.h 6

c.1.1 Phương thức Node \*newNode(char c, int freq, Node \*left, Node \*right) 6

c.1.2 Phương thức void encode(Node \*root, string str, map<char, string> &huffmanCode) 6

c.1.3 Phương thức void decode(Node \*root, int &index, string str) 6

c.1.4 Phương thức void buildHuffmanTree\_encode(string text) 6

# **I.BÀI 1 – BÀI SỐ 29**

## **1. Đề Bài**

Tìm hiểu về mã hóa Huffman, hàng đợi ưu tiên và thuật toán xây dựng cây Huffman. Xây dựng lớp

biểu diễn cây Huffman có các chức năng sau:

a. Cài đặt thuật toán xây dựng cây Huffman sử dụng hàng đợi ưu tiên.

b. Duyệt cây và gán các từ mã cho các ký tự.

c. Xây dựng ứng dụng nén chuỗi ký tự bằng thuật toán nén tĩnh và giải nén chuỗi ký tự đã

được nén.

## 2. Phân tích bài toán

### a. Yêu cầu

- Tìm hiểu mã hóa Huffman

- Tìm hiểu hàng đợi ưu tiên

- Thuật toán xây dựng cây Huffman

* Xây dựng cây Huffman sử dụng lớp Huffman được xây dựng và lưu trữ dưới dạng một file Header C++.

+ Các thuộc tính của cây Huffman bao gồm:

Node: Cấu trúc node cho cây Huffman

Char c: kí tự mà tại một node của cây đang chứa

Int freq: Tần số của kí tự xuất hiện trong chuỗi cần giải mã, mã hóa

Node \*left, Node \*right: Node trái, Node phải của cây

+ Các phương thức của cây Huffman bao gồm:

Node \*newNode(char c, int freq, Node \*left, Node \*right)

void encode(Node \*root, string str, map<char, string> &huffmanCode)

void decode(Node \*root, int &index, string str)

void buildHuffmanTree\_encode(string text)

void buildHuffmanTree\_decode(map<char, int> freq, string str)

* Chức năng của các phương thức có trong cây Huffman

+ Node \*newNode(char c, int freq, Node \*left, Node \*right): Tạo một node mới cho cây.

+ void encode(Node \*root, string str, map<char, string> &huffmanCode): Hàm đệ quy để gán các các kí tự vào một node của cây, gán thêm giá trị “0” hoặc “1” cho node trái và node phải.

+ void decode(Node \*root, int &index, string str): Duyệt đoạn code đã được mã hóa, đồng thời kiểm tra khi đến lá thì in ra được kí tự nằm ở nút lá cần được xác định.

+ void buildHuffmanTree\_encode(string text): Xây dựng cây Huffman mã hóa code

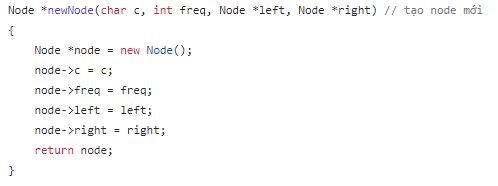
+ void buildHuffmanTree\_decode(map<char, int> freq, string str): Xây dựng cây Huffman giải mã code dựa trên tần suất xuất hiện và chuỗi đã mã hóa

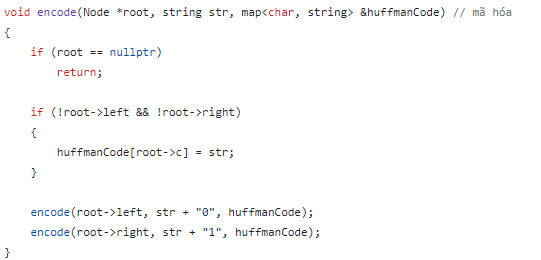
### b. Cài đặt các lớp và hàm main bằng C++

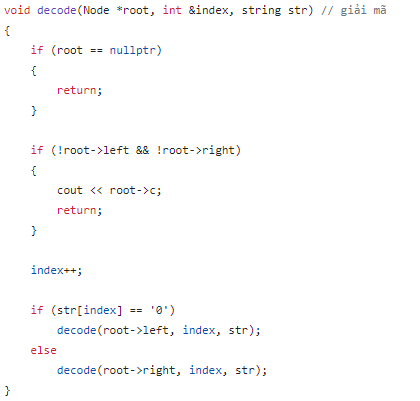
#### b.1 HuffmanTree.h

- Cây Huffman được xây dựng riêng tại một file Header C++ có tên là: **HuffmanTree.h**

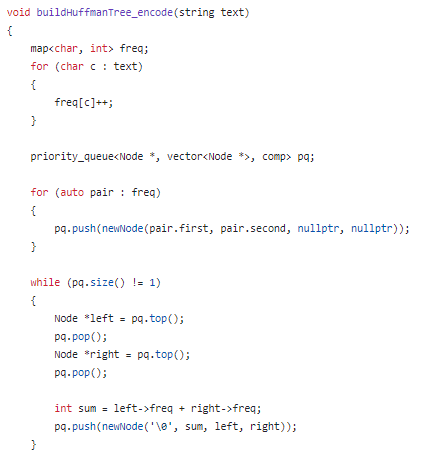
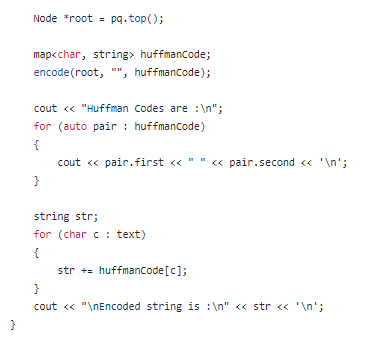
- Cài đặt phương thức:

 + Node \*newNode(char c, int freq, Node \*left, Node \*right)

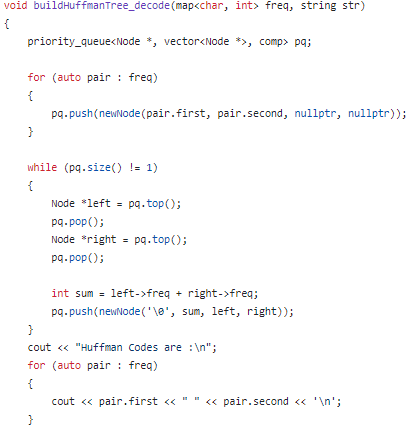
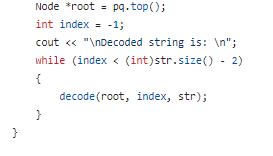
 + void encode(Node \*root, string str, map<char, string> &huffmanCode)

 + void decode(Node \*root, int &index, string str)

+ void buildHuffmanTree\_encode(string text)



+ void buildHuffmanTree\_decode(map<char, int> freq, string str)

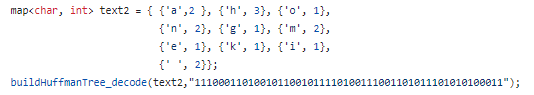
 

#### b.2 main.cpp

- Hàm main được xây dựng riêng trong một file tên main.cpp

- Sử dụng #include “HuffmanTree.h” để truy cập tới thư viện chứa cây Huffman

- Sử dụng  để chạy phương thức mã hóa của cây Huffman

- Sử dụng  để giải mã Huffman

### c. Phân tích thời gian chạy có trong phương thức của các lớp

#### c.1 HuffmanTree.h

##### c.1.1 Phương thức Node \*newNode(char c, int freq, Node \*left, Node \*right)

- Có 4 phép gán giá trị cho node

- Có 1 phép cấp phát bộ nhớ để tạo một node mới

- Có 1 phép trả về node đã được tạo

=> Thời gian chạy: O(6)

##### c.1.2 Phương thức void encode(Node \*root, string str, map<char, string> &huffmanCode)

- Có 2 lệnh if tương đương với có 2 phép toán

- Mỗi hàm gọi đệ quy cho node con trái và node con phải sẽ mất thời gian O(2)

- Tổng số hàm gọi đệ quy trong phương thức này phụ thuộc vào số lượng các giá trị của node lá nên sẽ mất tổng thời gian là O(n)

=> Thời gian chạy O(n)

##### c.1.3 Phương thức void decode(Node \*root, int &index, string str)

- Có 2 lệnh if tương đương với có 2 phép toán

- Có một lệnh tăng giá trị index lên 1 đơn vị là 1 phép toán

- Với mỗi hàm gọi đệ quy lại chính nó trong phương thức này phụ thuộc vào giá trị đầu vào của đoạn code cần giải mã nên phương thức sẽ mất tổng thời gian là O(n)

=> Thời gian chạy O(n)

##### c.1.4 Phương thức void buildHuffmanTree\_encode(string text)

- Một lệnh khai báo map và gán tên biến cho cấu trúc dữ liệu map : 1 phép toán

- Duyệt chuỗi để tính tần suất xuất hiện của các kí tự trong chuỗi cần mã hóa : 2\*n phép toán

- Khai báo priority\_queue : 1 phép toán

- Duyệt các tần suất khi đã lưu vào map và được đẩy vào priority\_queue dưới dạng một node mới : 2\*n+6 phép toán

- Vòng lặp cài đặt các node trái và phải từ các giá trị đã được đẩy vào priority\_queue : 12 phép toán.

- Khai báo một node cuối cùng còn lại trong priority\_queue là node gốc : 1 phép toán.

- Khai báo map và gán tên biến cho map : 1 phép toán.

- Dùng phương thức encode để giải mã: n phép toán với n là độ dài của chuỗi cần mã hóa

- Vòng lặp và dòng lệnh được in ra các quy ước của mã huffman : n+1 : phép toán

- Khai báo một chuỗi kí tự và duyệt vòng lặp in ra chuỗi đã mã hóa: n+2 phép toán

=> Thời gian chạy : O(2\*n)