# Bài tập lớn 2

Version: 1.00

## I- Giới thiệu

Trong bài tập lớn 2 này, sinh viên sẽ lập trình cho một bài toán cụ thể. Các kiến thức sau đây sẽ được sử dụng trong bài tập lớn:

- Tree, binary tree
- Heap
- String
- Thao tác trên bit

Ở bài tập lớn này, sinh viên được yêu cầu hoàn thiện một chương trình nén và giải nén cho dữ liệu text, chi tiết tại mục II.

## II- Mô tả

### 1. Bài toán

Trong thực tế sử dụng máy tính, chúng ta thường xuyên làm việc với một lượng lớn dữ liệu, và thông thường, để lưu trữ và sử dụng lâu dài, chúng được lưu vào cấu trúc tập tin (file), thư mục. Với lượng lưu trữ lớn, và trong nhiều ngữ cảnh, chúng ta có nhu cầu nén dữ liệu: tiết kiệm không gian lưu trữ, truyền tải qua mạng nhanh hơn, gọn nhẹ dễ dàng sao chép,… và từ đó, nhiều chương trình nén dữ liệu ra đời: WinZip, WinRar, 7-Zip, tar, …

Trong bài tập lớn này, sinh viên sẽ hiện thực một phương pháp nén dữ liệu đơn giản dựa trên "**Huffman coding**", chi tiết về phương pháp sinh viên tham khảo tại đây:

http://en.wikipedia.org/wiki/Huffman_coding

Như chúng ta đã biết, để biểu diễn các ký tự thông thường (a, b,…,A,…Z,0..9, ~!@#$%...) chúng ta sử dụng bảng mã ASCII, và một byte (8 bits) được sử dụng để biểu diễn cho một ký tự, ví dụ: 'A' được biểu diễn tương ứng với 65, '0' tương ứng với giá trị 48. Tuy nhiên, việc biểu diễn như vậy đôi lúc gây ra sự lãng phí, bởi vì trong thực tế các ký tự khác nhau có tần số sử dụng không giống nhau, ví dụ các ký tự 'a', 'e' thường được sử dụng nhiều hơn các ký tự khác trong văn bản. Vì vậy, ý tưởng cơ bản của "Huffman coding" là sử dụng chuỗi bit có độ dài khác nhau để biểu diễn cho các ký tự dựa vào tần số xuất hiện của các ký tự đó, ký tự có tần số xuất hiện càng cao thì sử dụng càng ít bit để biểu diễn cho ký tự đó. Chi tiết hơn về phương pháp sinh viên tham khảo theo đường link ở trên.

Để đơn giản, sinh viên được yêu cầu chỉ hiện thực chương trình nén, giải nén cho một file dạng text.

### 2. Giải thuật

a) Để thực hiện chương trình nén, giải nén theo phương pháp "Huffman coding" các bước sau sẽ được thực hiện:

Giải thuật nén:

```
1 Đọc file text, đếm số lượng xuất hiện của từng ký tự

2 Xây dựng Huffman tree ( theo giải thuật xây dựng cây Huffman)

3 Dựa vào cây vừa xây dựng, đọc lại file text, ứng với từng ký tự ghi ra chuỗi bit tương ứng
```

Giải thuật xây dựng cây Huffman (tham khảo chi tiết tại link):

```
1 Create a leaf node for each symbol and add it to the priority queue.

2 While there is more than one node in the queue:

        1 Remove the two nodes of highest priority (lowest probability) from the queue

        2 Create a new internal node with these two nodes as children and with probability equal to the sum of the two nodes' probabilities.

        3 Add the new node to the queue.

3 The remaining node is the root node and the tree is complete.
```

b) Để thực hiện việc giải nén giải thuật sau sẽ được thực hiện:

```
1 Đọc header của file, kiểm tra định dạng

2 Từ header đọc bộ từ điển nén (mỗi ký tự sẽ được nén bằng bao nhiêu bit, chuỗi bit như thế nào)

3 Xây dựng lại Huffman tree dựa vào bộ từ điển nén

4 Đọc phần body của file lần lượt từng bit và dựa vào Huffman tree vừa xây dựng để giải nén lấy ra các ký tự ban đầu
```

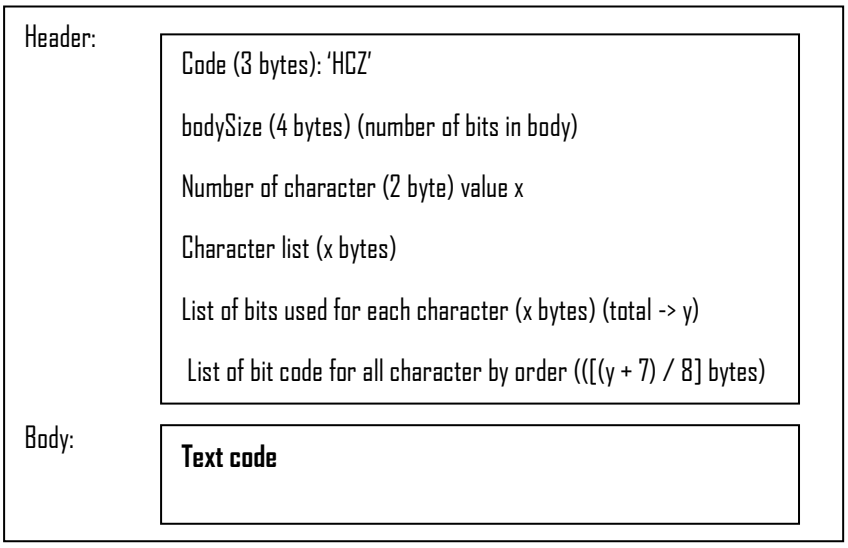
### 3. Hiện thực bài tập lớn

Sinh viên đọc kỹ đặc tả nội dung bài tập lớn, tham khảo đường link **Wikipedia** đính kèm theo.

Trong bài tập lớn này, sinh viên được cung cấp sẵn một số files mã nguồn bao gồm:

- HCZHeader.h, HCZHeader.cpp: định nghĩa class dùng để đọc, ghi, xử lý header cho file nén.
- Reader.h, Reader.cpp: định nghĩa class dùng để đọc thông tin từ file theo từng bit, byte, word, dword
- Writer.h, Writer.cpp: định nghĩa class dùng để ghi thông tin ra file theo từng bit, byte, word, dword
- Main.cpp: chứa hàm main sẽ được gọi lúc chương trình thực thi
- **HuffmanCode.h , HuffmanCode.cpp:** các file định nghĩa và hiện thực giải thuật nén, và giải nén dựa theo phương pháp HuffmanCode.
- Input.txt: chứa ví dụ mẫu được sử dụng để nén.
- Makefile: sử dụng để dịch chương trình, tạo ra file thực thi (sử dụng dưới cygwin hay Linux)

File nén mà chương trình sinh ra hay sử dụng có cấu trúc gồm header và liền sau đó là phần body theo mô tả ở dưới:




HCZHeader đã được hiện thực với đầy đủ các phương thức cần thiết để làm việc với Header này, sinh viên chỉ cần tìm hiểu và sử dụng hợp lý trong cả quá trình nén, và giải nén.

Trong đó:

bodySize: tổng kích thước (tính bằng bit) của phần text sau khi được nén.

Number of character: số lượng ký tự được sử dụng, giả sử giá trị này là *x*

Character list: danh sách các ký tự được sử dụng trong text ban đầu (*x* ký tự)

List of bits used for each character: danh sách số lượng bit biểu diễn cho từng ký tự (danh sách có *x* phần tử). Giả sử tổng tất cả các giá trị trong danh sách này là *y*

List of bit code for all character by order: chuỗi bit biểu diễn tương ứng theo thứ tự đó dùng trong mã hóa cho mỗi ký tự tương ứng với danh sách Character list

Ví dụ với đoạn text như sau:

abcdabcaba

Sau khi phân tích giả sử ta mã hóa các ký tự tương ứng với chuỗi bit như sau: *a-0, b-10, c-111, d-110,* độ dài tương ứng chuỗi bit là 1, 2, 3 và 3. Từ đó theo chuỗi text và chuỗi bit biểu diễn ta có các thông tin sau:

bodySize = 19, Number of character *x* = 4, Character list = {a, b, c, d}, List of bits used for each character = {1, 2, 3, 3}, List of bit code = "010111110" (biểu diễn trong máy tính dạng nhị phân).

Tương ứng với ví dụ này ta có chuỗi bit cho đoạn text được nén như sau:

0101111100101110100

**Chú ý**: trong trường hợp **bodySize** không chia hết cho 8 (không lấp hết byte cuối cùng) thì sinh viên chỉ sử dụng đến số bit trong **bodySize**, không sử dụng các bit còn lại. Đây là trường hợp có thể xảy ra khi giải nén file đã được nén. Ở ví dụ trên, chuỗi bit nén sẽ được bổ sung các bit bất kỳ (0 hoặc 1 vào cuối) cho tròn byte (3 bytes = 24 bit) trước khi ghi ra file, do đơn vị nhỏ nhất mà file quản lý là byte. Trong trường hợp này, **5 bit cuối cùng không có ý nghĩa**.

Sinh viên được yêu cầu chỉ chỉnh sửa hai file **HuffmanCode.h** và **HuffmanCode.cpp** để định nghĩa và hiện thực các nội dung của bài tập lớn. Sinh viên có thể thêm, bớt các hàm, thủ tục hay biến, … tuy nhiên sinh viên **không được phép sử dụng thêm thư viên nào**, chỉ sử dụng các thư viện đã được include sẵn. Hai phương thức được gọi từ chương trình chính là:

**zip (char* inputFile, char* outputFile)** thực hiện việc nén file input và tạo ra file nén

**upzip (char* inputFile, char* outputFile)** thực hiện việc giải nén một file đã được nén theo phương pháp trên.

Vì vậy sinh viên không được phép chỉnh sửa tên cũng như các tham số, **nếu không hiện thực được những hàm này thì giữ nguyên hiện trạng.**

### 4. Chấm bài tập lớn

Bài tập lớn sẽ được chấm dựa theo testcase, bài làm của sinh viên sẽ được biên dịch và thực thi trên số lượng lớn các testcase khác nhau. Một số lưu ý trong chấm bài tập lớn như sau:

- Với giải thuật trên, có thể có các kết quả nén khác nhau, tuy nhiên nếu được hiện thực đúng thì tỉ lệ nén sẽ không có nhiều thay đổi.
- Đối với testcase yêu cầu unzip, so khớp kết quả sinh ra của sinh viên với file trước khi bị nén, điểm tương ứng sẽ là 0/1
- Đối với testcase yêu cầu zip
    - File nén do sinh viên sinh ra sẽ được một **chương trình mẫu** giải nén ngược lại để so sánh với file trước khi được nén, nếu không thể phục hồi được file gốc thì bị điểm 0 cho testcase đó.
    - Sẽ chấm dựa vào mức độ tốt (tỉ lệ nén) và so sánh với tỉ lệ nén của solution, điểm của 1 testcase là 0 đến 1.1 (1.1 khi mà tỉ lệ nén cao hơn cả solution).
- Với cùng phương pháp như trên, ta có thể unzip một file nén bất kỳ đã được zip với cùng phương pháp.
- ***Điểm tổng hợp của bài tập lớn = tổng điểm * 10 / tổng số testcase***

## III- Nộp bài

Khi nộp bài, sinh viên sử dụng account đã được cấp phát để nộp bài qua hệ thống Sakai. Sinh viên chỉ nộp đúng hai file **HuffmanCode.h** và **HuffmanCode.cpp (tên file phải được viết đúng dạng chữ hoa, chữ thường)**. File được nộp phải là file chương trình gốc, SINH VIÊN KHÔNG ĐƯỢC NÉN FILE KHI NỘP BÀI. **Sinh viên phải kiểm tra chương trình của mình trên Cygwin trước khi nộp.**

Thời hạn chót để nộp bài là **14:00** ngày **thứ sáu 1/11/2013**. KHÔNG nhận bài được gửi qua mail hoặc bất kỳ hình thức nào khác. Bài nộp trễ sẽ KHÔNG được nhận.

## IV- Xử lý gian lận

Bài tập lớn phải được sinh viên **TỰ LÀM**. Sinh viên sẽ bị coi là gian lận nếu:

- Có sự giống nhau bất thường giữa mã nguồn của các bài nộp. Trong trường hợp này, **TẤT CẢ** các bài nộp đều bị coi là gian lận. Do vậy sinh viên phải bảo vệ mã nguồn bài tập lớn của mình.
- Sinh viên không hiểu mã nguồn do chính mình viết, trừ những phần mã được cung cấp sẵn trong chương trình khởi tạo. Sinh viên có thể tham khảo từ bất kỳ nguồn tài liệu nào, tuy nhiên phải đảm bảo rằng mình hiểu rõ ý nghĩa của tất cả những dòng lệnh mà mình viết. Trong trường hợp không hiểu rõ mã nguồn của nơi mình tham khảo, sinh viên được đặc biệt cảnh báo là **KHÔNG ĐƯỢC** sử dụng mã nguồn này; thay vào đó nên sử dụng những gì đã được học để viết chương trình.

Trong trường hợp bị kết luận là gian lận, sinh viên sẽ bị điểm 0 cho toàn bộ môn học (không chỉ bài tập lớn).

**KHÔNG CHẤP NHẬN BẤT KỲ GIẢI THÍCH NÀO VÀ KHÔNG CÓ BẤT KỲ NGOẠI LỆ NÀO!**