**HUFFMAN ALGORITHM**

1. **Huffman Coding**
2. Khái quát
3. Định nghĩa

* Mã Huffman là một loại mã tiền tố tối ưu cụ thể thường được sử dụng để nén dữ liệu không mất dữ liệu. Quá trình tìm kiếm hoặc sử dụng một đoạn mã như vậy được tiến hành bằng phương pháp mã hóa Huffman.
* Đầu ra từ thuật toán của Huffman có thể được xem như một bảng mã có độ dài thay đổi để mã hóa một ký hiệu nguồn. Thuật toán rút ra bảng này từ xác suất ước tính hoặc tần suất xuất hiện (trọng số) cho mỗi giá trị có thể có của ký hiệu nguồn. Các ký hiệu phổ biến hơn thường được biểu diễn bằng cách sử dụng ít bit hơn các ký hiệu ít phổ biến hơn.

1. Thuật toán Huffman Coding

Gồm 3 bước

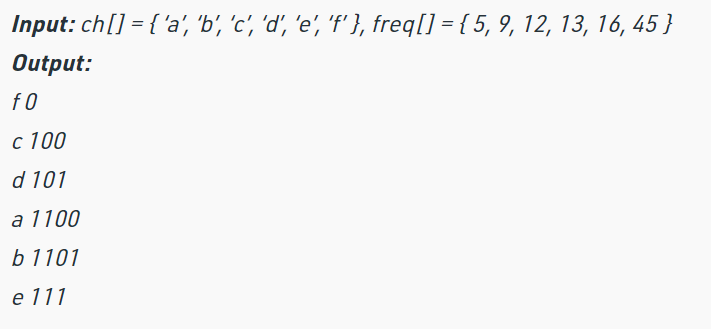
* Bước 1: Đếm tần suất xuất hiện của các phần tử trong chuỗi đầu vào.
* Bước 2: Xây dựng cây Huffman (cây nhị phân mã hóa).
* Bước 3: Từ cây Huffman, ta có được các giá trị mã hóa. Lúc này, ta có thể xây dựng chuỗi mã hóa từ các giá trị này.

1. Huffman Coding sử dụng **priority queue**

Thuật toán xây dựng đơn giản nhất sử dụng hàng đợi ưu tiên **(priority queue)** trong đó nút có xác suất thấp nhất được ưu tiên cao nhất

* Tạo một nút lá cho mỗi biểu tượng và thêm nó vào hàng đợi ưu tiên
* Trong khi có nhiều hơn một nút trong hàng đợi
* Xóa hai nút ưu tiên cao nhất (xác suất thấp nhất) khỏi hàng đợi
* Tạo một nút bên trong mới với hai nút này khi còn nhỏ và có xác suất bằng tổng của xác suất của hai nút
* Thêm nút mới vào hàng đợi
* Nút còn lại là nút gốc và cây hoàn thành

1. **Huffman Encoding**
2. Ví dụ



1. Tiếp cận

* **Push** tất cả các ký tự trong ch [] được ánh xạ tới tần số tương ứng freq [] trong hàng đợi ưu tiên.
* Để tạo cây Huffman, hãy **pop** hai nút từ hàng đợi ưu tiên.
* **Assign** hai nút **pop** từ hàng đợi ưu tiên là con trái và phải của nút mới.
* **push** nút mới được hình thành trong hàng đợi ưu tiên.
* Lặp lại tất cả các bước trên cho đến khi kích thước của hàng đợi ưu tiên trở thành 1.
* Đi qua cây Huffman (có gốc là nút duy nhất còn lại trong hàng đợi ưu tiên) để lưu trữ mã Huffman
* In tất cả các mã Huffman được lưu trữ cho mỗi ký tự trong ch [].

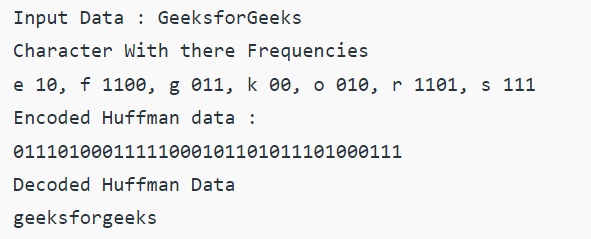
1. Cài đặt

<https://ideone.com/K3xWNk>

*Reference:*

<https://www.geeksforgeeks.org/huffman-coding-using-priority-queue/>

1. **Huffman Decoding**
2. Ví dụ

****

1. Tiếp cận

Để giải mã dữ liệu được mã hóa, chúng ta sử dụng cây Huffman. Chúng ta lặp lại thông qua dữ liệu được mã hóa nhị phân. Để tìm ký tự tương ứng với các bit hiện tại, ta sử dụng các bước đơn giản sau.

* B1: Bắt đầu từ gốc và làm theo cho đến khi tìm thấy một chiếc lá.
* B2: Nếu bit hiện tại là 0, di chuyển sang nút bên trái của cây.
* B3: Nếu bit là 1, chúng ta di chuyển đến nút bên phải của cây.
* B4: Trong quá trình truyền tải, nếu bắt gặp một nút lá thì in ký tự của nút lá cụ thể đó và sau đó tiếp tục lặp lại dữ liệu được mã hóa bắt đầu từ bước 1.

1. Cài đặt

<https://ideone.com/Mqnmsc>

*Reference*

<https://www.geeksforgeeks.org/huffman-decoding/>

1. **Adaptive Huffman Coding**
2. Khái quát

Adaptive Huffman Coding còn được gọi là Dynamic Huffman Coding, nó được thực hiện bằng thuật toán Vitter.

1. Adaptive Huffman Encoding
2. Tiếp cận

Mã hóa Adaptive Huffman cho một chuỗi chứa các kí tự trong bảng chữ cái Tiếng Anh:

Gọi m là tổng số chữ cái trong bảng chữ cái Tiếng Anh. Vậy m = 26.

Đối với thuật toán Vitter, hãy tìm một tham số e và r sao cho: m = 2e+ r và 0 <= r <= 2e

Do đó, với m = 26, ta nhận được e = 4 và r = 10.

Có hai loại mã là NYT Code và Fixed Code.

NYT Code = duyệt cây từ nút gốc đến nút NYT cụ thể đó.

Đối với Fixed code được tính theo hai trường hợp sau:

+) Nếu 0 <= k <= 2r, khí đó kí tự Sk được mã hóa dưới dạng biểu diễn nhị phân của (k-1) bằng (e+1) bit.

+) Ngược lại, kí tự Sk được mã hóa dưới dạng biểu diễn nhị phân của (k-r-1) bằng e bit.

Trong đó k là thứ tự của kí tự đang xét trong bảng chữ cái Tiếng Anh.

* Cập nhật:

Cập nhật cây theo thuật toán Vitter tuân theo đánh số ngầm định (fre):

* Các nút được đánh số theo thứ tự tang dần từ dưới lên trên (theo cấp độ) và từ trái sang phải.
* Các nút có cùng trọng lượng và cùng loại với nhau tạo thành một khối.
* Các khối có liên quan với nhau theo thứ tự fre tăng dần của chúng.
* Node bên trong được biểu diễn bằng hình Oval. Fre của các node trong bằng tổng fre của các nút con.
* Node ngoài được biểu diễn bằng hình vuông. Fre của các node ngoài, ban đầu là 1 và sau các lần lặp thì tăng lên 1.
* Các bước cập nhật cây:
* Khởi tạo cây bằng NYT Node.
* Nếu kí tự đang xét xuất hiện lần đầu tiên thì mở rộng NYT Node thành một NYT Node mới ở bên trái và một node ở bên phải hình vuông chứa kí tự đó và có fre là 1.
* Gán tổng fre các node con cho node cha.
* Nếu kí tự đang xét đã có trên cây thì cập nhật lại fre tại node chứa kí tự đó và các node đi qua trước khí duyệt cây tới node này.
* Giả mã Adaptive Huffman:

B1: Đọc chuỗi nhị phân đầu vào

B2: Nếu gặp node lá là NYT thì đọc các bit tiếp theo:

* Nếu giá trị thập phân của e bít tiếp theo <= r, chuyển đổi e + 1bit thành giá trị thập phân rồi cộng thêm 1 ta được vị trí của kí tự cần giải mã trong bảng chữ cái Tiếng Anh.
* Ngược lại nếu giá trị thập phân của e bit tiếp theo > r, chuyển đổi e bit đấy sang giá trị thập phân rồi cộng thêm r + 1, ta được vị trí của kí tự cần giải mã trong bảng chữ cái Tiếng Anh.

1. Cài đặt

<https://ideone.com/7Ydcyi>

1. Adaptive Huffman Decoding
2. Tiếp cận

B1: Đọc chuỗi nhị phân

B2: Nếu nút lá gặp phải là NYT

* Đọc các bit tiếp theo
* Nếu giá trị bit e <r, chuyển đổi (e+1) bit thành giá trị thập phân của chúng rồi cộng thêm 1
* Nếu giá trị bit e> r, chuyển đổi e bit sang giá trị thập phân của chúng rồi cộng thêm 1

1. Cài đặt

<https://ideone.com/pCSZOB>