Data compression

Created by anhtt-fit@mail.hut.edu.vn
Updated by huonglt-fit@mail.hut.edu.vn

Nén dữ liệu

- Dữ liệu trong bộ nhớ sử dụng một kích thước cố định để biểu diễn
- Thường cách này không đủ để truyền dữ liệu
- Để tăng tốc và sử dụng bộ nhớ hiệu quả, các ký hiệu cần dùng bộ nhớ ít nhất để biểu diễn.
- Các phương pháp nén dữ liệu
  - Mã hóa các ký hiệu có xác suất cao với ít bit hơn
    - Shannon-Fano, Huffman, UNIX compact
  - Mã hóa chuỗi ký hiệu bằng các vị trí liên tiếp trong từ điển
    - PKZIP, ARC, GIF, UNIX compress, V.42bis
  - Nén có mất dữ liệu
    - JPEG and MPEG

Mã với độ dài thay đổi

- Ví dụ 'A' xuất hiện 50 lần trong văn bản, nhưng 'B' chỉ xuất hiện 10 lần
- Mã ASCII dùng 8 bits cho 1 ký tự, tổng số bit 'A' và 'B' chiếm = 60 * 8 = 480
- Nếu 'A' dùng mã 4-bit, 'B' dùng mã 12-bit, tổng số bit = 50 * 4 + 10 * 12 = 320

Luật nén:

- Sử dụng ít bit nhất
- Không có mã nào là phần đầu của mã khác
- Cho phép giải mã từ trái sang phải mà không có nhập nhằng

Mã với độ dài thay đổi

- Không có mã nào là phần đầu của mã khác:
  - Ví dụ, ta không thể dùng mã 10 cho A và 100 cho B vì 10 là phần đầu của 100
- Cho phép giải mã từ trái sang phải mà không có nhập nhằng
  - Nếu thấy 10, biết đó là A. 10 không là phần bắt đầu của mã khác

Huffman code

- Mã hóa sử dụng cây mã, bắt đầu từ lá
- Mã nén được xây dựng sử dụng phương pháp xây dựng mã Huffman nhị phân

Huffman code Algorithm

① Tạo nút lá cho mỗi ký hiệu mã
   - Thêm xác suất cho mỗi nút lá

② Lấy 2 lá có xác suất nhỏ nhất và nối chúng vào 1 nút mới
   - Thêm 1 hoặc 0 cho 2 nhánh đó
   - Xác suất của nút mới = tổng xác suất 2 nút con

③ Nếu chỉ còn lại 1 nút, việc xây dựng mã hoàn tất. Nếu không, quay lại (2)

Demo

- 65demo-huffman.ppt

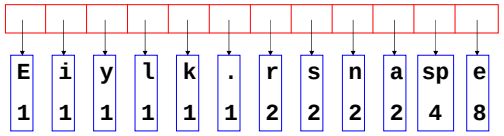
Nén văn bản

- Ví dụ:
  Eerie eyes seen near lake.
- Đến tổng số lần xuất hiện của các ký tự trong đoạn

Char	Freq.	Char	Freq.	Char	Freq.
E	1	y	1	k	1
e	8	s	2	.	1
r	2	n	2		
i	1	a	2		
space	4	l	1		

Dựng cây

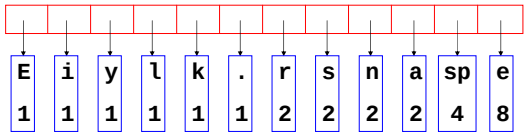
- Hàng đợi sau khi chèn các nút



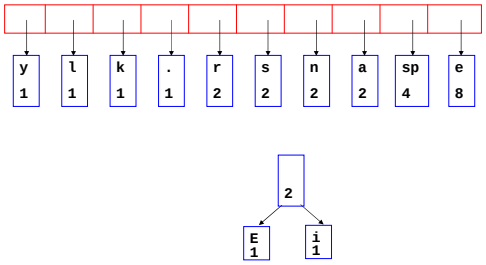
Dựng cây

- Khi hàng đợi có >= 2 nút
  - Tạo nút mới
  - Lấy nút ra khỏi hàng đợi và gắn nó thành cây con trái
  - Lấy nút tiếp theo ra khỏi hàng đợi và gắn nó thành cây con phải
  - Xác suất của nút mới = tổng xác suất của con trái và con phải
  - Đưa nút mới vào hàng đợi

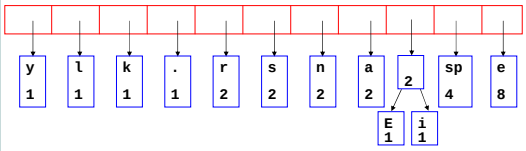
Dựng cây



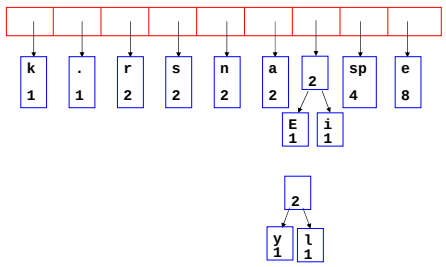
Dựng cây



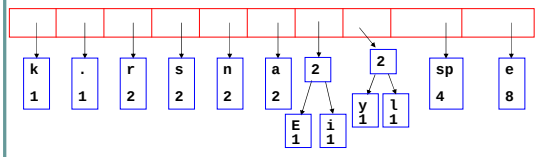
Dựng cây



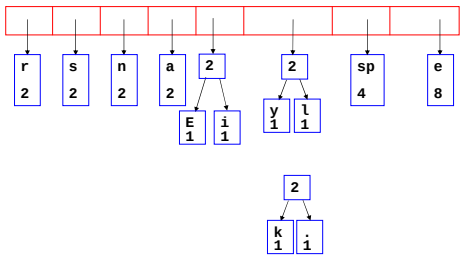
Dựng cây



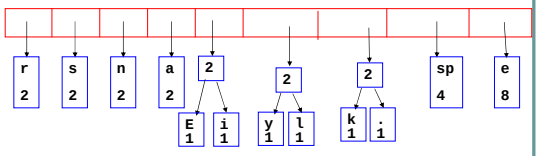
Dựng cây



Dựng cây

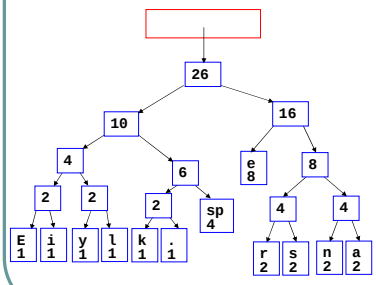


Dựng cây



- Vẫn còn tiếp

Cuối cùng



Sau khi đẩy nút này ra, chỉ còn lại 1 nút trong hàng đợi

Cài đặt

- Dùng JRB để biểu diễn cây
  - Mỗi nút mới được tạo là 1 JRB
  - Các cạnh đi từ nút cha đến nút con
  - 2 cạnh được tạo và gán nhãn 0 hoặc 1 khi nút cha được tạo
- Dùng Dllist hoặc JRB để biểu diễn hàng đợi. 1 nút trong hàng đợi có
  - key= tần suất của nút trong cây.
  - Value = con trỏ trỏ đến nút trong cây

Quiz 1

- Sử dụng graph API định nghĩa trong bài trước để viết hàm dựng cây Huffman từ 1 xâu như sau

```
typedef struct {
   Graph graph;
   JRB root;
} HuffmanTree;
HuffmanTree makeHuffman (char * buffer, int size);
```

Bảng mã Huffman

- Để nén xâu dữ liệu, cần dựng bảng mã từ cây Huffman. Cấu trúc dữ liệu sau dùng để biểu diễn bảng mã

```
typedef struct {
   int size;
   char bits[2];
} Coding;
Coding huffmanTable[256];
```

- huffmanTable['A'] có mã 'A'.
- Nếu size = 0, ký tự 'A' không xuất hiện trong xâu.
- Bits chứa mã huffman (chuỗi bit) của chuỗi đã cho

Quiz 2

- Viết hàm tạo bảng mã Huffman từ cây Huffman
  - void createHuffmanTable(HuffmanTree htree, Coding* htable);
- Viết hàm nén 1 bộ đệm xâu thành chuỗi Huffman
  - void compress(char * buffer, int size, char* huffman, int* nbit);
- bộ đệm có *size* ký tự. Sau khi nén, bộ đệm huffman buffer gồm nbit bits ở đầu ra.
- Để viết hàm này, cần tạo ra 1 hàm để thêm 1 ký tự mới vào bộ đệm huffman như sau
  - void addHuffmanChar(char * ch, Coding* htable, char* huffman, int* nbit);