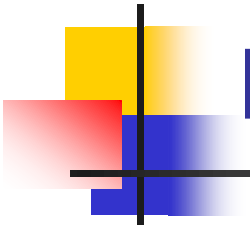
IT4853
Tìm kiếm và trình diễn thông tin

Bài 11. Nén chỉ mục ngược

IIR.C5. Index Compression

TS. Nguyễn Bá Ngọc, *Bộ môn Hệ thống thông tin, Viện CNTT & TT*
ngocnb@soict.hust.edu.vn

Hà Nội, 2016



Nội dung chính

- Các quy luật phân bố từ vựng
- Nén từ điển
- Nén danh sách mã văn bản

Quy luật Heap

$$M = kT^b,$$

- Trong đó M là kích thước bộ từ vựng; T là số từ trong bộ dữ liệu; k, b là các hằng số.
- Quan hệ tuyến tính trong mặt phẳng log-log:

$$\log(M) = \log(k) + b \log(T)$$

Quy luật Heap: Xác định các hằng số

Có thể dự đoán kích thước bộ từ vựng trước khi hoàn thành quá trình xây dựng chỉ mục ngược.

Least square error line trên các tập giá trị X, Y:

$$b_1 = \frac{\mathrm{cov}(X,Y)}{\mathrm{var}(X)^2} \qquad b_1 = \frac{\sum (X_i - \overline{X}) \cdot (Y_i - \overline{Y})}{\sum (X_i - \overline{X})^2}$$

$$b_0 = \overline{Y} - b_1 \overline{X}$$

$y = b_0 + b_1 x$

$b = b_1$

$\log(k) = b_0$

Quy luật Zipf

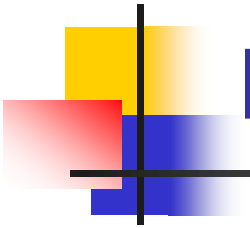
$$cf_i = K/i\ ,$$

- Trong đó K là hằng số; cf_i là tần suất bộ dữ liệu (*là số lần từ thứ i xuất hiện trong bộ dữ liệu*); i là chỉ số trong danh sách từ sắp xếp theo thứ tự giảm dần cf.

Quy luật Zipf (2)

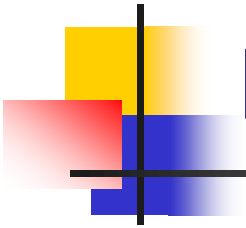
- $cf_2 = cf_1/2$; $cf_3 = cf_1/3$; v.v.
- Mối liên hệ tuyến tính giữa $\log(cf_i)$ và $\log(i)$:
  - $\log(cf_i) = \log(K) - \log(i)$

Có rất ít từ được sử dụng phổ biến nhưng có rất nhiều từ hiếm.



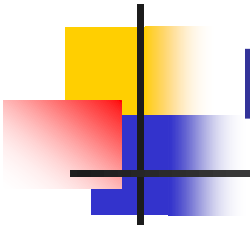
Nội dung chính

- Các quy luật phân bố từ vựng
- Nén từ điển
- Nén danh sách mã văn bản



Nén bảo toàn vs. không bảo toàn

- Nén bảo toàn:
  - Dữ liệu được bảo toàn sau khi giải nén;
  - Phổ biến nhất trong tìm kiếm.
- Nén không bảo toàn:
  - Loại bỏ một phần dữ liệu, tỉ lệ nén thường cao hơn phương pháp bảo toàn;
  - Có thể coi các phép lọc trong quá trình tách từ (chuẩn hóa cách viết, loại từ dừng, v.v.) là những phương pháp nén không bảo toàn.

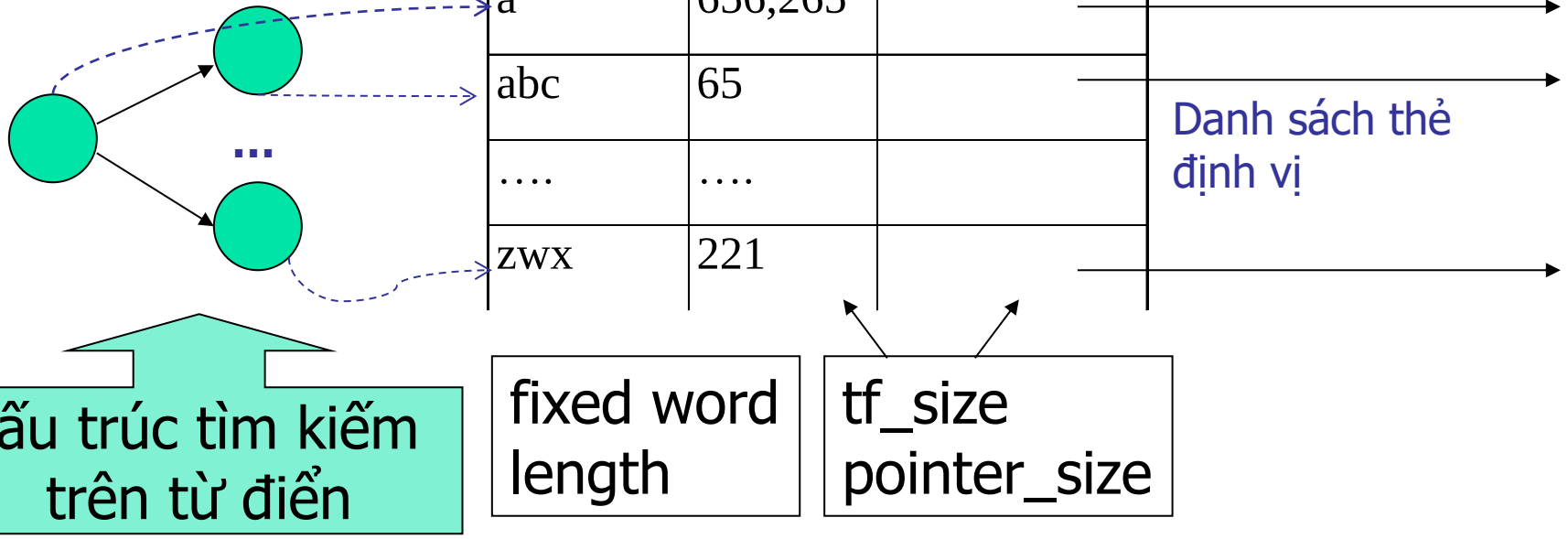


Lý do nén từ điển

- Thực hiện truy vấn luôn bắt đầu với tìm kiếm từ trong từ điển:
  - Cần sử dụng cấu trúc dữ liệu trong bộ nhớ để tìm kiếm nhanh;
- Áp dụng phương pháp nén giúp:
  - Lưu từ điển kích thước lớn trong bộ nhớ;
  - Giảm thời gian tải dữ liệu từ ổ đĩa.

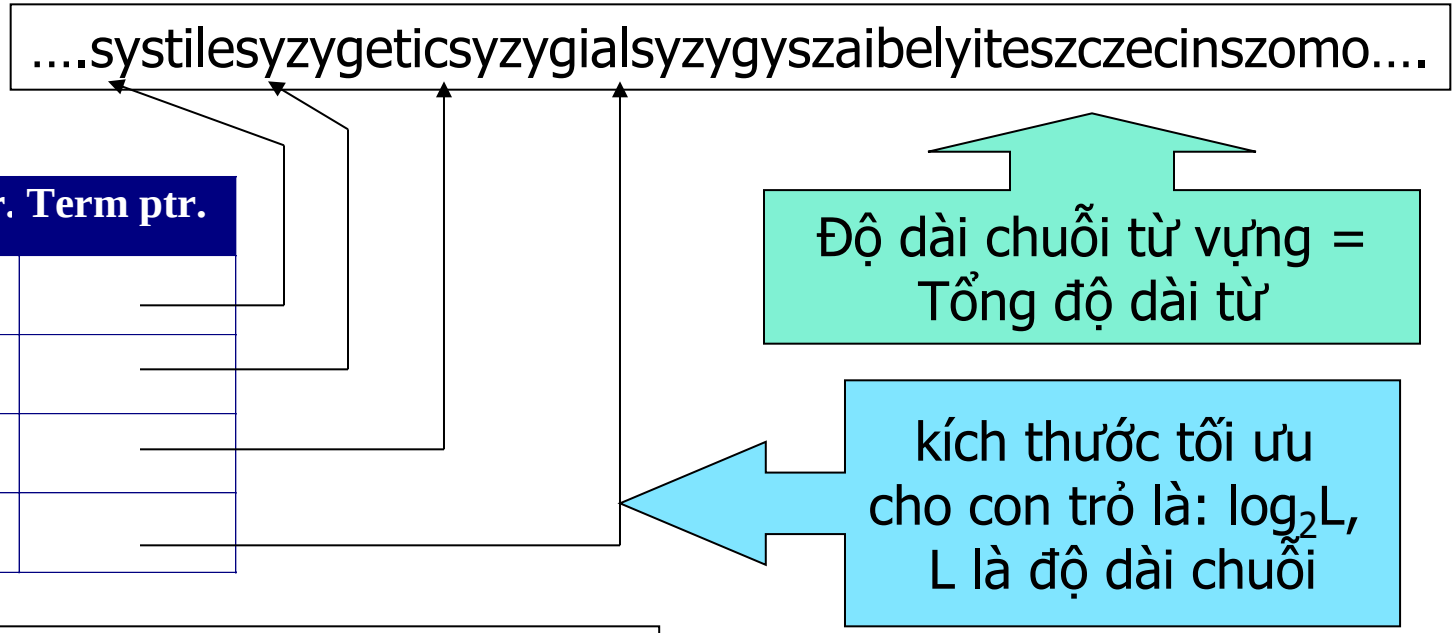
Mảng phần tử kích thước tĩnh

- Mảng phần tử kích thước tĩnh



Chuỗi ký tự dài

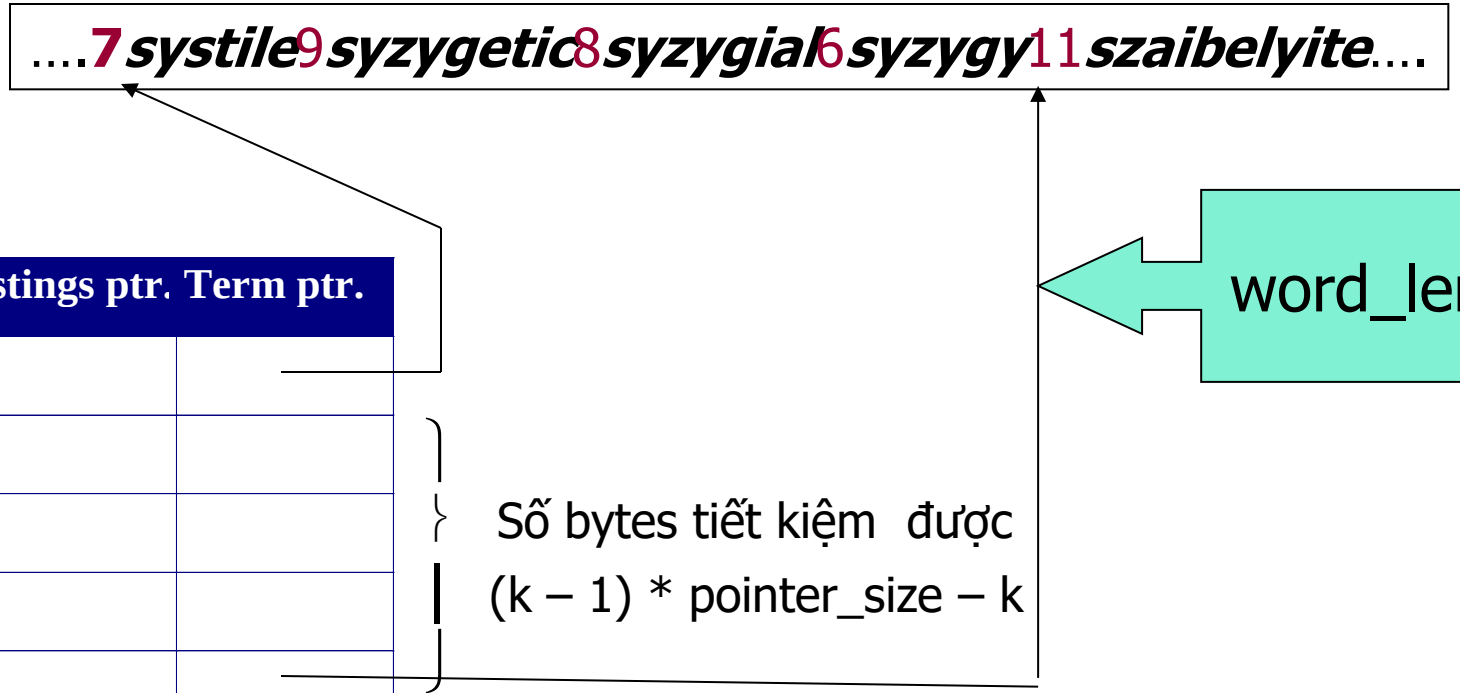
- Lưu bộ từ vựng như một chuỗi ký tự dài :
  - Con trỏ tới từ tiếp theo là dấu hiệu kết thúc từ hiện tại

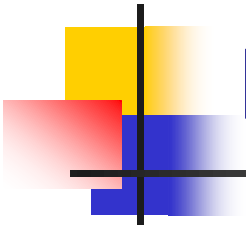


tf_size	pointer_size

Phân đoạn chuỗi ký tự dài

- Lưu con trỏ tới từ đầu tiên trong khối *k* từ.
  - Như ví dụ: *k*=4.
- Bổ xung 1 byte để lưu độ dài từ





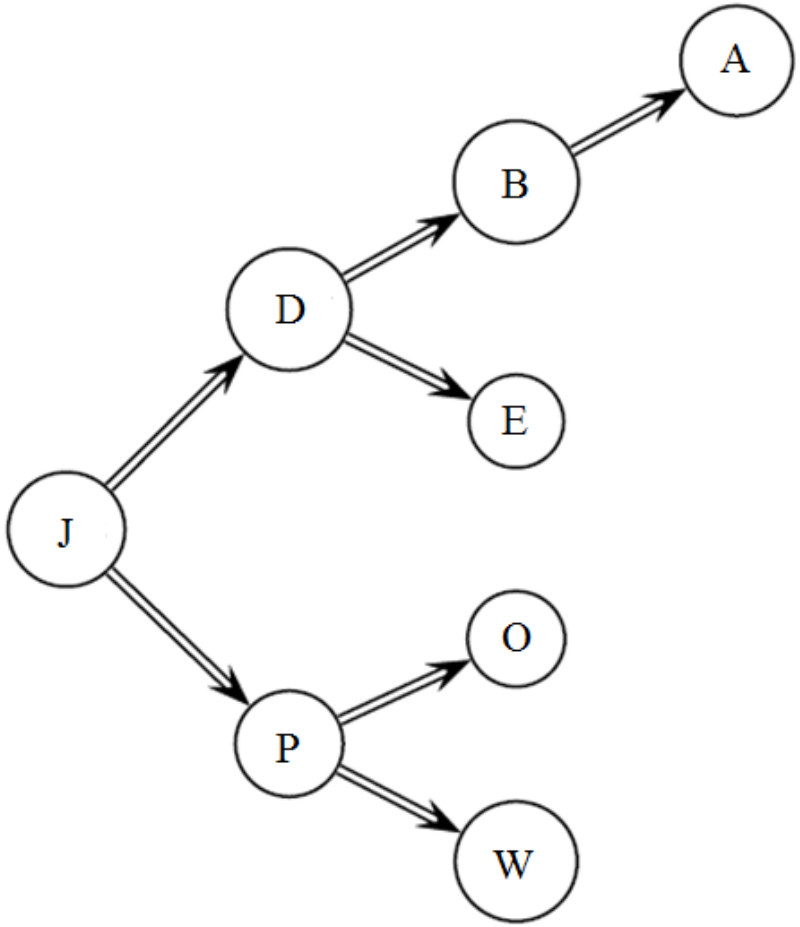
Phân đoạn

- Ví dụ với kích thước khối $k = 5$
- Khi chúng ta sử dụng 3 bytes/con trỏ, nếu không phân đoạn sẽ cần 5 x 3 = 15 bytes,
- Nếu sử dụng phân đoạn sẽ cần 3 + 5 = 8 bytes.
  - Tiết kiệm 7 bytes cho mỗi khối.

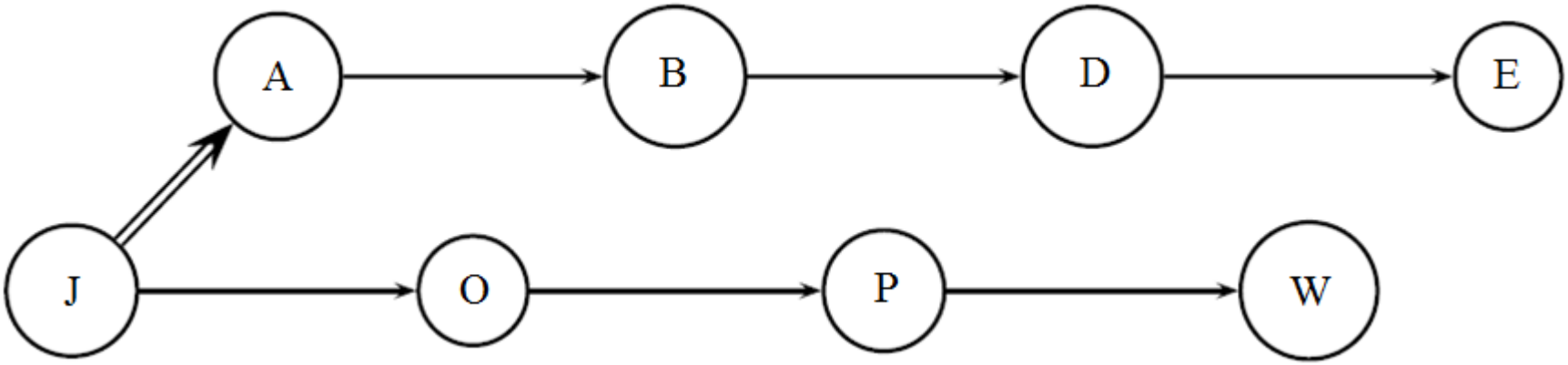
Thao tác này giảm kích thước từ điển,
k lớn hơn sẽ tiết kiệm nhiều hơn,
vì sao không sử dụng *k* lớn?

Từ điển không phân đoạn

- Giả sử xác suất sử dụng từ là đồng nhất,
- Số so sánh trung bình để tìm một từ là (1+2·2+4·3+4)/8 ~2.6



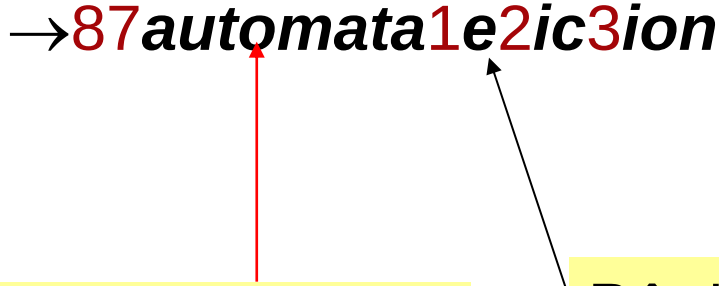
Từ điển có phân đoạn



- Tìm kiếm nhị phân trên khối
- Tìm kiếm tuần tự trong mỗi khối.
  - Với khối 4 từ, số so sánh trung bình = (1+2·2+2·3+2·4+5)/8 = 3 so sánh

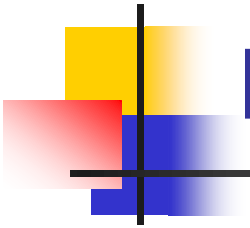
Chuỗi ký tự dài, phân đoạn và Front-coding

- Đặc điểm: Những từ đã sắp xếp thường có phần bắt đầu giống nhau
- Front-coding: Trong khối, lưu hoàn chỉnh từ đầu tiên và phần khác biệt của các từ tiếp theo
  8***automata***8***automate***9***automatic***10***automation***



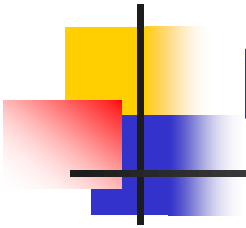
Phần đầu ***automat***

Độ dài phần mở rộng ngoài ***automat.***



Nội dung chính

- Các quy luật phân bố từ
- Nén từ điển
- Nén danh sách mã văn bản



Nén danh sách mã văn bản

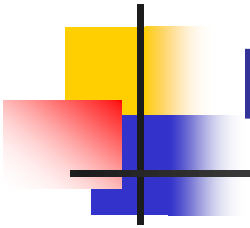
- Xét trường hợp đơn giản nhất khi chỉ lưu mã văn bản theo trật tự tăng dần trong danh sách thẻ định vị.
  - Ví dụ, mô hình Boolean.
- Mục đích nén:
  - Giảm kích thước danh sách thẻ định vị;
  - Lưu số lượng lớn thẻ định vị trong bộ nhớ;
  - Giảm thời gian đọc từ ổ đĩa.

Biểu diễn nhị phân của mã văn bản

- Số bit tối ưu để biểu diễn mã văn bản là $\log_2$(DocID) bits:
  - Nếu sử dụng lượng bit cố định (>=$\log_2$(max(docID))) sẽ lãng phí bộ nhớ khi lưu mã số nhỏ;
  - Cần thay đổi số lượng bit phù hợp với mã văn bản.

Danh sách khoảng cách

- Các mã văn bản trong danh sách được lưu theo thứ tự tăng dần, ví dụ:
  - ***Máy tính***: 33,47,154,159,202 …
- Có thể thay bằng khoảng cách
  - 33,14,107,5,43 …



Mã VB

- Mã VB: Variable Bytes Code. Là phương pháp mã hóa sử dụng số byte thay đổi phù hợp với kích thước mã văn bản.
- Mã hóa:
  - Gom nhóm 7 bits,
  - Sử dụng 1 byte để lưu một nhóm,
  - Đặt bit cao nhất (bit c) của byte phải nhất bằng 1, với các bytes còn lại đặt c = 0,
  - Dãy byte thu được là mã VB của khoảng cách G.

Mã VB (2)

```
VBENCODENUMBER(n)
1  bytes ← ⟨⟩
2  while true
3  do PREPEND(bytes, n mod 128)
4     if n < 128
5        then BREAK
6     n ← n div 128
7  bytes[LENGTH(bytes)] += 128
8  return bytes
```

```
VBDECODE(bytestream)
1  numbers ← ⟨⟩
2  n ← 0
3  for i ← 1 to LENGTH(bytestream)
4  do if bytestream[i] < 128
5        then n ← 128 × n + bytestream[i]
6        else n ← 128 × n + (bytestream[i] − 128)
7             APPEND(numbers, n)
8             n ← 0
9  return numbers
```

Ví dụ

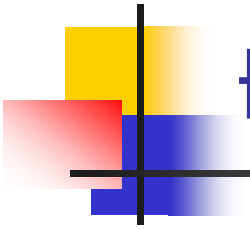
docIDs	824	829	215406
gaps		5	214577
VB code	00000110 10111000	10000101	00001101 00001100 10110001

Danh sách thẻ định vị được lưu như dãy byte liên tiếp

000001101011100010000101000011010000110010110001

*Quan trọng: Có thể giải mã VB theo tiến trình đọc từ ổ đĩa (đọc tới đâu giải nén tới đó).

Với khoảng cách nhỏ (5), VB sử dụng cả một bytes.



Đơn vị mã hóa tối ưu cho mã VB

- Nếu sử dụng đơn vị mã hóa lớn sẽ lãng phí bộ nhớ đối với các khoảng cách nhỏ, ngược lại nếu sử dụng đơn vị nhỏ sẽ lãng phí bộ nhớ đối với giá trị lớn.
  - Có thể sử dụng các đơn vị mã hóa khác: 32 bits, 16 bits, 4 bits tùy theo đặc điểm phân bố giá trị số;
  - Hoặc gom các giá trị nhỏ thành giá trị lớn hơn, v.v.

Mã Unary code

- Biểu diễn số *n* như chuỗi *n* số 1 thêm số 0 ở cuối.
- Unary code của 3 là 1110.
- Unary code của 40 là

11111111111111111111111111111111111110 .

- Unary code của 80 là:

11111111111111111111111111111111111111111111111
11111111111111111111111111110

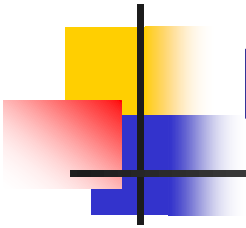
Hiệu quả ứng dụng?

Mã Gamma

- Biểu diễn một khoảng cách G bằng *offset* và *length*
- *offset là* mã nhị phân của *G* loại bỏ bit đứng đầu
  - Ví dụ 13 = 1101_2 ; offset(13) = 101
- length là Unary Code của độ dài của offset
  - Với 13: offset = 101, length = 1110.
- Mã Gamma = *length* + *offset*
  - Mã Gamma của 13 là 1110101

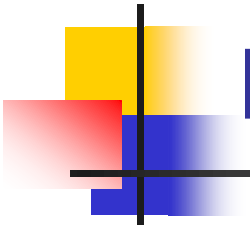
Ví dụ mã Gamma

number	length	offset	γ-code
0			none
1	0		0
2	10	0	10,0
3	10	1	10,1
4	110	00	110,00
9	1110	001	1110,001
13	1110	101	1110,101
24	11110	1000	11110,1000
511	111111110	11111111	111111110,11111111
1025	11111111110	0000000001	11111111110,0000000001



Mã Gamma vs. mã VB

- Đều có thể giải mã cùng tiến trình đọc dữ liệu.
- Mã Gamma có tỉ lệ nén ổn định cho mọi giá trị mã văn bản và nén tốt hơn mã VB;
- Mã Gamma sử dụng các thao tác trên bits nên chậm hơn mã VB.



Bài tập 11.1

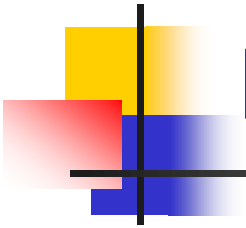
Cho danh sách mã văn bản sau:

1, 3, 10, 120, 121

a) Hãy xác định danh sách khoảng cách;

b) Hãy mã hóa kết quả mục a bằng mã VB, đơn vị mã hóa là 8 bits;

c) Hãy mã hóa kết quả mục a bằng mã Gamma.



Bài tập 11.2

a) Dãy bits sau là mã VB của danh sách khoảng cách, đơn vị mã hóa là 4 bits (nibbles):

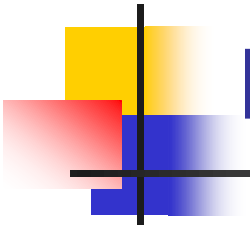
1010 0001 1011 0101 1000

Hãy xác định danh sách mã văn bản ban đầu.

b) Dãy bits sau là mã Gamma của danh sách khoảng cách:

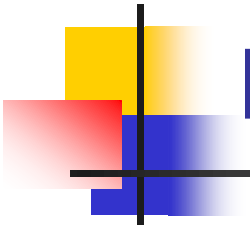
110011110000101

Hãy xác định danh sách mã văn bản ban đầu.



Bài tập 11.3

Hãy chứng minh kích thước bộ từ vựng là hữu hạn trên cơ sở luật Zipf và vô hạn trên cơ sở luật Heap. Chúng ta có thể suy diễn luật Heap từ luật Zipf hay không?



Bài tập 11.4

Giả sử cho một bộ từ điển với 100 000 từ. Kích thước cố định cho một từ là 20 bytes, sử dụng 4 bytes để lưu tần suất văn bản, và 4 bytes để lưu con trỏ.

Hãy ước lượng kích thước từ điển nếu chia khối trong hai trường hợp, k = 8 và k = 16.