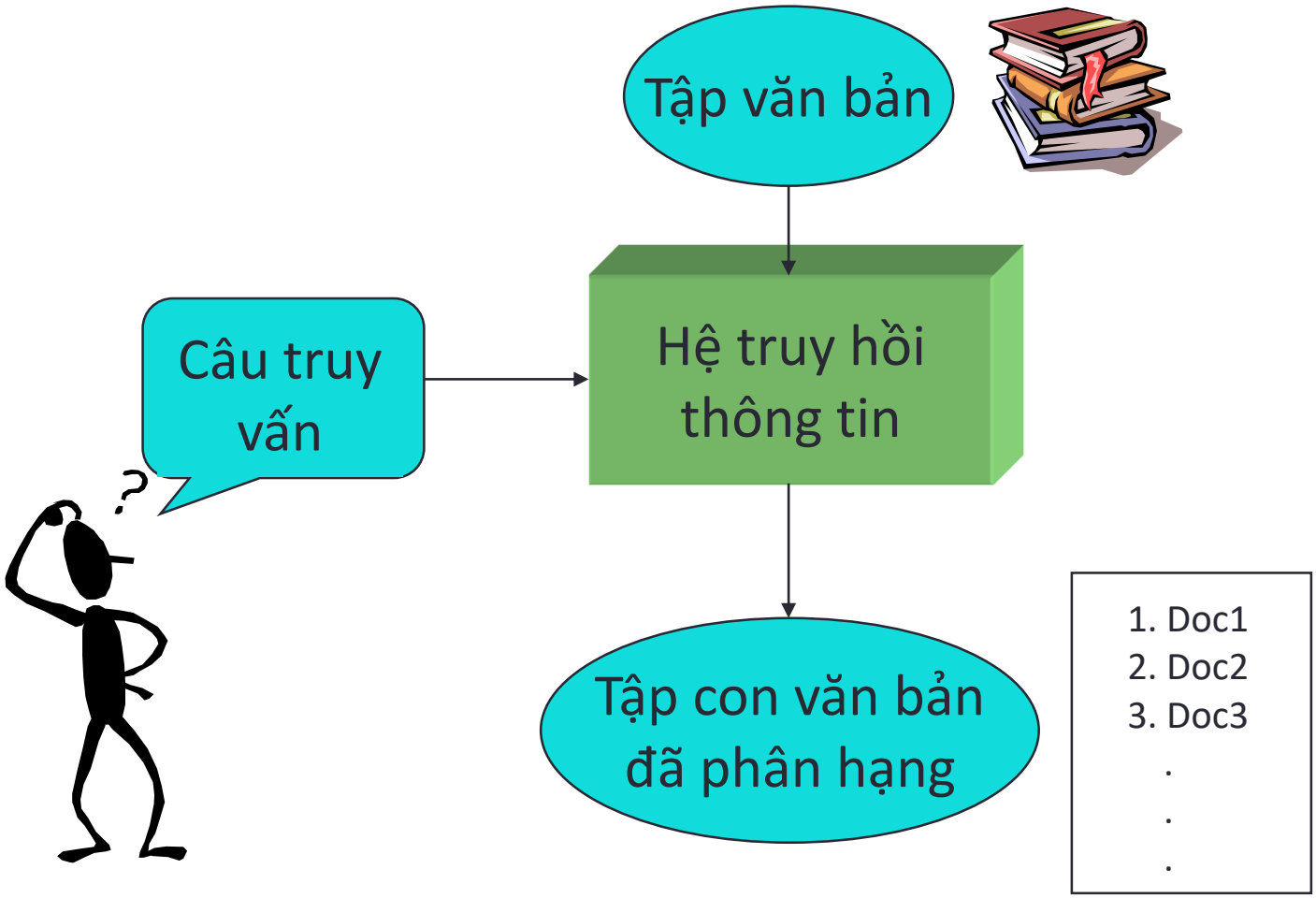
Tổng quan truy hồi thông tin & truy hồi Boole

Nguyễn Mạnh Hiển
hiennm@tlu.edu.vn

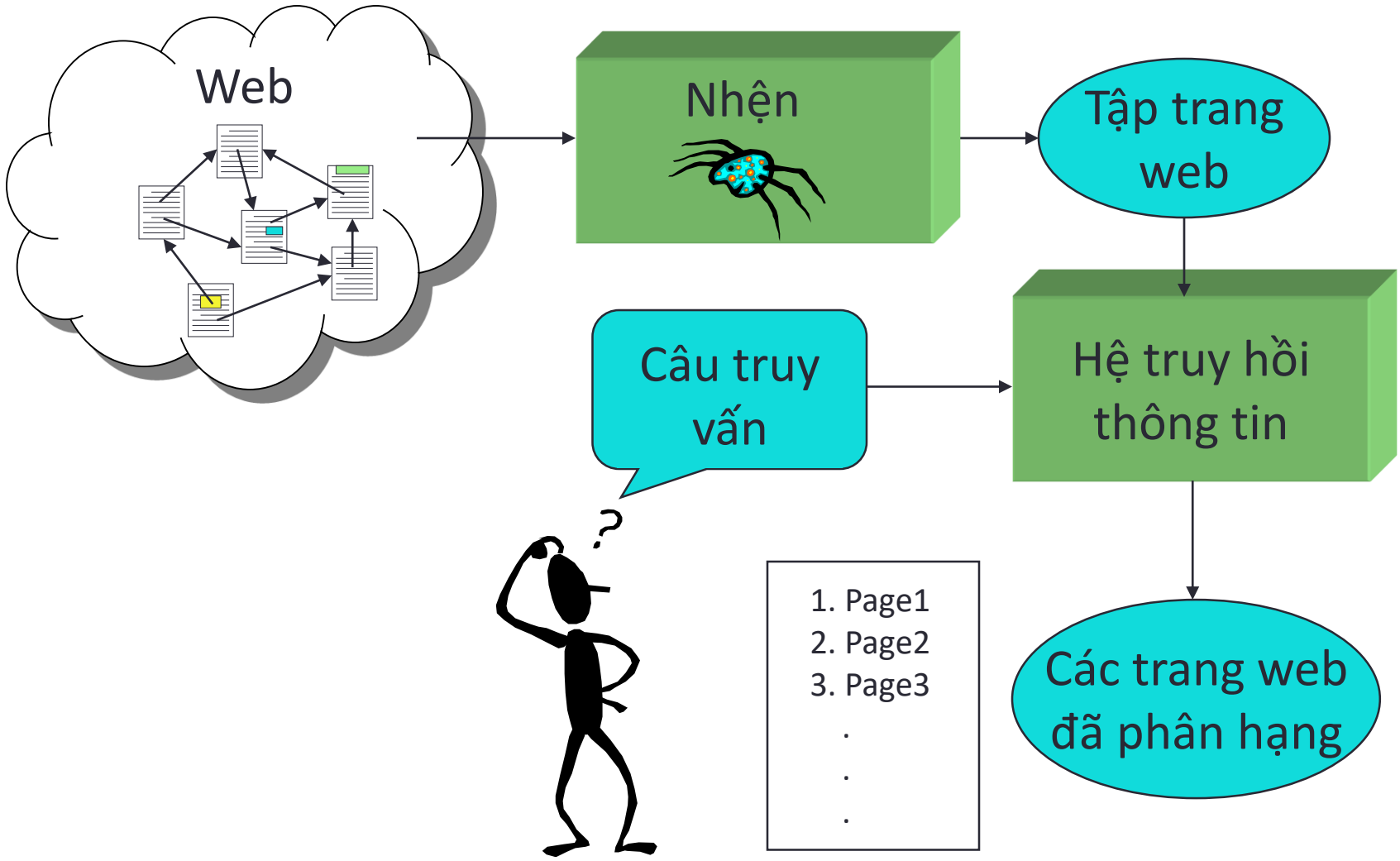
Tác vụ truy hồi thông tin

- Cho:
  - **Tập văn bản** (document collection).
  - **Câu truy vấn** (query) của người dùng dưới dạng xâu ký tự gồm một hoặc nhiều từ.
- Tìm:
  - Tập con văn bản **phù hợp** (relevant) với câu truy vấn, trong đó các **văn bản** (document) thường được phân hạng theo độ phù hợp từ cao xuống thấp.
- Ví dụ:
  - Tìm thông tin trên web dùng Google.
  - Tìm email trong một ứng dụng quản lý email, như Gmail và Thunderbird.

Hệ truy hồi thông tin



Hệ tìm kiếm web



Dữ liệu không có cấu trúc

- Hệ truy hồi thông tin làm việc với **dữ liệu không có cấu trúc** (unstructured data):
  - Không có cấu trúc rõ ràng về mặt ngữ nghĩa và không dễ cho máy tính xử lý.
  - Văn bản (text) và ảnh (image) là những dạng dữ liệu không có cấu trúc phổ biến.
  - Môn học này tập trung vào **dữ liệu văn bản** (text data).
- Dữ liệu không có cấu trúc ngược với **dữ liệu có cấu trúc** (structured data) hay gặp trong cơ sở dữ liệu quan hệ.
  - Ví dụ, dữ liệu về một sản phẩm có cấu trúc rõ ràng, gồm mã sản phẩm, tên sản phẩm, hãng sản xuất, ngày sản xuất, đơn giá, số lượng, v.v…

Dữ liệu không có cấu trúc (tiếp)

- Trong nhiều trường hợp, dữ liệu văn bản không phi cấu trúc hoàn toàn.
  - Ví dụ, một trang web có thể bao gồm phần tiêu đề, phần thân và phần ghi chú cuối trang; phần thân lại được chia tiếp thành các đoạn văn.
- Hệ truy hồi thông tin có thể hỗ trợ tìm kiếm **bán cấu trúc** (semi-structured).
  - Ví dụ, tìm các văn bản với phần tiêu đề chứa từ Java và phần thân chứa từ threading.

Hỗ trợ duyệt và lọc các văn bản

- Hệ truy hồi thông tin có thể hỗ trợ người dùng tổ chức, duyệt và lọc các văn bản (bên cạnh việc tìm kiếm với câu truy vấn).
- **Phân cụm văn bản** (text clustering): Chia các văn bản thành các **nhóm/cụm** (cluster) có nội dung tương tự nhau.
- **Phân loại văn bản** (text classification): Phân các văn bản chưa biết chủ đề vào các chủ đề cho trước.
  - Các chủ đề đó còn được gọi là các **lớp** (class).

Phân loại hệ truy hồi thông tin theo kích cỡ (1)

- **Tìm kiếm web (cỡ lớn):**
  - Tìm trên hàng tỉ văn bản (trang web) được lưu trên hàng triệu máy tính.
  - Những vấn đề riêng cần giải quyết:
    - Thu thập các văn bản về một chỗ để **lập chỉ mục** (indexing).
    - Hệ thống phải làm việc hiệu quả trên cỡ lớn.
    - Xử lý những mặt đặc thù của web, như khai thác thông tin siêu văn bản (hypertext) và chống các thủ thuật bẩy thứ hạng trang web của chủ trang web.

Phân loại hệ truy hồi thông tin theo kích cỡ (2)

- **Tìm kiếm cá nhân (cỡ nhỏ):**
  - Các hệ điều hành có hỗ trợ tìm kiếm tài liệu.
  - Các chương trình email, ngoài tìm kiếm, còn cung cấp chức năng phân loại văn bản dưới dạng bộ lọc thư rác.
  - Những vấn đề riêng cần giải quyết:
    - Xử lý nhiều kiểu tài liệu trên máy tính cá nhân, như .docx, .xlsx, .pptx, .odt, .txt, .pdf, .htm, v.v…
    - Hệ thống tìm kiếm phải gọn nhẹ (về mặt khởi động, xử lý, tiêu thụ không gian đĩa) để tránh gây phiền nhiễu cho người dùng.

Phân loại hệ truy hồi thông tin theo kích cỡ (3)

- **Tìm kiếm doanh nghiệp (cỡ trung):**
  - Ví dụ, tìm kiếm trên các văn bản nội bộ của một công ty, tìm trong cơ sở dữ liệu bằng sáng chế hoặc tìm các bài báo khoa học trong một lĩnh vực chuyên môn nào đó.
  - Những văn bản này thường được lưu trên hệ thống file nằm trên một hoặc nhiều máy tính.
  - Dữ liệu văn bản cũng được lưu trữ nhiều trong các cơ sở dữ liệu quan hệ.

Ví dụ về truy hồi thông tin

- Cho tập văn bản gồm các vở kịch của Shakespeare.
- Yêu cầu: Tìm những vở kịch thỏa mãn câu truy vấn Brutus AND Caesar AND NOT Calpurnia.
- Cách tìm đơn giản:
  - Quét qua tất cả các vở kịch.
  - Ghi chú lại những vở kịch chứa cả hai từ Brutus và Caesar.
  - Loại bỏ ngay vở kịch nào chứa từ `Calpurnia`.
- Quét tuyến tính (linear scan) như thế rất hiệu quả với những tập văn bản có kích cỡ vừa phải.
  - Tập vở kịch Shakespeare có tổng cộng chưa đến một triệu từ.

Ví dụ về truy hồi thông tin (tiếp)

- Trên thực tế, thường có thêm các yêu cầu sau:
  - Xử lý nhanh các tập văn bản lớn chứa hàng tỉ đến hàng ngàn tỉ từ.
  - Cho phép **đối sánh từ** (matching) linh hoạt hơn.
    - Ví dụ: Câu truy vấn có thể dưới dạng Romans NEAR countrymen, trong đó NEAR có thể định nghĩa là "trong phạm vi 5 từ" hoặc "trong cùng một câu".
  - Cho phép phân hạng các văn bản trả về.
- Để tránh lặp lại việc quét tuyến tính tập văn bản mỗi khi có câu truy vấn mới, ta phải xây dựng trước một cấu trúc dữ liệu gọi là **chỉ mục ngược** (inverted index) từ tập văn bản đã cho.

Ma trận từ-văn bản

Các vở kịch

từ xuất hiện

từ vắng mặt

Các từ

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	…
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
…							

Nhìn theo hàng, ta có một véctơ nhị phân ứng với mỗi từ, cho biết từ đang xét xuất hiện trong những văn bản (vở kịch) nào.

Truy hồi Boole

- Mô hình truy hồi Boole:
  - Câu truy vấn là một biểu thức Boole, trong đó các từ được kết hợp với nhau bằng các phép lôgic gồm AND, OR và NOT.
  - Mỗi văn bản được xem là một tập từ.
- Để trả lời câu truy vấn Brutus AND Caesar AND NOT Calpurnia:
  - Từ ma trận từ-văn bản, lấy ra véctơ cho ba từ Brutus (110100), Caesar (110111) và Calpurnia (010000).
  - Đảo bit véctơ thứ ba (thực hiện phép NOT) được 101111.
  - Thực hiện phép AND trên ba véctơ:

    110100 AND 110111 AND 101111 = 100100

$\rightarrow$ Câu trả lời là hai vở kịch "Antony and Cleopatra" và "Hamlet" (tham khảo ma trận từ-văn bản trong slide trước).

“Antony and Cleopatra” và “Hamlet”

Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to Domitius Enobarbus]: Why, Enobarbus,
When Antony found Julius Caesar dead,
He cried almost to roaring; and he wept
When at Philippi he found Brutus slain.

Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius Caesar: I was killed i’ the
Capitol; Brutus killed me.

Đánh giá hệ truy hồi thông tin

- Người dùng diễn tả nhu cầu thông tin bằng một câu truy vấn.
- Một văn bản (trả về bởi hệ truy hồi thông tin) là **phù hợp** (relevant) nếu người dùng thấy văn bản đó chứa thông tin có giá trị cho nhu cầu thông tin cá nhân của người dùng.
- Truy hồi Boole thiếu thực tế vì nó chỉ quan tâm các từ xuất hiện hay vắng mặt trong các văn bản.
  - Ví dụ, nếu người dùng quan tâm đến chủ đề pipeline leaks (rò rỉ đường ống), thì sẽ quan tâm đến văn bản chứa các từ pipeline rupture (vỡ đường ống), mặc dù văn bản đó không chứa từ leaks.

Đánh giá hệ truy hồi thông tin (tiếp)

- **Độ chính xác** (precision): Có bao nhiêu văn bản phù hợp trong những văn bản đã được trả về?
  - Ví dụ: Nếu hệ đã trả về 100 văn bản và trong số đó có 70 văn bản phù hợp, thì độ chính xác là 70%.
- **Độ thu hồi** (recall): Có bao nhiêu văn bản phù hợp đã được trả về so với tất cả văn bản phù hợp (gồm cả trả về và không trả về)?
  - Ví dụ: Nếu có tất cả 100 văn bản phù hợp trong tập văn bản và hệ đã trả về được 60 văn bản như vậy, thì độ thu hồi là 60%.

Kịch bản thực tế hơn

- Giả sử:
  - Có 1 triệu văn bản.
  - Mỗi văn bản dài khoảng 1000 từ (2–3 trang sách).
  - Mỗi từ tốn trung bình 6 byte (bao gồm cả dấu cách và dấu câu) $\rightarrow$ tập văn bản có kích cỡ 6 GB.
  - Có khoảng 500 nghìn từ riêng biệt trong tập văn bản.
- Ma trận từ-văn bản có kích cỡ 500K x 1M, tức là chứa 500 tỉ số 0/1 $\rightarrow$ quá lớn (500 GB) để lưu trọn vào bộ nhớ máy tính.

Kịch bản thực tế hơn (tiếp)

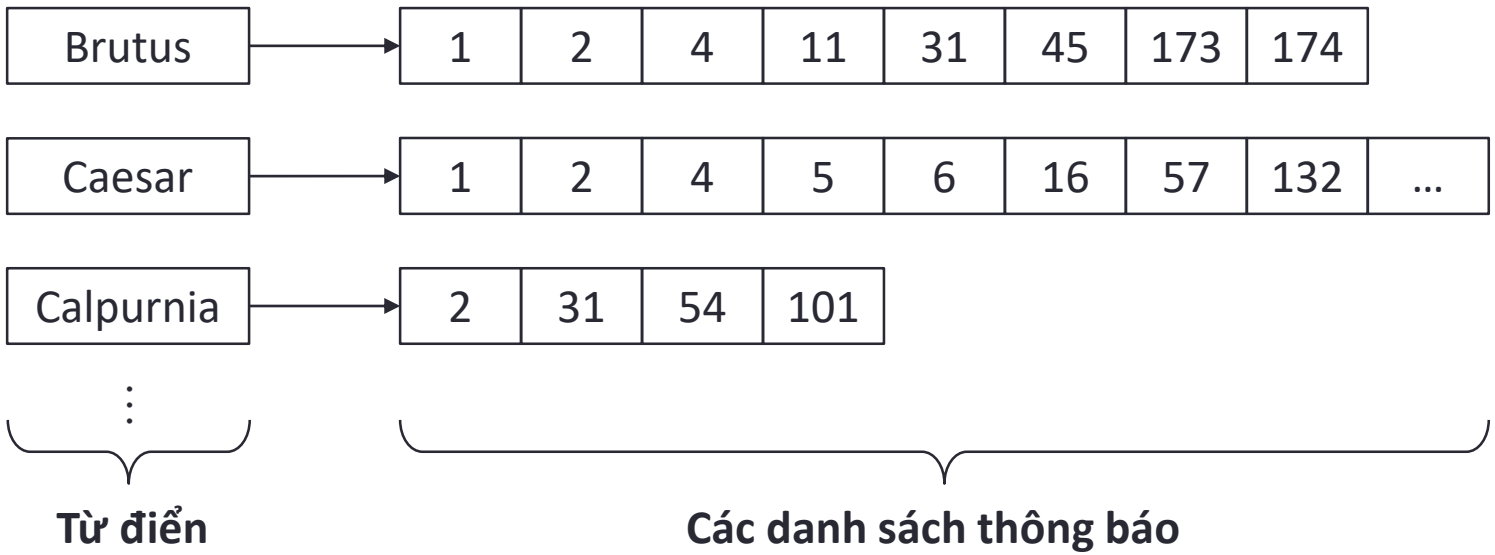
- Nhận xét:
  - Ma trận từ-văn bản cực thưa (rất ít phần tử khác 0).
  - Mỗi văn bản dài 1000 từ, tức là mỗi cột của ma trận có không quá 1000 số 1.

    → Ma trận có không quá 1M (cột) x 1K = 1 tỉ số 1.

    → Có không quá 1 tỉ / 500 tỉ = 0,2% các ô bằng 1; nói cách khác, có ít nhất 99,8% các ô trong ma trận bằng 0.
- Cách làm tốt hơn là chỉ lưu trữ các vị trí 1 trong ma trận.

  → Đó chính là ý tưởng về **chỉ mục ngược** (inverted index), trong đó mỗi từ trỏ tới một danh sách chỉ gồm các văn bản chứa từ đó.

Chỉ mục ngược



- **Từ điển** (dictionary) thường nhỏ gọn thì lưu trong bộ nhớ; các **thông báo** (posting) thường lớn thì lưu trên đĩa.
- Mỗi từ trong từ điển trỏ tới một danh sách thông báo (đã được sắp xếp tăng dần).
- Mỗi thông báo gồm mã (ID) của một văn bản có chứa từ đang xét.

Minh họa xây dựng chỉ mục ngược

Bước 1. Thu thập các văn bản cần lập chỉ mục:

Friends, Romans, countrymen.	So let it be with Caesar	…

Bước 2. Tách từ và biến mỗi văn bản thành một danh sách từ:

Friends	Romans	countrymen	So	…

Bước 3. Chuẩn hóa các từ dùng các phép xử lý ngôn ngữ:

friend	roman	countryman	so	…

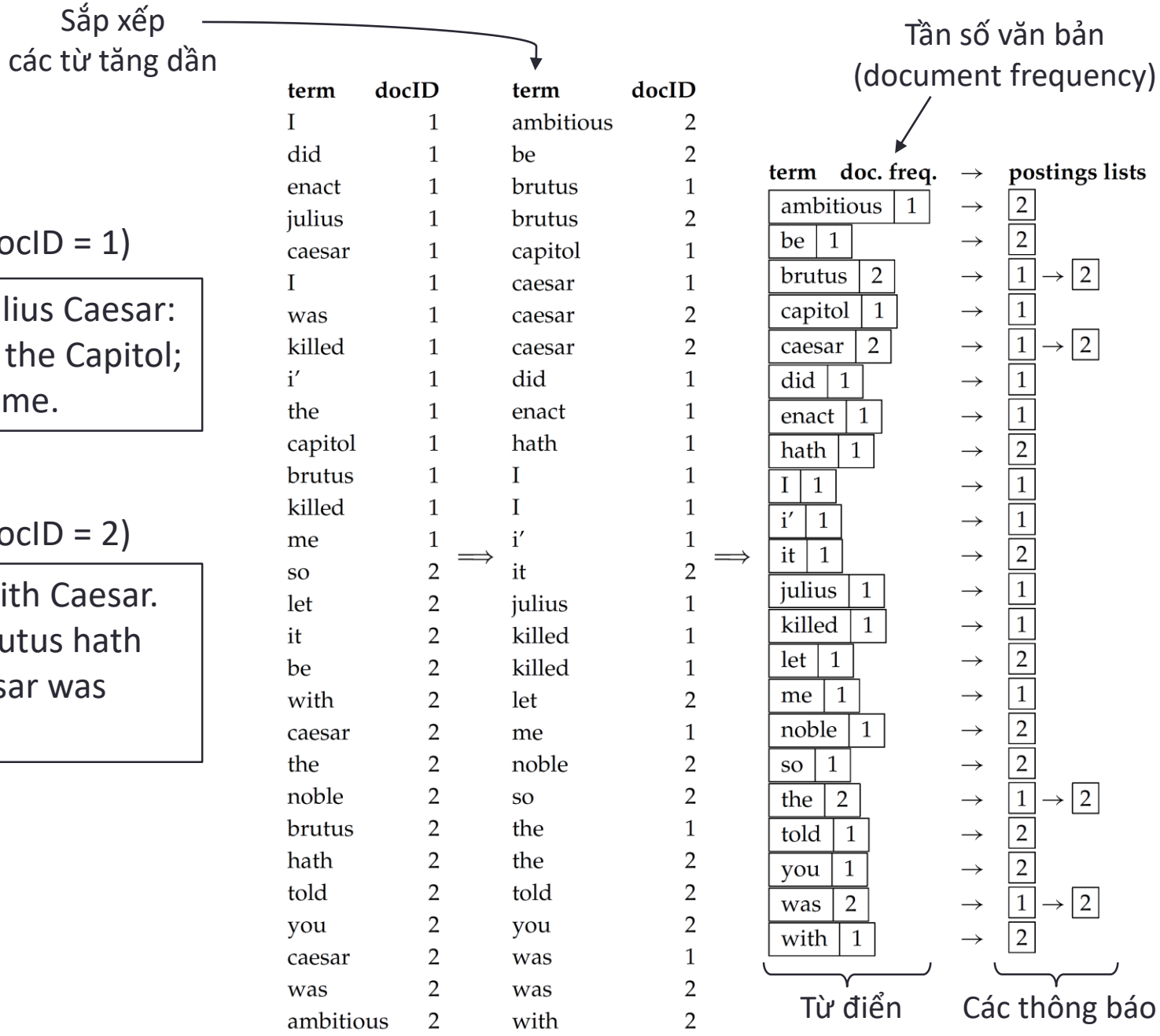
Bước 4. Lập chỉ mục ngược gồm từ điển và các danh sách thông báo → xem slide tiếp theo…

Văn bản 1 (docID = 1)

I did enact Julius Caesar:
I was killed i' the Capitol;
Brutus killed me.

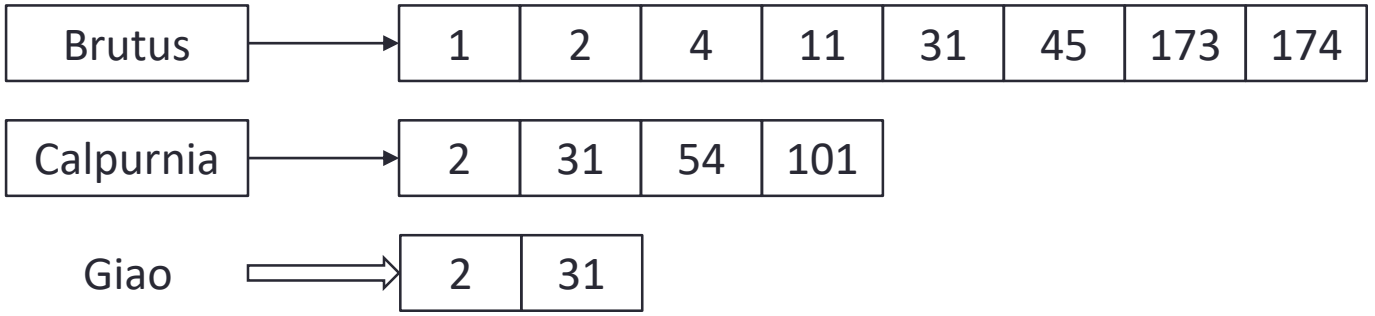
Văn bản 2 (docID = 2)

So let it be with Caesar.
The noble Brutus hath
told you Caesar was
ambitious:



Xử lý câu truy vấn Boole

- Xét câu truy vấn: Brutus AND Calpurnia
- Các bước xử lý trên chỉ mục ngược:
  - Định vị từ Brutus trong từ điển.
  - Lấy về danh sách thông báo của từ Brutus.
  - Định vị từ Calpurnia trong từ điển.
  - Lấy về danh sách thông báo của từ Calpurnia.
  - Lấy giao của hai danh sách thông báo.



Thuật toán lấy giao của hai danh sách thông báo

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

Thuật toán mất thời gian $O(n_1+n_2)$, trong đó n_1 và n_2 là chiều dài của các danh sách thông báo.

Tối ưu hóa câu truy vấn (1)

- Là tìm cách xử lý câu truy vấn sao cho tốn ít công sức nhất.
- Để tối ưu hóa câu **truy vấn hội** (trong đó các từ kết hợp với nhau bằng phép AND):
  - Xử lý các từ theo thứ tự tăng dần của tần số văn bản (số văn bản chứa mỗi từ): Lấy giao của hai danh sách đầu tiên, rồi lại giao với danh sách thứ ba, và cứ như vậy cho đến hết.
  - Chú ý: Giao của hai danh sách không dài hơn danh sách ngắn hơn.
  - Nếu ta bắt đầu bằng cách lấy giao của hai danh sách thông báo ngắn nhất, thì tất cả các kết quả trung gian đều không dài hơn danh sách thông báo ngắn nhất.

    $\rightarrow$ Có vẻ như lượng công việc phải làm là ít nhất.

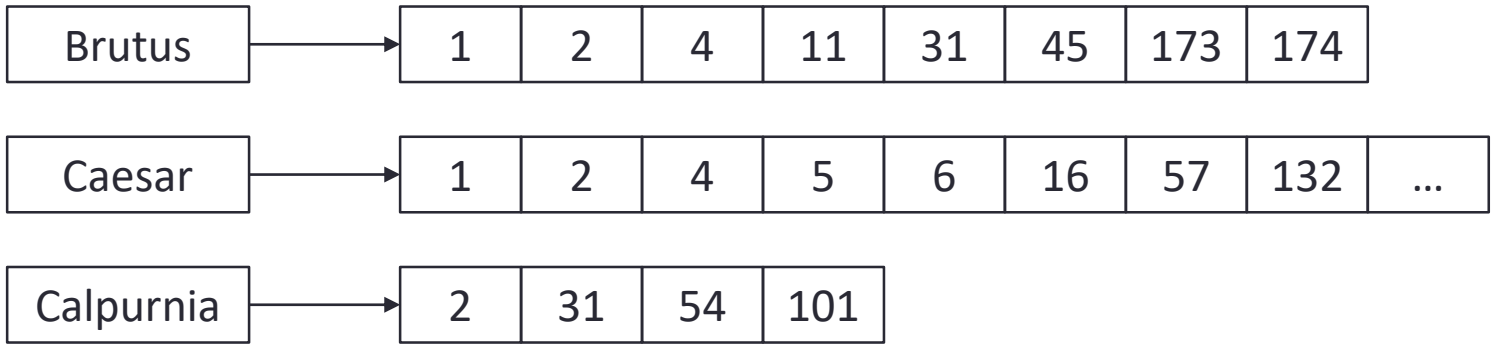
Tối ưu hóa câu truy vấn (2)

- Xét câu truy vấn hội:

  Brutus AND Caesar AND Calpurnia

- Để tối ưu hóa câu truy vấn đó, ta sẽ xử lý theo trình tự sau:

  (Calpurnia AND Brutus) AND Caesar



Tối ưu hóa câu truy vấn (3)

- Xét câu truy vấn tổng quát hơn:

  (madding OR crowd) AND (ignoble OR strife) AND (killed OR slain)

  - Ước lượng kích cỡ của mỗi phép OR (phép tuyển) bằng tổng tần số văn bản của mỗi từ trong phép OR đó.
  - Sau đó, xử lý các phép AND (phép hội) theo trình tự tăng dần của kích cỡ các phép OR.
- Trên thực tế, người dùng thường gõ vào câu truy vấn hội thuần túy, gồm một chuỗi từ được kết hợp với nhau bằng các phép AND (trên Google thì không cần gõ tường minh từ AND).

Thuật toán xử lý câu truy vấn hội

```
INTERSECT(⟨t_1, ..., t_n⟩)
1  terms ← SORTBYINCREASINGFREQUENCY(⟨t_1, ..., t_n⟩)
2  result ← postings(first(terms))
3  terms ← rest(terms)
4  while terms ≠ NIL and result ≠ NIL
5  do result ← INTERSECT(result, postings(first(terms)))
6     terms ← rest(terms)
7  return result
```

- Danh sách kết quả trung gian “result” nằm trong bộ nhớ.
- Danh sách “postings(first(terms))” được đọc từ đĩa vào bộ nhớ trước khi lấy giao với danh sách “result”.

Những yêu cầu vượt khỏi mô hình Boole

- Từ điển cần chịu đựng được lỗi chính tả.
- Hỗ trợ câu truy vấn cụm từ, như "operating system".
- Hỗ trợ câu truy vấn lân cận, như Gates NEAR Microsoft.
- Mô hình Boole chỉ giữ thông tin xuất hiện hay vắng mặt của các từ trong các văn bản, nhưng thường ta muốn gán trọng số lớn hơn cho văn bản chứa một từ nhiều lần hơn so với các văn bản khác → cần lưu thêm thông tin **tần số từ** (term frequency) vào danh sách thông báo.
- Mô hình Boole chỉ trả về tập văn bản khớp chính xác với câu truy vấn, nhưng thường ta muốn **phân hạng** (rank) các kết quả trả về → cần cơ chế **chấm điểm** (scoring) các văn bản để phản ánh độ phù hợp của chúng với câu truy vấn.

Bài tập

1. Vẽ chỉ mục ngược cho tập văn bản sau đây:

Doc1: new home sales top forecasts

Doc2: home sales rise in july

Doc3: increase in home sales in july

Doc4: july new home sales rise

Bài tập

2. Xét tập văn bản sau đây:

Doc1: breakthrough drug for schizophrenia

Doc2: new schizophrenia drug

Doc3: new approach for treatment of schizophrenia

Doc4: new hopes for schizophrenia patients

a. Vẽ ma trận từ-văn bản.

b. Vẽ chỉ mục ngược.

c. Kết quả trả về cho những câu truy vấn sau đây là gì?

1. schizophrenia AND drug
2. for AND NOT (drug OR approach)

Bài tập

3. Cho kích thước của các danh sách thông báo trong chỉ mục ngược như sau:

Từ	Kích thước danh sách thông báo
eyes	213.312
kaleidoscope	87.009
marmalade	107.913
skies	271.658
tangerine	46.653
trees	316.812

Hãy tối ưu hóa (đề xuất trình tự xử lý) câu truy vấn sau đây:

(tangerine OR trees) AND (marmalade OR skies) AND (kaleidoscope OR eyes)

Bài tập

4. Xét một câu truy vấn hội, hỏi việc xử lý các danh sách thông báo theo thứ tự tăng dần của kích thước danh sách có đảm bảo tối ưu hay không? Giải thích vì sao tối ưu, hoặc nêu một ví dụ cho thấy cách xử lý đó không tối ưu?

5. Viết một thuật toán lấy hợp của hai danh sách thông báo (theo kiểu thuật toán lấy giao đã học) để xử lý câu truy vấn x OR y.

6. Viết một thuật toán xử lý câu truy vấn x AND NOT y.