CHƯƠNG 4
CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN LƯU TRỮ NGOÀI

Bộ môn CÔNG NGHỆ PHẦN MỀM
Khoa Công nghệ Thông tin & Truyền thông
ĐẠI HỌC CẦN THƠ

NỘI DUNG

- Mục tiêu.
- Mô hình và đánh giá các xử lý ngoài.
- Sắp xếp ngoài.
- Lưu trữ thông tin trong tập tin:
  - Tập tin tuần tự
  - Tập tin bảng băm
  - Tập tin chỉ mục
  - Tập tin B-cây

MỤC TIÊU

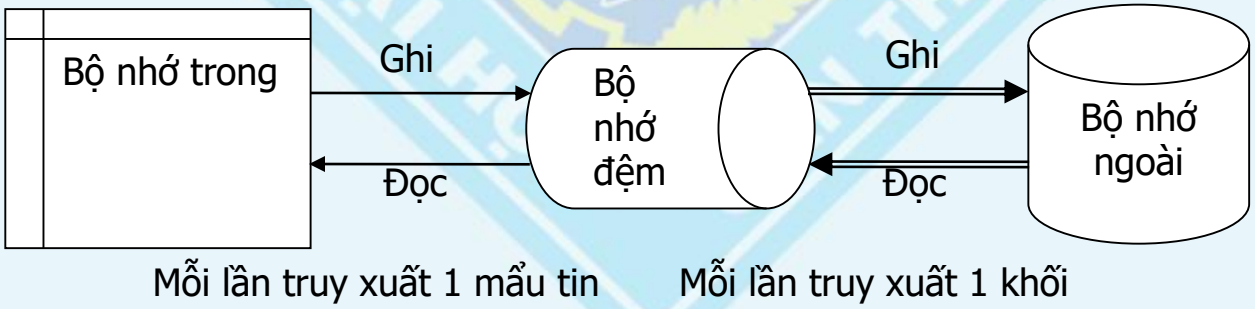
- Biết mô hình xử lý ngoài.
- Hiểu tiêu chuẩn để đánh giá giải thuật xử lý ngoài. Vận dụng trong việc cải tiến giải thuật xử lý ngoài.
- Hiểu giải thuật sắp xếp trộn để sắp xếp ngoài và phương pháp cải tiến tốc độ sắp xếp trộn.
- Hiểu cách thức tổ chức lưu trữ và các giải thuật tìm kiếm, xen, xoá thông tin trên các tập tin tuần tự, tập tin chỉ mục, tập tin bảng băm.
- Vận dụng được cách thức tổ chức lưu trữ và các giải thuật tìm kiếm, xen, xoá thông tin trên tập tin B-cây.

Tại sao phải xử lí ngoài

- Trong các giải thuật mà chúng ta đã đề cập từ trước tới nay, chúng ta đã giả sử rằng số lượng các dữ liệu vào là khá nhỏ để có thể chứa hết ở bộ nhớ trong (main memory).
- Nhưng điều gì sẽ xảy ra nếu ta muốn xử lý phiếu điều tra dân số toàn quốc hay thông tin về quản lý đất đai cả nước chẳng hạn?
- Trong các bài toán như vậy, số lượng dữ liệu vượt quá khả năng lưu trữ của bộ nhớ trong.
- Để có thể giải quyết các bài toán đó chúng ta phải dùng bộ nhớ ngoài để lưu trữ và xử lý.
- Các thiết bị lưu trữ ngoài như băng từ, đĩa từ đều có khả năng lưu trữ lớn nhưng đặc điểm truy nhập hoàn toàn khác với bộ nhớ trong.
- Chúng ta cần tìm các cấu trúc dữ liệu và giải thuật thích hợp cho việc xử lý dữ liệu lưu trữ trên bộ nhớ ngoài

Mô hình xử lí ngoài

- Hệ điều hành chia bộ nhớ ngoài thành các khối (block) có kích thước bằng nhau, kích thước này thay đổi tùy thuộc vào hệ điều hành nhưng nói chung là từ 512 bytes đến 4096 bytes.
- Có thể xem một tập tin bao gồm nhiều mẩu tin được lưu trong các khối.
- Mỗi khối lưu một số nguyên vẹn các mẩu tin.
- Kiểu dữ liệu tập tin là kiểu thích hợp nhất cho việc biểu diễn dữ liệu được lưu trong bộ nhớ ngoài.



Đánh giá các giải thuật xử lý ngoài

- Đối với bộ nhớ ngoài thì thời gian tìm một khối để đọc vào bộ nhớ trong là rất lớn so với thời gian thao tác trên dữ liệu trong khối đó.
- Chúng ta tập trung vào việc xét số lần đọc khối vào bộ nhớ trong và số lần ghi khối ra bộ nhớ ngoài, ta gọi chung là phép truy xuất khối (block access).
- Nếu số lần truy xuất khối ít thì giải thuật có hiệu quả.
- Để cải tiến giải thuật, ta không thể tìm cách tăng kích thước một khối (Vì kích thước các khối là cố định) mà chúng ta phải tìm cách giảm số lần truy xuất khối.

Sắp xếp ngoài

- Sắp xếp ngoài là sắp xếp dữ liệu được tổ chức thành một tập tin lưu trong bộ nhớ ngoài.
- Mỗi tập tin bao gồm nhiều mẩu tin, mỗi mẩu tin bao gồm nhiều trường, trong đó có một trường khoá.
- Tương tự như sắp xếp trong, sắp xếp ngoài là sự tổ chức lại các mẩu tin sao cho các khóa của chúng được sắp thứ tự tương ứng với quy luật sắp xếp.

Sắp xếp trộn: Khái niệm về đường

- Đường độ dài k là một tập hợp k mẩu tin đã được sắp thứ tự theo khoá.
- Cho tập tin chứa các mẩu tin $r_1, r_2, ..., r_n$, ta nói tập tin được tổ chức thành đường có độ dài k nếu ta chia tập tin thành các đoạn k mẩu tin liên tiếp và mỗi đoạn là một đường, đoạn cuối có thể không có đủ k mẩu tin, trong trường hợp này ta gọi đoạn ấy là đuôi (tail).
- **Ví dụ:** Tập tin gồm 14 mẩu tin có khóa là các số nguyên được tổ chức thành 4 đường độ dài 3 và một đuôi có độ dài 2

5	6	9	13	26	27	1	5	8	12	14	17	23	25

Sắp xếp trộn: Giải thuật

- Để sắp xếp tập tin F có n mẩu tin ta sử dụng 4 tập tin F1, F2, G1 và G2.
- Phân phối các mẩu tin của tập tin đã cho F luân phiên vào trong hai tập tin F1 F2. Như vậy hai tập tin này xem như được tổ chức thành các đường độ dài 1.
- Bước 1: Đọc 2 đường, mỗi đường độ dài 1 từ hai tập tin F1, F2 và trộn lại thành đường độ dài 2 và ghi luân phiên vào trong hai tập tin G1, G2. Đổi vai trò của F1 cho G1, F2 cho G2.
- Bước 2: Đọc 2 đường, mỗi đường độ dài 2 từ hai tập tin F1, F2 và trộn lại thành đường độ dài 4 và ghi luân phiên vào trong hai tập tin G1, G2. Đổi vai trò của F1 cho G1, F2 cho G2.
- Quá trình trên cứ tiếp tục và sau i bước thì độ dài của một đường là 2^i. Nếu $2^i \geq n$ thì giải thuật kết thúc, lúc đó tập tin G2 sẽ rỗng và tập tin G1 chứa các mẩu tin đã được sắp.

Sắp xếp trộn: Đánh giá giải thuật

- Giải thuật kết thúc khi $2^i \geq n$, tức là sau i bước với $i \geq \log n$.
- Mỗi bước phải đọc từ 2 tập tin và ghi vào 2 tập tin, mỗi tập tin có trung bình n/2 mẩu tin.
- Giả sử mỗi một khối lưu trữ được b mẩu tin thì mỗi tập tin sẽ được lưu trong n/(2b) khối.
- Mỗi bước cần đọc và ghi trong 4 tập tin, nên mỗi bước truy xuất 2n/b khối mà chúng ta cần logn bước vậy tổng cộng chúng ta cần:

$$\frac{2n}{b}\log n$$

Sắp xếp trộn: Ví dụ

- Cho tập tin F có 23 mẩu tin với khóa là các số nguyên như sau: 2 31 13 5 98 96 10 40 54 85 65 9 30 39 90 13 10 8 69 77 8 10 22.
- Để bắt đầu ta phân phối các mẩu tin của F luân phiên vào hai tập tin F1 và F2 được tổ chức thành các đường có độ dài 1

F1	2	13	98	10	54	65	30	90	10	69	8	22
F2	31	5	96	40	85	9	39	13	8	77	10	

Sắp xếp trộn: Ví dụ: Bước 1

Từ 2 tập tin F1 và F2

F1	2	13	98	10	54	65	30	90	10	69	8	22
F2	31	5	96	40	85	9	39	13	8	77	10	

Trộn các đường độ dài 1 của F1 và F2 được các đường độ dài 2 và ghi luân phiên vào trong hai tập tin G1, G2.

G1	2	31	96	98	54	85	30	39	8	10	8	10
G2	5	13	10	40	9	65	13	90	69	77	22	

Sắp xếp trộn: Ví dụ: Bước 2

Đổi vai trò của F1 và G1, F2 và G2 cho nhau, ta được 2 tập tin F1 và F2 mới:

F1	2	31	96	98	54	85	30	39	8	10	8	10
F2	5	13	10	40	9	65	13	90	69	77	22	

Trộn các đường độ dài 2 của F1 và F2 được các đường độ dài 4 và ghi luân phiên vào trong hai tập tin G1, G2.

G1	2	5	13	31	9	54	65	85	8	10	69	77
G2	10	40	96	98	13	30	39	90	8	10	22	

Sắp xếp trộn: Ví dụ: Bước 3

Đổi vai trò của F1 và G1, F2 và G2 cho nhau, ta được 2 tập tin F1 và F2 mới:

F1	2	5	13	31	9	54	65	85	8	10	69	77
F2	10	40	96	98	13	30	39	90	8	10	22	

Trộn các đường độ dài 4 của F1 và F2 được các đường độ dài 8 và ghi luân phiên vào trong hai tập tin G1, G2.

G1	2	5	10	13	31	40	96	98	8	8	10	10	22	69	77
G2	9	13	30	39	54	65	85	90							

Sắp xếp trộn: Ví dụ: Bước 4

Đổi vai trò của F1 và G1, F2 và G2 cho nhau, ta được 2 tập tin F1 và F2 mới:

F1	2	5	10	13	31	40	96	98	8	8	10	10	22	69	77
F2	9	13	30	39	54	65	85	90							

Trộn các đường độ dài 8 của F1 và F2 được các đường độ dài 16 và ghi luân phiên vào trong hai tập tin G1, G2.

G1	2	5	9	10	13	13	30	31	39	40	54	65	85	90	96	98
G2	8	8	10	10	22	69	77									

Sắp xếp trộn: Ví dụ: Bước 5

Đổi vai trò của F1 và G1, F2 và G2 cho nhau, ta được 2 tập tin F1 và F2 mới:

F1	2	5	9	10	13	13	30	31	39	40	54	65	85	90	96	98

F2	8	8	10	10	22	69	77

Trộn các đường độ dài 16 của F1 và F2 được các đường độ dài 23 và ghi vào trong tập tin G1.

G1	2	5	8	8	9	10	10	10	13	13	22	30	31	39	40	54	65	69	77	85	90	96	98

G2

Sắp xếp trộn cải tiến

- Sắp xếp trộn bắt đầu từ các đường độ dài 1 cho nên phải sau logn bước giải thuật mới kết thúc.
- Để tăng tốc độ, chúng ta phải giảm số bước.
- Mỗi lần đọc vào bộ nhớ trong k mẩu tin, dùng sắp xếp trong để sắp xếp k mẩu tin này và ghi luân phiên vào hai tập tin F1 và F2.
- Như vậy chúng ta bắt đầu sắp xếp trộn với các tập tin được tổ chức thành các đường độ dài k.
- Sau i bước thì độ dài mỗi đường là $k.2^i$. Giải thuật kết thúc khi $k2^i \geq n$ hay $i \geq \log \frac{n}{k}$
- Do đó số phép truy xuất khối sẽ là $\frac{2n}{b} \log \frac{n}{k} < \frac{2n}{b} \log n$

Ví dụ về sắp xếp trộn cải tiến

- SX tập tin F có 23 mẩu tin với khóa là các số nguyên 2 31 13 5 98 96 10 40 54 85 65 9 30 39 90 13 10 8 69 77 8 10 22.
- Ta giả sử bộ nhớ trong có thể chứa được 3 mẩu tin.

F1	2	13	31	10	40	54	30	39	90	8	69	77
F2	5	96	98	9	65	85	8	10	13	10	22	

Ví dụ về sắp xếp trộn cải tiến: Bước 1

Xuất phát sắp xếp trộn từ hai tập tin F1 và F2 đã được tổ chức thành các đường độ dài 3:

F1	2	13	31	10	40	54	30	39	90	8	69	77
F2	5	96	98	9	65	85	8	10	13	10	22	

Trộn các đường độ dài 3 của F1 và F2 được các đường độ dài 6 và ghi luân phiên vào trong hai tập tin G1, G2:

G1	2	5	13	31	96	98	8	10	13	30	39	90
G2	9	10	40	54	65	85	8	10	22	69	77	

Ví dụ về sắp xếp trộn cải tiến: Bước 2

Đổi vai trò của F1 và G1, F2 và G2 cho nhau, ta được hai tập tin F1 và F2 mới:

F1	2	5	13	31	96	98	8	10	13	30	39	90
F2	9	10	40	54	65	85	8	10	22	69	77	

Trộn các đường độ dài 6 của F1 và F2 được các đường độ dài 12 và ghi luân phiên vào trong hai tập tin G1, G2:

G1	2	5	9	10	13	31	40	54	65	85	96	98
G2	8	8	10	10	13	22	30	39	69	77	90	

Ví dụ về sắp xếp trộn cải tiến: Bước 3

Đổi vai trò của F1 và G1, F2 và G2 cho nhau, ta được hai tập tin F1 và F2 mới:

F1	2	5	9	10	13	31	40	54	65	85	96	98

F2	8	8	10	10	13	22	30	39	69	77	90

Trộn các đường độ dài 12 trong 2 tập tin F1 và F2 được 1 đường ghi vào trong tập tin G1, còn G2 rỗng

G1	2	5	8	8	9	10	10	10	13	13	22	30	31	39	40	54	65	69	77	85	90	96	98

G2

Giải thuật trộn nhiều đường (multiway merge)

- Sử dụng m tập tin (m là một số chẵn > 4) F[1], F[2],... , F[m].
- Gọi h = m/2, giả sử bộ nhớ trong có thể chứa k mẩu tin.
- **Khởi đầu**: Đọc từ tập tin F vào bộ nhớ trong k mẩu tin, sắp xếp trong k mẩu tin này thành một đường rồi ghi luân phiên vào các tập tin F[1], F[2], ... , F[h].
- **Bước 1**: Trộn các đường độ dài k của h tập tin F[1], F[2], ..., F[h] thành một đường độ dài k.h và ghi luân phiên vào trong h tập tin F[h+1], F[h+2], ... , F[m]. Đổi vai trò của F[i] và F[h+i]] cho nhau (với $1 \le i \le h$).
- **Bước 2**: Trộn các đường độ dài kh của h tập tin F[1], F[2], ..., F[h] thành một đường độ dài $k.h^2$ và ghi luân phiên vào trong h tập tin F[h+1], F[h+2], ... , F[m]. Đổi vai trò của F[i] và F[h+i]] cho nhau (với $1 \le i \le h$).
- Sau i bước thì độ dài mỗi đường là $k.h^i$ và giải thuật kết thúc khi $k.h^i \ge n$ và khi đó tập tin đã được sắp chính là một đường ghi trong F[h+1].

Đánh giá giải thuật sắp xếp trộn nhiều đường

- Theo trên thì giải thuật kết thúc sau i bước, với $kh^i \geq n$ hay số bước là: $$i \geq \log_h \frac{n}{k}$$
- Mỗi bước ta phải đọc từ h tập tin và ghi vào trong h tập tin, trung bình mỗi tập tin có n/h mẩu tin. Giả sử mỗi khối lưu được b mẩu tin thì mỗi bước phải truy xuất số khối là: $$\frac{2*h*n}{h*b} = \frac{2n}{b}$$
- Vậy tổng cộng ta chỉ cần số phép truy xuất khối là: $\frac{2n}{b}\log_h \frac{n}{k}$
- So sánh với số phép truy xuất khối của MergeSort, ta thấy rõ ràng $$\frac{2n}{b}\log_h \frac{n}{k} < \frac{2n}{b}\log \frac{n}{k}$$

Ví dụ về sắp xếp trộn nhiều đường

- Sắp xếp tập tin F có 23 mẩu tin với khóa là các số nguyên: 2 31 13 5 98 96 10 40 54 85 65 9 30 39 90 13 10 8 69 77 8 10 22.
- Sử dụng 6 tập tin, giả sử bộ nhớ trong có thể chứa được 3 mẩu tin.

F[1]	2	13	31	9	65	85	8	69	77
F[2]	5	96	98	30	39	90	10	22	
F[3]	10	40	54	8	10	13			

Ví dụ về sắp xếp trộn nhiều đường: Bước 1

Từ 3 tập tin đã có

F[1]	2	13	31	9	65	85	8	69	77
F[2]	5	96	98	30	39	90	10	22	
F[3]	10	40	54	8	10	13			

Trộn các đường độ dài 3 trong các tập tin F[1], F[2], F[3] thành các đường độ dài 9 và ghi vào trong các tập tin F[4], F[5] và F[6].

F[4]	2	5	10	13	31	40	54	96	98
F[5]	8	9	10	13	30	39	65	85	90
F[6]	8	10	22	69	77				

Ví dụ về sắp xếp trộn nhiều đường: Bước 2

Đổi vai trò các tập tin F[1], F[2] và F[3] cho F[4],F[5] và F[6] tương ứng

F[4]	2	5	10	13	31	40	54	96	98
F[5]	8	9	10	13	30	39	65	85	90
F[6]	8	10	22	69	77				

Trộn các đường độ dài 9 trong các tập tin F[1], F[2], F[3] thành đường độ dài 23 và ghi vào trong các tập tin F[4] còn các tập tin F[5] và F[6] đều rỗng.

F[4]	2	5	8	8	9	10	10	10	13	13	22	30	31	39	40	54	65	69	77	85	90	96	98

Lưu trữ thông tin trong tập tin

- Chúng ta sẽ coi một tập tin như là một chuỗi tuần tự các mẩu tin, mỗi mẩu tin bao gồm nhiều trường (field).
- Các mẩu tin có độ dài cố định và khảo sát các thao tác trên tập tin là:
  - Insert: Thêm một mẩu tin vào trong một tập tin.
  - Delete: Xoá một mẩu tin từ trong tập tin.
  - Modify: Sửa đổi thông tin trong các mẩu tin.
  - Retrieve: Tìm lại thông tin được lưu trong tập tin.

Tập tin tuần tự

- **Tổ chức:** một danh sách liên kết của các khối, các mẩu tin được lưu trữ trong các khối theo một thứ tự bất kỳ.
- **Tìm mẩu tin:**
  - Đọc từng khối, với mỗi khối ta tìm mẩu tin cần tìm trong khối, nếu không tìm thấy ta lại đọc tiếp một khối khác.
  - Quá trình cứ tiếp tục cho đến khi tìm thấy mẩu tin hoặc duyệt qua toàn bộ các khối của tập tin và trong trường hợp đó thì mẩu tin không tồn tại trong tập tin.

Tập tin tuần tự (tt)

- **Thêm mẩu tin mới:**
  - Đưa mẩu tin này vào khối cuối cùng của tập tin nếu như khối đó còn chỗ trống.
  - Ngược lại nếu khối cuối cùng đã hết chỗ thì xin cấp thêm một khối mới, thêm mẩu tin vào khối mới và nối khối mới vào cuối danh sách.
- **Sửa đổi mẩu tin:** Tìm mẩu tin cần sửa, thực hiện các sửa đổi cần thiết sau đó ghi lại mẩu tin vào vị trí cũ trong tập tin.

Tập tin tuần tự (tt)

- **Xoá mẩu tin:**
  - Tìm mẩu tin đó, nếu tìm thấy ta có thể thực hiện một trong các cách xoá sau đây:
  - Một là xoá mẩu tin cần xoá trong khối lưu trữ nó, nếu sau khi xoá, khối trở nên rỗng thì xoá khối khỏi danh sách.
  - Hai là đánh dấu xoá mẩu tin bằng một trong hai cách:
    - Thay thế mẩu tin bằng một giá trị nào đó mà giá trị này không bao giờ là giá trị thật của bất kỳ một mẩu tin nào.
    - Sử dụng bit xóa

Đánh giá tập tin tuần tự

- Đây là một phương pháp tổ chức tập tin đơn giản nhất nhưng kém hiệu quả nhất.
- Giả sử tập tin có n mẩu tin và mỗi khối lưu trữ được k mẩu tin thì toàn bộ tập tin được lưu trữ trong n/k khối.
- Do đó mỗi lần tìm (hoặc thêm hoặc sửa hoặc xoá) một mẩu tin thì phải truy xuất n/k khối.

Tăng tốc độ cho các thao tác tập tin

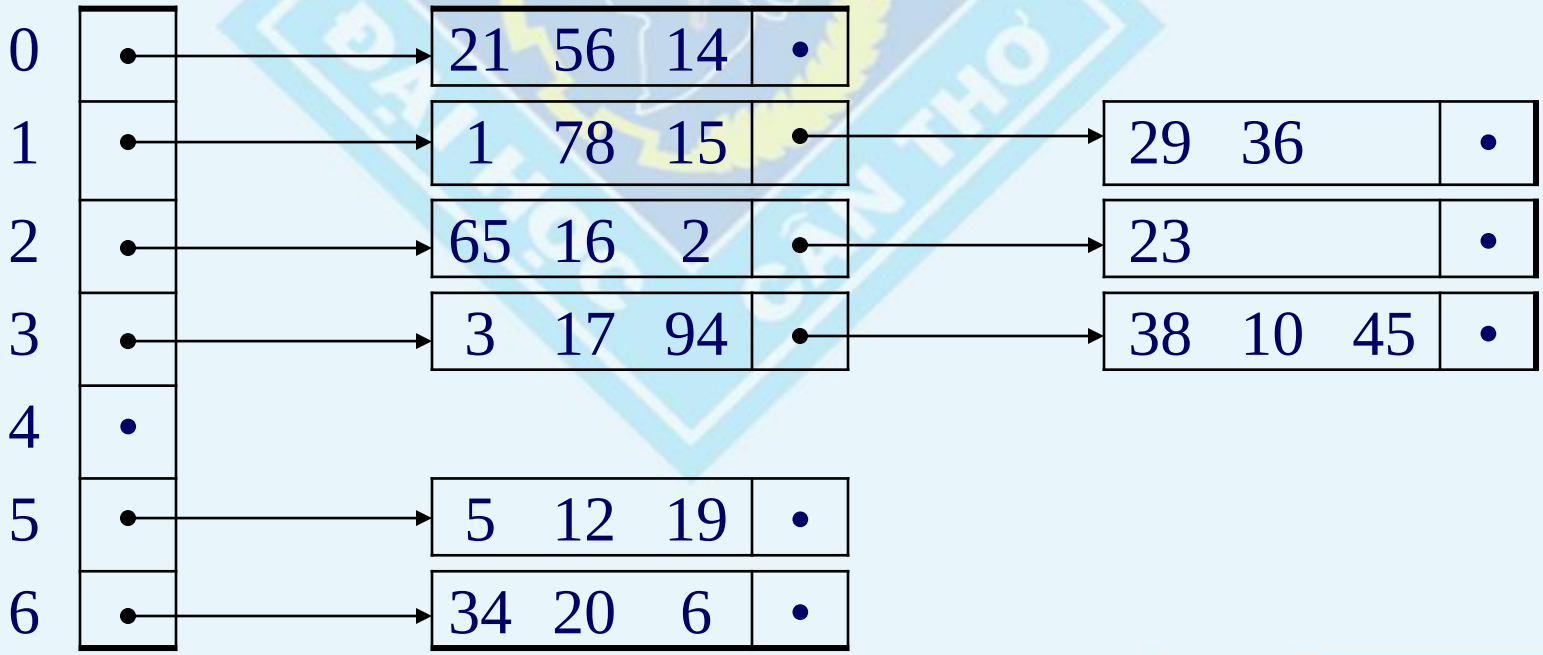
- Để cải thiện tốc độ thao tác trên tập tin, chúng ta phải tìm cách giảm số lần truy xuất khối.
- Muốn vậy phải tìm các cấu trúc sao cho khi tìm một mẩu tin chỉ cần truy xuất một số nhỏ các khối.
- Để tạo ra các tổ chức tập tin như vậy chúng ta phải giả sử rằng mỗi mẩu tin có một khoá (key).
- Hai mẩu tin khác nhau thì khoá của chúng phải khác nhau.

Tập tin băm (hash files): Tổ chức

- Bảng băm là mảng một chiều B gồm m phần tử B[0], B[1], ..., B[m-1]).
- Mỗi phần tử là một con trỏ, trỏ tới phần tử đầu tiên của danh sách liên kết các khối.
- Để phân phối mẩu tin có khóa x vào trong các danh sách liên kết, ta dùng hàm băm.
- h(x) = x MOD m.

Ví dụ về tập tin bảng băm

- Một tập tin có 24 mẩu tin với giá trị khóa là các số nguyên: 3, 5, 12, 65, 34, 20, 21, 17, 56, 1, 16, 2, 78, ,94, 38 ,15 ,23, 14, 10, 29, 19, 6, 45, 36
- Giả sử chúng ta có thể tổ chức tập tin này vào trong bảng băm gồm 7 phần tử và giả sử mỗi khối có thể chứa được tối đa 3 mẩu tin. Với mỗi mẩu tin r có khóa là x ta xác định h(x) = x MOD 7 và đưa mẩu tin r vào trong một khối của danh sách liên kết được trỏ bởi B[h(x)].



Tập tin băm: Tìm mẩu tin

- Để tìm một mẩu tin r có khóa là x, chúng ta xác định h(x) chẳng hạn h(x) = i, khi đó ta chỉ cần tìm r trong danh sách liên kết được trỏ bởi B[i].
- Chẳng hạn để tìm mẩu tin r có khóa là 36, ta tính h(36) = 36 MOD 7 = 1.
- Như vậy nếu mẩu tin r tồn tại trong tập tin thì nó phải thuộc một khối nào đó được trỏ bởi B[1].

Tập tin băm: Thêm mẩu tin mới

- Để thêm mẩu tin r có khoá x, trước hết ta phải tìm mẩu tin r.
- Nếu tìm thấy thì thông báo "mẩu tin đã tồn tại"
- Ngược lại ta sẽ tìm một khối (trong danh sách các khối của lô được trỏ bởi B[h(x)]) còn chỗ trống và thêm r vào khối này.
- Nếu không còn khối nào đủ chỗ cho mẩu tin mới ta yêu cầu hệ thống cấp phát một khối mới và đặt mẩu tin r vào khối này rồi nối khối mới này vào cuối danh sách liên kết của lô.

Tập tin băm: Xoá mẩu tin

- Để xoá mẩu tin r có khoá x, trước hết ta phải tìm mẩu tin này.
- Nếu không tìm thấy thì thông báo "Mẩu tin không tồn tại".
- Nếu tìm thấy thì đặt bít xoá cho nó.
- Ta cũng có thể xoá hẳn mẩu tin r và nếu việc xoá này làm khối trở nên rỗng thì ta giải phóng khối này (xoá khối khỏi danh sách liên kết các khối).

Tập tin băm: Đánh giá

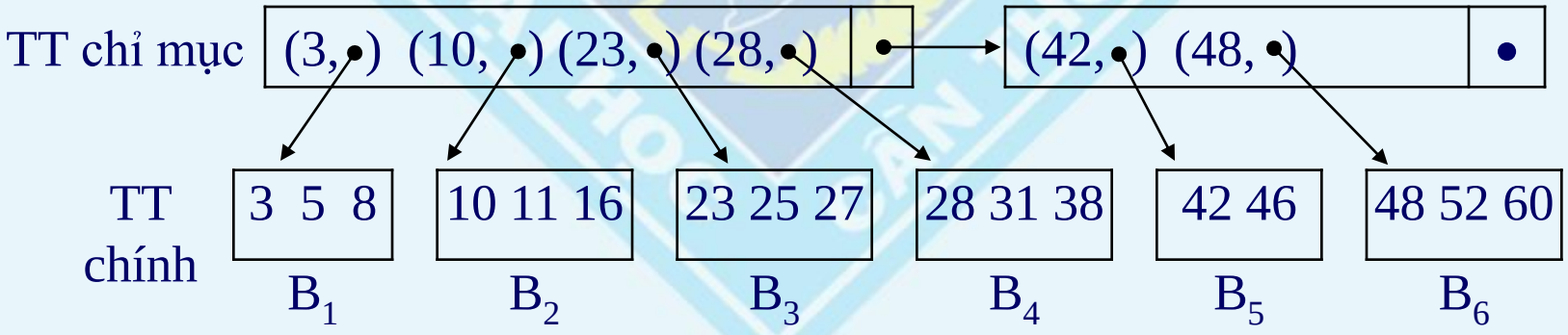
- Giả sử tập tin có n mẩu tin và mỗi khối lưu trữ được k mẩu tin thì tập tin cần n/k khối.
- Trung bình mỗi danh sách liên kết (mỗi lô) của bảng băm có n/mk khối (do bảng băm có m lô), mà chúng ta chỉ tìm trong một danh sách liên kết nên ta chỉ phải truy xuất n/mk khối.
- Số này nhỏ hơn m lần so với cách tổ chức tập tin tuần tự (trong tập tin tuần tự ta cần truy xuất tất cả các khối, tức là n/k khối).

Tập tin chỉ mục (index file): Tổ chức

- Tập tin chính và tập tin chỉ mục thưa (sparse index).
- Tập tin chính bao gồm các khối lưu các mẩu tin sắp thứ tự theo giá trị khóa.
- Tập tin chỉ mục thưa bao gồm các khối chứa các cặp (x,p) trong đó x là khoá của mẩu tin đầu tiên trong một khối của tập tin chính, còn p là con trỏ, trỏ đến khối đó.

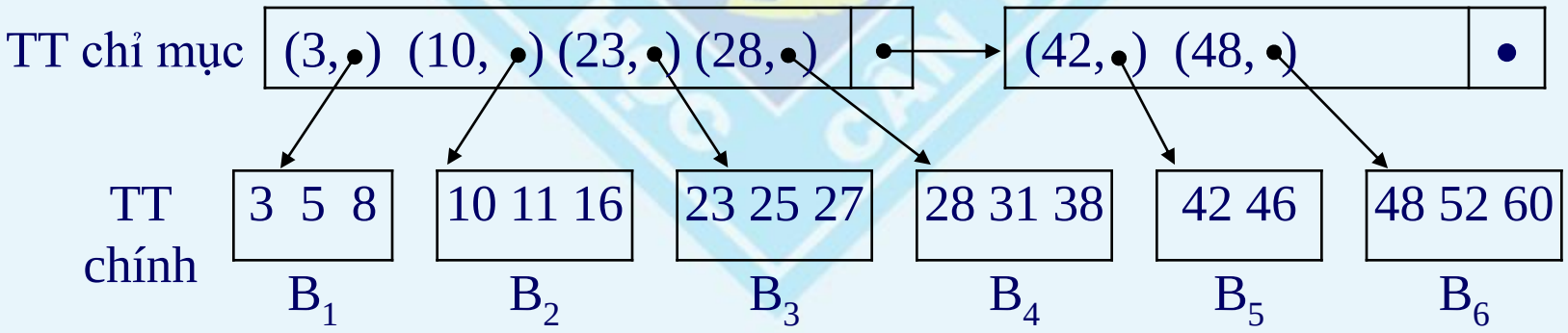
Tập tin chỉ mục: Ví dụ

Các khối trong tập tin chính lưu trữ được tối đa 3 mẩu tin, mỗi khối trong tập tin chỉ mục lưu trữ được tối đa 4 cặp khoá – con trỏ.



Tập tin chỉ mục: Tìm mẩu tin

- Để tìm mẩu tin r có khoá x, ta phải tìm cặp (z,p) với z là giá trị lớn nhất và z ≤ x. Mẩu tin r có khoá x nếu tồn tại thì sẽ nằm trong khối được trỏ bởi p.
- Chẳng hạn để tìm mẩu tin r có khoá 46 trong tập tin được biểu diễn trong hình sau:

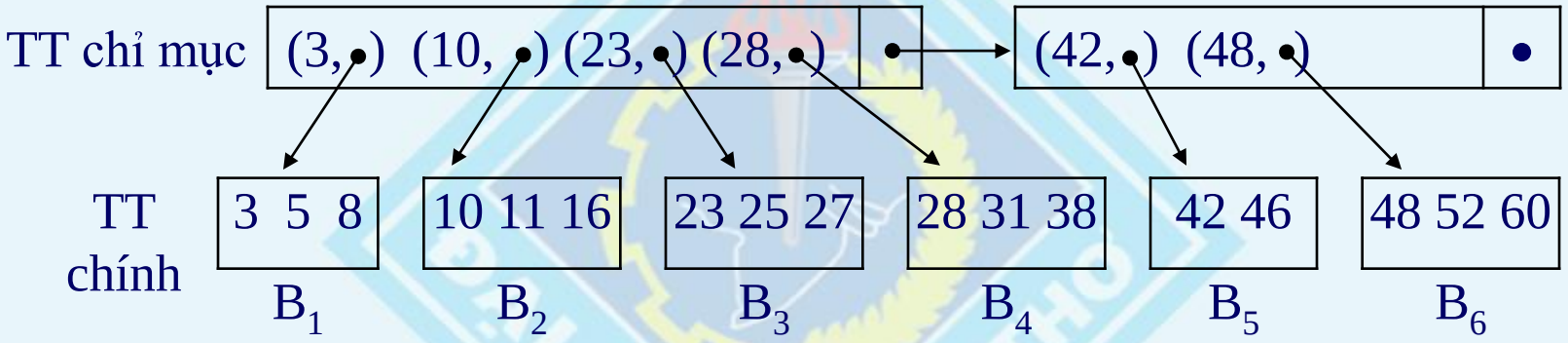


Tập tin chỉ mục: Thêm mẩu tin mới

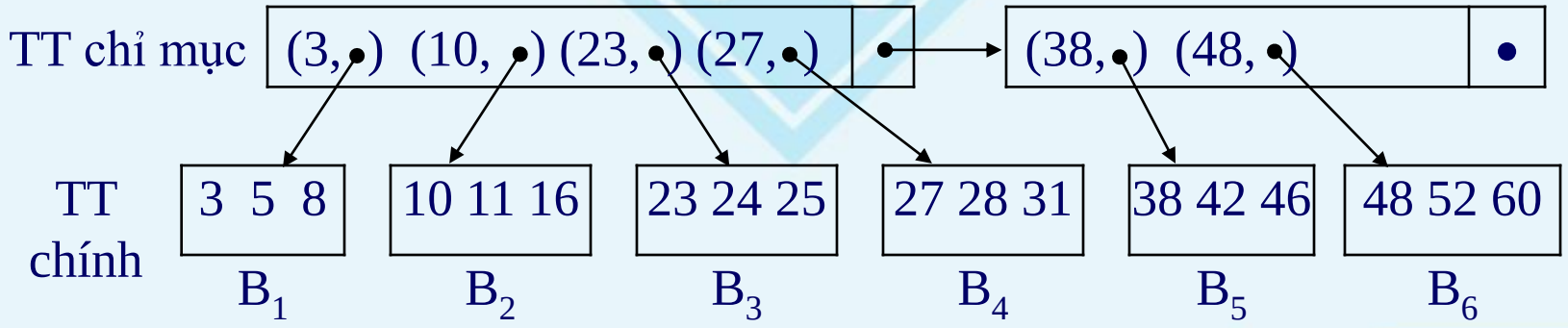
- Giả sử tập tin chính được lưu trong các khối B1, B2, ..., Bm.
- Dùng thủ tục tìm kiếm để xác định một khối Bi nào đó.
- Nếu tìm thấy thì thông báo "mẩu tin đã tồn tại", ngược lại, Bi là nơi có thể chứa mẩu tin r.
- Nếu Bi còn chỗ trống thì xen r vào đúng vị trí của nó trong Bi.
- Chỉnh lại tập tin chỉ mục nếu mẩu tin mới trở thành mẩu tin đầu tiên trong khối Bi.
- Nếu Bi không còn chỗ trống để xen thì ta phải xét khối Bi+1 để có thể chuyển mẩu tin cuối cùng trong khối Bi thành mẩu tin đầu tiên của khối Bi+1 và xen mẩu tin r vào đúng vị trí của nó trong khối Bi .
- Điều chỉnh lại tập tin chỉ mục cho phù hợp với trạng thái mới của Bi+1.
- Quá trình này có thể dẫn đến việc ta phải xét khối Bm, nếu Bm đã hết chỗ thì yêu cầu hệ thống cấp thêm một khối mới Bm+1, chuyển mẩu tin cuối cùng của Bm sang Bm+1, mẩu tin cuối cùng của Bm-1 sang Bm… Xen mẩu tin r vào khối Bi và cập nhật lại tập tin chỉ mục.
- Việc cấp phát thêm khối mới Bm+1 đòi hỏi phải xen thêm một cặp khoá-con trỏ vào khối cuối cùng của tập tin chỉ mục, nếu khối này hết chỗ thì xin cấp thêm một khối mới để xen cặp khóa-con trỏ này.

Ví dụ thêm mẩu tin mới

Xen mẩu tin r với khóa x = 24 vào trong tập tin được biểu diễn trong hình sau:



Cấu trúc của tập tin sau khi thêm mẩu tin r như sau:



Tập tin chỉ mục: Xoá mẩu tin

- Tìm r, nếu không tìm thấy thì thông báo "Mẩu tin không tồn tại".
- Ngược lại ta xoá mẩu tin r trong khối chứa nó.
- Nếu mẩu tin bị xoá là mẩu tin đầu tiên của khối thì phải cập nhật lại giá trị khoá trong tập tin chỉ mục.
- Trong trường hợp khối trở nên rỗng sau khi xoá mẩu tin thì giải phóng khối đó và xoá cặp (khoá, con trỏ) của khối trong tập tin chỉ mục.
- Việc xoá trong tập tin chỉ mục cũng có thể dẫn đến việc giải phóng khối trong tập tin này.

Tập tin chỉ mục: Đánh giá

- Ta thấy việc tìm một mẩu tin chỉ đòi hỏi phải đọc chỉ một số nhỏ các khối (một khối trong tập tin chính và một số khối trong tập tin chỉ mục).
- Tuy nhiên trong việc xen thêm mẩu tin, như trên đã nói, có thể phải đọc và ghi tất cả các khối trong tập tin chính.
- Đây chính là nhược điểm của tập tin chỉ mục.

Cây tìm kiếm m-phân

- Cây tìm kiếm m-phân (m-ary tree) là sự tổng quát hoá của cây tìm kiếm nhị phân trong đó mỗi nút có thể có m nút con.
- Giả sử n1 và n2 là hai con của một nút nào đó, n1 bên trái n2 thì tất cả các con của n1 có giá trị nhỏ hơn giá trị của các nút con của n2.

B-cây

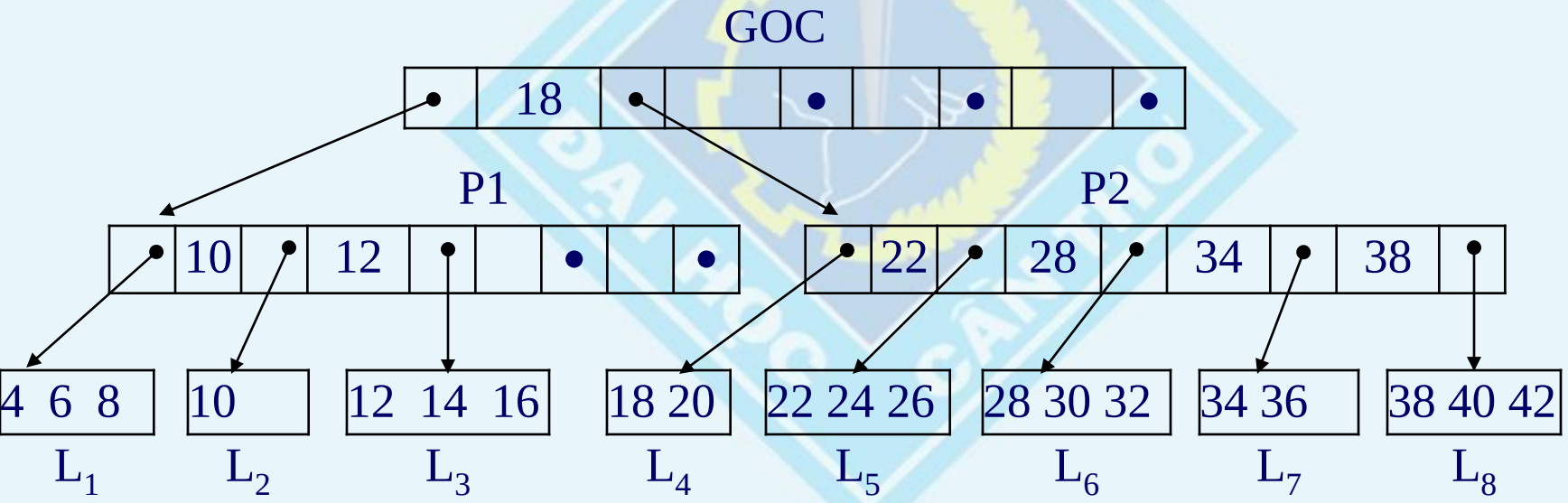
- B-cây bậc m là cây tìm kiếm m-phân cân bằng có các tính chất sau:
  - Nút gốc hoặc là lá hoặc có ít nhất hai nút con,
  - Mỗi nút, trừ nút gốc và nút lá, có từ $\lceil m/2 \rceil$ đến m nút con và
  - Các đường đi từ gốc tới lá có cùng độ dài.

Tập tin B-cây:Tổ chức

- Mỗi nút trên cây là một khối trên đĩa, các mẩu tin của tập tin được lưu trữ trong các nút lá trên B-cây và lưu theo thứ tự của khoá.
- Giả sử mỗi nút lá lưu trữ được nhiều nhất b mẩu tin.
- Mỗi nút không phải là nút lá có dạng (p0,k1,p1,k2,p2,...,kn,pn), với pi ($0 \leq i \leq n$) là con trỏ, trỏ tới nút con thứ i của nút đó và ki là các giá trị khóa. Các khoá trong một nút được sắp thứ tự, tức là $k1 < k2 < ... < kn$.
- Tất cả các khoá trong cây con được trỏ bởi p_0 đều nhỏ hơn k_1.
- Tất cả các khoá nằm trong cây con được trỏ bởi p_i ($0 < i < n$) đều lớn hơn hoặc bằng k_i và nhỏ hơn k_{i+1}.
- Tất cả các khoá nằm trong cây con được trỏ bởi p_n đều lớn hơn hoặc bằng k_n.

Tập tin B-cây: Ví dụ

Cho tập tin bao gồm 20 mẩu tin với giá trị khóa là các số nguyên được tổ chức thành B-cây bậc 5 với các nút lá chứa được nhiều nhất 3 mẩu tin.



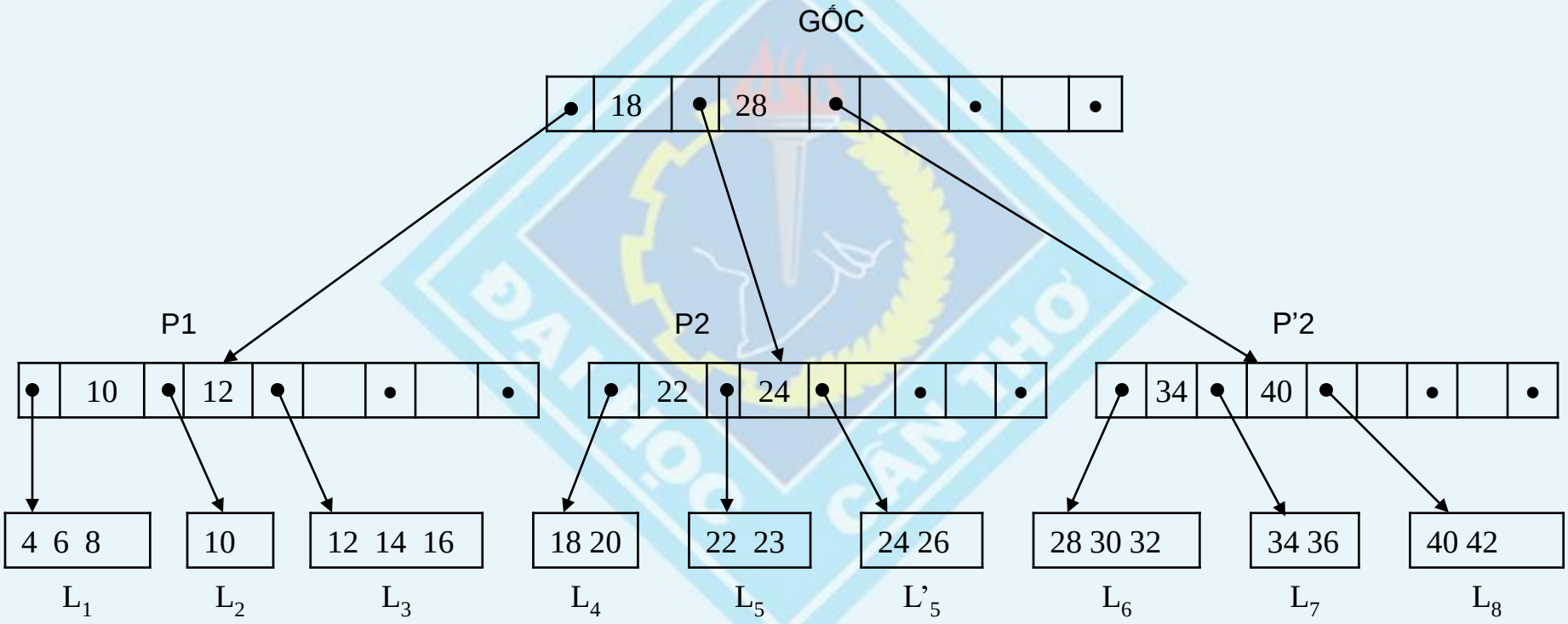
Tập tin B-cây: Tìm mẩu tin

- Băt đầu gốc đến nút lá chứa r (nếu r tồn tại trong tập tin).
- Tại mỗi bước,đưa nút trong (p0, k1, p1, ..., kn, pn) vào bộ nhớ trong và xác định mối quan hệ giữa x với các giá trị khóa ki.
  - Nếu $k_i \leq x < k_{i+1}$ $(0 < i < n)$ chúng ta sẽ xét tiếp nút được trỏ bởi pi.
  - Nếu $x < k1$ ta sẽ xét tiếp nút được trỏ bởi p0.
  - Nếu $x \geq kn$ ta sẽ xét tiếp nút được trỏ bởi pn.
- Quá trình trên sẽ dẫn đến việc xét một nút lá.
- Trong nút lá này ta sẽ tìm mẩu tin r với khóa x bằng tìm kiếm tuần tự hoặc tìm kiếm nhị phân.

Tập tin B-cây: Thêm mẩu tin mới

- Tìm r. Việc tìm kiếm này sẽ dẫn đến nút lá L.
- Nếu tìm thấy thì thông báo "Mẩu tin đã tồn tại", ngược lại thì L là nút lá mà ta có thể xen r vào trong đó.
- Nếu khối L này còn chỗ cho r thì ta thêm r vào sao cho đúng thứ tự của nó trong khối L và giải thuật kết thúc.
- Nếu L không còn chỗ thì cấp phát một khối mới L'.
- Dời $\lceil b/2 \rceil$ mẩu tin nằm ở cuối khối L sang L' rồi xen r vào L hoặc L' sao cho việc xen đảm bảo thứ tự các khoá trong khối.
- Giả sử nút P là cha của L.
- Xen đệ quy để xen vào P một khóa k' và con trỏ p' tương ứng của L'.
- Trong trường hợp trước khi xen k' và p', P đã có đủ m con thì ta phải cấp thêm một khối mới P' và chuyển một số con của P sang P' và xen con mới vào P hoặc P' sao cho cả P và P' đều có ít nhất $\lceil m/2 \rceil$ con.
- Việc chia cắt P => phải xen một khóa và một con trỏ vào nút cha của P...
- Quá trình này có thể sẽ dẫn tới nút gốc và cũng có thể phải chia cắt nút gốc, trong trường hợp này phải tạo ra một nút gốc mới mà hai con của nó là hai nửa của nút gốc cũ.
- Khi đó chiều cao của B-cây sẽ tăng lên 1.

Kết quả xen 23



Thank you