# CHƯƠNG 4:
# CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT LƯU TRỮ NGOÀI

## 4.1 TỔNG QUAN

### 4.1.1 Mục tiêu

Sau khi học chương này, sinh viên cần nắm được các vấn đề sau:

- Tiêu chuẩn để đánh giá giải thuật xử lý ngoài.
- Giải thuật sắp xếp trộn để sắp xếp ngoài và phương pháp cải tiến tốc độ sắp xếp trộn.
- 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 và đặc biệt là tập tin B-cây.

### 4.1.2 Kiến thức cơ bản cần thiết

- Cấu trúc dữ liệu danh sách liên kết.
- Các cấu trúc dữ liệu cây và bảng băm.
- Vấn đề tìm kiếm tuần tự và tìm kiếm nhị phân.
- Các thao tác trên kiểu dữ liệu tập tin.

### 4.1.3 Tài liệu tham khảo

**A.V. Aho, J.E. Hopcroft, J.D. Ullman**; *Data Structures and Algorithms*; Addison-Wesley; 1983. (Chapter 10).

**Đinh Mạnh Tường**; *Cấu trúc dữ liệu & Thuật toán*; Nhà xuất bản khoa học và kĩ thuật; Hà nội-2001. (Chương 7).

### 4.1.4 Nội dung cốt lõi

Trong chương này chúng ta sẽ nghiên cứu hai vấn đề chính là sắp xếp dữ liệu được lưu trong bộ nhớ ngoài và kĩ thuật lưu trữ tập tin. Trong kĩ thuật lưu trữ tập tin chúng ta sẽ sử dụng các cấu trúc dữ liệu tuần tự, bảng băm, tập tin chỉ mục và cấu trúc B-cây.

## 4.2 MÔ HÌNH XỬ LÝ 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.

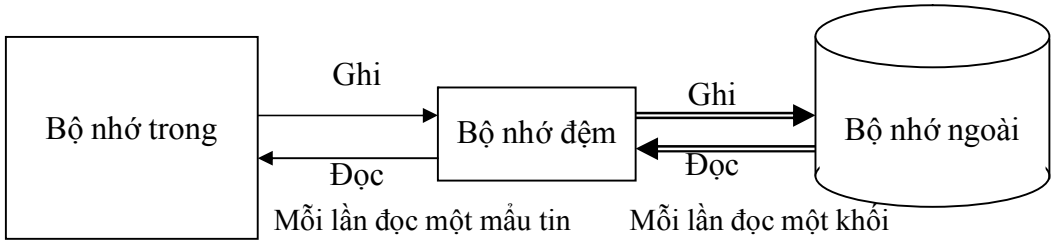
Kiểu dữ liệu tập tin là kiểu thích hợp nhất cho việc biểu diễn dữ liệu được lưu trong bộ nhớ ngoài. Hệ điều hành chia bộ nhớ ngoài thành các khối (block) có kích thước bằng nhau, kích thước này thay đổi tùy thuộc vào hệ điều hành nhưng nói chung là từ 512 bytes đến 4096 bytes.

Trong quá trình xử lý, việc chuyển giao dữ liệu giữa bộ nhớ trong và bộ nhớ ngoài được tiến hành thông qua vùng nhớ đệm (buffer). Bộ đệm là một vùng dành riêng của bộ nhớ trong mà kích thước bằng với kích thước của một khối của bộ nhớ ngoài.

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, không có mẩu tin nào bị chia cắt để lưu trên hai khối khác nhau.

Trong thao tác đọc, nguyên một khối của tập tin được chuyển vào trong bộ đệm và lần lượt đọc các mẩu tin có trong bộ đệm cho tới khi bộ đệm rỗng thì lại chuyển một khối từ bộ nhớ ngoài vào bộ đệm.

Để ghi thông tin ra bộ nhớ ngoài, các mẩu tin lần lượt được xếp vào trong bộ đệm cho đến khi đầy bộ đệm thì nguyên một khối được chuyển ra bộ nhớ ngoài. Khi đó bộ đệm trở nên rỗng và lại có thể xếp tiếp các mẩu tin vào trong đó.




*Hình 4-1: Mô hình giao tiếp giữa bộ nhớ trong, bộ nhớ ngoài và vùng nhớ đệm*

Như vậy đơn vị giao tiếp giữa bộ nhớ trong và bộ đệm là mẩu tin còn giữa bộ đệm và bộ nhớ ngoài là khối.

Hình 4-1 mô tả hoạt động của bộ nhớ trong, bộ đệm và bộ nhớ ngoài trong thao tác đọc và ghi tập tin

## 4.3 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. Ví dụ giả sử ta có một khối có thể lưu 1000 số nguyên được lưu trên đĩa quay với vận tốc 1000 vòng/ phút thì thời gian để đưa đầu từ vào rãnh chứa khối và quay đĩa để đưa khối đến chỗ đầu từ hết khoảng 100 mili giây. Với thời gian này máy có thể thực hiện 100000 lệnh, tức là đủ để sắp xếp các số nguyên này theo giải thuật QuickSort. Vì vậy khi đánh giá các

giải thuật thao tác trên bộ nhớ ngoài, 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). Vì kích thước các khối là cố định nên ta không thể tìm cách tăng kích thước một khối mà chúng ta phải tìm cách giảm số lần truy xuất khối.

## 4.4 SẮP XẾP NGOÀI

Sắp xếp dữ liệu được tổ chức như một tập tin hoặc tổng quát hơn, sắp xếp dữ liệu được lưu trên bộ nhớ ngoài gọi là sắp xếp ngoài.

### 4.4.1 Sắp xếp trộn (merge sorting)

#### 4.4.1.1 Khái niệm về đường

Đường độ dài k là một tập hợp k mẩu tin đã đươc sắp thứ tự theo khoá tức là, nếu các mẩu tin $r_1,r_2,...,r_k$ có khoá lần lượt là $k_1,k_2,...,k_k$ tạo thành một đường thì $k_1 \leq k_2 \leq ... \leq k_k$.

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 4-1:** 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 |

#### 4.4.1.2 Giải thuật

Để sắp xếp tập tin F có n mẩu tin ta sử dụng 4 tập tin F1, F2, G1 và G2.

Khởi đầu ta 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 được 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 hai đường này 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 hai đường này 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 2I. Nếu 2I ( n thì giải thuật kết thúc, lúc đó tập tin G2 sẽ rỗng và tập tin G1 chứa các mẩu tin đã được sắp.

#### 4.4.1.3 Đánh giá giải thuật sắp xếp trộn

Ta thấy giải thuật kết thúc 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 bước cần đọc và ghi $\frac{2*2*n}{2*b} = \frac{2n}{b}$ khối mà chúng ta cần logn bước vậy tổng cộng chúng ta cần $\frac{2n}{b}\log n$ phép truy xuất khối.

**Ví dụ 4-2:** 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

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

**Bước 1:** 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 | F1 |
| G2 | 5 13 | 10 40 | 9 65 | 13 90 | 69 77 | 22 | F2 |

**Bước 2:** Đổi vai trò của F1 và G1, F2 và G2 cho nhau. Trộn các đường độ dài 2 trong hai tập tin F1 và F2 được các đường độ dài 4 rồi ghi luân phiên vào trong hai tập tin G1 và G2:

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G1 | 2 5 | 13 31 | 9 54 | 65 85 | 8 10 | 69 77 | F1 |
| G2 | 10 40 | 96 98 | 13 30 | 39 90 | 8 10 | 22 | F2 |

**Bước 3:** Đổi vai trò của F1 và G1, F2 và G2 cho nhau. Trộn các đường độ dài 4 trong hai tập tin F1 và F2 được các đường độ dài 8 rồi ghi luân phiên vào trong hai tập tin G1 và G2:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G1 | 2 | 5 | 10 | 13 | 31 | 40 | 96 | 98 | 8 | 8 | 10 | 10 | 22 | 69 | 77 | F1 |
| G2 | 9 | 13 | 30 | 39 | 54 | 65 | 85 | 90 | | | | | | F2 | | |

**Bước 4**: Đổi vai trò của F1 và G1, F2 và G2 cho nhau. Trộn các đường độ dài 8 trong hai tập tin F1 và F2 được các đường độ dài 16 rồi ghi luân phiên vào trong 2 tập tin G1 và G2.

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| G1 | 2 | 5 | 9 | 10 | 13 | 13 | 30 | 31 | 39 | 40 | 54 | 65 | 85 | 90 | 96 | 98 | F1 |
| G2 | 8 | 8 | 10 | 10 | 22 | 69 | 77 | | | | | | F2 | | | | |

**Bước 5:** Đổi vai trò của F1 và G1, F2 và G2 cho nhau. Trộn các đường độ dài 16 trong hai tập tin F1 và F2 được 1 đường độ dài 23 rồi 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Tập tin G1 chứa các mẩu tin đã được sắp còn tập tin G2 rỗng.

#### 4.4.1.4 Chương trình

```
procedure Merge(k:integer; f1,f2,g1,g2: File of RecordType);
{Thủ tục này trộn các đường độ dài k và trong hai tập tin f1
và f2 thành các đường đọ dài 2k và ghi luân phiên vào trong
hai tập tin g1 và g2}

var
OutSwithh : boolean; {Nếu OutSwitch = TRUE thì  ghi vào tập
tin g1, ngược lại ghi vào  g2}

Winner: integer; {Để chỉ định mẩu tin hiện hành nào trong hai
tập tin f1 và f2 sẽ được ghi ra tập tin g1 hoặc g2}

Used: array[1..2] of integer; { Used[ij] ghi  số mẩu tin đã
được đọc trong đường hiện tại của tập tin fj }

Fin : array[1..2] Of boolean; {Fin[j] sẽ có giá trị TRUE nếu
đã đọc hết các  mẩu tin trong đường hiện hành của fj hoạc đx
đến cuối tập tin fj }

Current: array[1..2] Of RecordType; { Current[j] để lưu mẩu
tin hiện hành của tập tin f[j]}

procedure GetRecord(i:integer);
{Nếu đã đọc hết các mẩu tin trong đường hiện hành của tập tin
fi hoặc đã đến cuối tập tin fi thì đặt fin[i] = TRUE nếu
không thì đọc một mẩu tin của tập tin fi vào trong
current[i]}

begin
     Used[i] := Used[i] + 1;
     if (Used[i] = k+1 ) or (i = 1) and ( eof(f1)) or (i = 2
     and ( eof(f2)) then fin[i] := TRUE
     else if i=1 then Read(f1, current[1])
          else read(f2, current[2]);
end;
begin
     { Khởi tạo }
     OutSwitch := TRUE;

ReSet(f1);

ReSet(f2);
```

```
ReWrite(g1);
ReWrite(g2);
while (not eof(f1)) or (not eof(f2)) do begin
          {Bắt đầu đọc các mẩu tin từ trong hai đường hiện
          hành của hai tập tin f1,f2 }
          Used[1] := 0; Used[2] := 0;
          Fin[1] := FALSE ; Fin[2] := FALSE ;
          GetRecord(1) ; GetRecord(2);
          while ( not fin[1] ) or (not fin[2]) do begin
               {Trộn hai đường }
               { Chọn Winner }
               if Fin[1] then Winner := 2
               else if Fin[2] then Winner := 1
                    else  if  current[1].key  <  Current[2].key
                    then
                         Winner := 1
                    else   Winner := 2;
               if OutSwitch then Write(g1, Current[winner] )
               else Write(g2, current[winner] );
               GetRecord(Winner);
          end;
          OutSwitch := Not OutSwitch;
end;
end;
```

### 4.4.2 Cải tiến sắp xếp trộn

Ta thấy quá trình sắp xếp trộn nói trên bắt đầu từ các đường độ dài 1 cho nên phải sau logn bước giải thuật mới kết thúc. Chúng ta có thể tiết kiệm thời gian bằng cách chọn một số k thích hợp sao cho k mẩu tin có thể đủ chứa trong bộ nhớ trong. Mỗi lần đọc vào bộ nhớ trong k mẩu tin, dùng sắp xếp trong (chẳng hạn dùng QuickSort) 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 sẽ kết thúc khi $k2^i \geq n$ hay $i \geq \log\frac{n}{k}$. Do đó số phép truy xuất khối sẽ là $\frac{2n}{b}\log\frac{n}{k}$. Dễ thấy $\frac{2n}{b}\log\frac{n}{k} < \frac{2n}{b}\log n$ tức là ta tăng được tốc độ sắp xếp trộn.

**Ví dụ 4-3**: Lấy tập tin F có 23 mẩu tin với khóa là các số nguyên như trong ví dụ 4-2:

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, ta đọc lần lượt 3 mẩu tin của F vào bộ nhớ trong , dùng một sắp xếp trong để sắp xếp chúng và ghi phiên vào 2 tập tin F1 và F2.

| | | | |
|---|---|---|---|
| 2 13 31 | 10 40 54 | 30 39 90 | 8 69 77 |

F1

| 5 96 98 | 9 65 85 | 8 10 13 | 10 22 |
|---|---|---|---|

F2

**Bước 1:** 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 |
|---|---|---|

F1

| G2 | 9 10 40 54 65 85 | 8 10 22 69 77 |
|---|---|---|

F2

**Bước 2**: Đổi vai trò của F1 và G1, F2 và G2 cho nhau. Trộn các đường độ dài 6 trong 2 tập tin F1 và F2 được các đường độ dài 12 rồi ghi luân phiên vào trong 2 tập tin G1 và G2:

| G1 | 2 5 9 10 13 31 40 54 65 85 96 98 |
|---|---|

F1

| G2 | 8 8 10 10 13 22 30 39 69 77 90 |
|---|---|

F2

**Bước 3**: Đổi vai trò của F1 và G1, F2 và G2 cho nhau. 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 77 85 90 96 98 |
|---|---|

Tập tin G1 chứa các mẩu tin đã được sắp còn tập tin G2 rỗng.

### 4.4.3 Trộn nhiều đường (multiway merge)

#### 4.4.3.1 Giải thuật

Để sắp xếp tập tin F có n mẩu tin ta sử dụng m tập tin (m là một số chẵn) F[1], F[2],... , F[m]. Trong trường hợp m=4 ta có giải thuật sắp xếp trộn bình thường.

Gọi h = m/2, ta có nội dung của phương pháp như sau (ta vẫn giả sử bộ nhớ trong có thể chứa k mẩu tin).

**Khởi đầu**: Mỗi lần đọc từ tập tin F vào bộ nhớ trong k mẩu tin, sử dụng một sắp xếp trong để sắp xếp 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].

#### 4.4.3.2 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 $i \geq \log_h \frac{n}{k}$. Mỗi bước ta phải đọc từ h tập tin và ghi vào trong h tập tin, trung bình mỗi tập tin có $\frac{n}{h}$ mẩu tin. Ta vẫn giả sử mỗi khối lưu được b mẩu tin thì mỗi bước phải truy xuất $\frac{2*h*n}{h*b} = \frac{2n}{b}$ khối. Do chúng ta cần $\log_h \frac{n}{k}$ bước nên tổng cộng ta chỉ cần $\frac{2n}{b}\log_h^{\frac{n}{k}}$ phép truy xuất khối.Ta thấy rõ ràng $\frac{2n}{b}\log_h^{\frac{n}{k}} < \frac{2n}{b}\log\frac{n}{k}$ và thủ tục mergeSort nói trên là một trường hợp đặc biệt khi h = 2.

**Ví dụ 4-4**: Lấy tập tin F có 23 mẩu tin với khóa là các số nguyên như trong ví dụ 4-2

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 để sắp xếp tập tin F. Ta giả sử bộ nhớ trong có thể chứa được 3 mẩu tin, ta đọc lần lượt 3 mẩu tin của F vào bộ nhớ trong , dùng một sắp xếp trong để sắp xếp chúng và ghi phiên vào 3 tập tin F[1], F[2] và F[3] như sau:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 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 | | | |

**Bước 1**: Trộn các đường độü 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[1] |
| F[5] | 8 | 9 | 10 | 13 | 30 | 39 | 65 | 85 | 90 | F[2] |
| F[6] | 8 | 10 | 22 | 69 | 77 | | | | | F[3] |

**Bước 2**: Đổi vai trò của F[1] cho F[4], F[2] cho F[5] và F[3] cho F[6]. Trộn các đường độ dài 9 trong các tập tin F[1], F[2], F[3] thành 1 đường độ dài 23 và ghi vào trong tập tin F[4].

| | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

Tập tin F[4] chứa các mẩu tin đã được sắp còn F[5] và F[6] rỗng.

## 4.5 LƯU TRỮ THÔNG TIN TRONG TẬP TIN

Trong phần này ta sẽ nghiên cứu các cấu trúc dữ liệu và giải thuật cho lưu trữ (storing) và lấy thông tin (retrieving) trong các tập tin được lưu trữ ngoài. Chúng ta sẽ coi một tập tin như là một chuỗi tuần tự các mẩu tin, mỗi mẩu tin bao gồm nhiều trường (field). Một trường có thể có độ dài cố định hoặc độ dài thay đổi. Ở đây ta sẽ xét 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 của tập tin, và
- Retrieve: Tìm lại thông tin được lưu trong tập tin.

Sau đây ta sẽ nghiên cứu một số cấu trúc dữ liệu dùng để lưu trữ tập tin. Với mỗi cấu trúc chúng ta sẽ trình bày tổ chức, cách thức tiến hành các thao tác tìm, thêm, xoá mẩu tin và có đánh giá về cách tổ chức đó. Sự đánh giá ở đây chủ yếu là đánh giá xem để tìm một mẩu tin thì phải đọc bao nhiêu khối vì các thao tác khác đều phải sử dụng thao tác tìm.

### 4.5.1 Tập tin tuần tự

#### 4.5.1.1 Tổ chức

Tập tin tuần tự là một danh sách liên kết của các khối, các mẩu tin được lưu trữ trong các khối theo một thứ tự bất kỳ.

#### 4.5.1.2 Tìm mẩu tin

Việc tìm kiếm một mẩu tin có giá trị xác định được thực hiện bằng cách 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.

#### 4.5.1.3 Thêm mẩu tin mới

Việc thêm một mẩu tin có thể thực hiện đơn giản bằng cách 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.

#### 4.5.1.4 Sửa đổi mẩu tin

Để sửa đổi một mẩu tin có giá trị cho trước, ta tìm mẩu tin cần sửa đổi rồi thực hiện các sửa đổi cần thiết sau đó ghi lại mẩu tin vào vị trí cũ trong tập tin.

#### 4.5.1.5 Xoá mẩu tin

Để xoá một mẩu tin, trước hết ta cũng cần 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 (giải phóng bộ nhớ).

Hai là đánh dấu xoá mẩu tin bằng một cách nào đó. Nghĩa là chỉ xoá mẩu tin một cách logic, vùng không gian nhớ vẫn còn dành cho mẩu tin. Việc đánh dấu có thể được thực hiện 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.
- Mỗi một mẩu tin có một bít xóa, bình thường bit xóa của mẩu tin có giá trị 0, muốn xóa mẩu tin ta đặt cho bit xóa giá trị 1. Với phương pháp này thì một mẩu tin sau khi bị đánh dấu xoá cũng có thể phục hồi được bằng cách đặt bit xoá của mẩu tin giá trị 0.

#### 4.5.1.6 Đánh giá

Đây là một phương pháp tổ chức tập tin đơn giản nhất nhưng kém hiệu quả nhất. Ta thấy tập tin là một danh sách liên kết của các khối nên các thao tác trên tập tin đều đòi hỏi phải truy xuất hầu như tất cả các khối, từ khối đầu tiên đến khối cuối cùng.

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 $\frac{n}{k}$ khối, do đó mỗi lần tìm (hoặc thêm hoặc sửa hoặc xoá) một mẩu tin thì phải truy xuất $\frac{n}{k}$ khối.

### 4.5.2 Tăng tốc độ cho các thao tác tập tin

Nhược điểm của cách tổ chức tập tin tuần tự ở trên là các thao tác trên tập tin rất chậm. 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 phép truy xuất một số nhỏ các khối của tập tin.

Để tạo ra các tổ chức tập tin như vậy chúng ta phải giả sử rằng mỗi mẩu tin có một khoá (key), đó là một tập hợp các trường mà căn cứ vào đó ta có thể phân biệt các mẩu tin với nhau. Hai mẩu tin khác nhau thì khoá của chúng phải khác nhau. Chẳng hạn mã sinh viên trong mẩu tin về sinh viên, biển số xe trong quản lí các phương tiện vận tải đường bộ.

Sau đây ta sẽ xét một số cấu trúc như thế.

### 4.5.3 Tập tin băm (hash files)

#### 4.5.3.1 Tổ chức

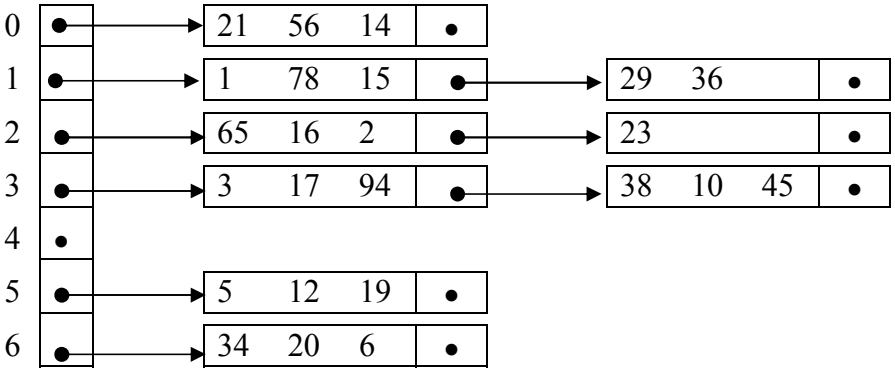
Ta sẽ sử dụng bảng băm mở để lưu trữ tập tin. Bảng băm là một bảng có m phần tử, mỗi phần tử được đánh số từ 0 đến m-1 (đơn giản nhất là mảng một chiều B gồm m phần tử B[0], B[1], ..., B[m-1]). Mỗi phần tử là một con trỏ, trỏ tới phần tử đầu tiên của danh sách liên kết các khối.

Để phân phối các mẩu tin có khóa x vào trong các danh sách liên kết, ta dùng hàm băm (hash function). Hàm băm h(x) ánh xạ mỗi giá trị khoá x với một số nguyên từ 0 đến m-1. Nếu h(x) = i thì mẩu tin r có khóa x sẽ được đưa vào một khối nào đó trong danh sách liên kết được trỏ bởi B[i].

Có nhiều phương pháp để xác định hàm băm. Cách đơn giản nhất là "nguyên hóa" giá trị khóa x (nếu x không phải là một số nguyên) sau đó ta cho h(x) = x MOD m.

**Ví dụ 4-5**: 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)].




Mảng B Các lô được tổ chức bởi các danh sách liên kết.

*Hình 4-2: Tập tin được tổ chức bởi bảng băm*

#### 4.5.3.2 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].

#### 4.5.3.3 Thêm mẩu tin

Để thêm mẩu tin r có khoá x, trước hết ta phải tìm xem đã có mẩu tin nào trong tập tin có khóa x chưa. Nếu có ta cho một thông báo "mẩu tin đã tồn tại" vì theo giả thiết các mẩu tin không có khoá trùng nhau. 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.

#### 4.5.3.4 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).

#### 4.5.3.5 Đánh giá

Giả sử tập tin có n mẩu tin và mỗi khối lưu trữ được k mẩu tin thì tập tin cần $\frac{n}{k}$ khối. Trung bình mỗi danh sách liên kết (mỗi lô) của bảng băm có $\frac{n}{m.k}$ khối (do bảng băm có m lô), mà chúng ta chỉ tìm trong một danh sách liên kết nên ta chỉ phải truy xuất $\frac{n}{m.k}$ 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 $\frac{n}{k}$ khối). Chẳng hạn với 24 mẩu tin như trong ví dụ trên, với cách tổ chức tập tin tuần tự ta cần đúng 8 khối để lưu trữ (vì mỗi khối chứa tối đa 3 mẩu tin). Như vậy để tìm một mẩu tin, chẳng hạn mẩu tin có khóa 36 chúng ta phải đọc đúng 8 khối (do mẩu tin có khóa 36 nằm trong khối cuối cùng). Nhưng với cách tổ chức tập tin bảng băm chúng ta chỉ cần trung bình $\frac{8}{7}$ lần đọc khối . Trong thực tế ta chỉ cần 2 lần đọc khối (vì mẩu tin có khóa 36 nằm trong khối thứ 2 của lô được trỏ bởi B[1]).

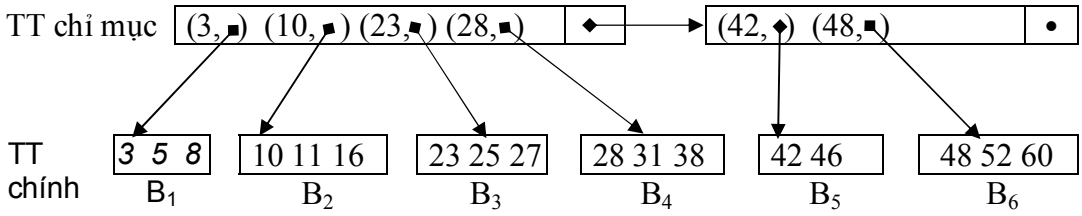
### 4.5.4 Tập tin chỉ mục (index file)

#### 4.5.4.1 Tổ chức

Một cách khác thường gặp là tập tin được sắp xếp theo khoá, rồi chúng ta tiến hành tìm kiếm như là tìm một từ trong từ điển, tức là tìm kiếm theo từ đầu tiên trên mỗi trang.

Để thực hiện được điều đó ta sử dụng hai tập tin: 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.

**Ví dụ 4-6**: Ta có tập tin được tổ chức thành tập tin chỉ mục với mỗi 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. Hình sau minh hoạ tập tin chỉ mục này.




*Hình 4-3: Tập tin chỉ mục*

#### 4.5.4.2 Tìm kiếm

Để tìm mẩu tin r có khoá x, ta phải tìm cặp (z,p) với z là giá trị lớn nhất và $z \leq x$. Mẩu tin r có khoá x nếu tồn tại thì sẽ nằm trong khối được trỏ bởi p.

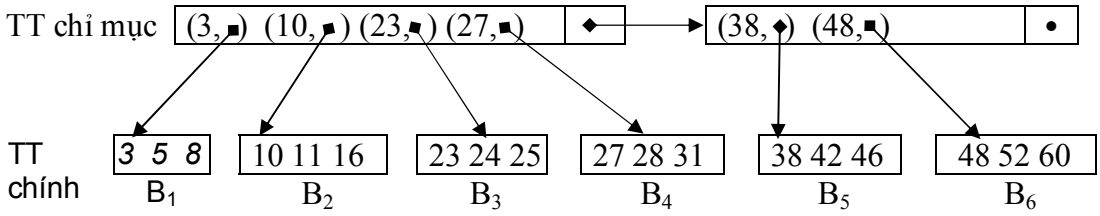
Chẳng hạn để tìm mẩu tin r có khoá 46 trong tập tin của ví dụ 4-6, ta tìm trong tập tin chỉ mục được cặp (42, p), trong đó 42 là giá trị khoá lớn nhất trong tập tin chỉ mục mà $42 \leq 46$ và p là con trỏ, trỏ tới khối $B_5$ của tập tin chính. Trong khối $B_5$, ta tìm thấy mẩu tin có khoá 46.

Việc tìm một mẩu tin trong một khối của tập tin chính có thể tiến hành bằng tìm kiếm tuần tự hoặc bằng tìm kiếm nhị phân bởi lẽ các mẩu tin trong một khối đã được săp thứ tự.

#### 4.5.4.3 Thêm mẩu tin

Giả sử tập tin chính được lưu trong các khối $B_1$, $B_2$, ..., $B_m$. Để xen một mẩu tin r với khóa x vào trong tập tin, ta phải dùng thủ tục tìm kiếm để xác định một khối $B_i$ nào đó. Nếu tìm thấy thì thông báo "mẩu tin đã tồn tại", ngược lại, $B_i$ là nơi có thể chứa mẩu tin r. Nếu $B_i$ còn chỗ trống thì xen r vào đúng vị trí của nó trong $B_i$. Ta phải chỉnh lại tập tin chỉ mục nếu mẩu tin mới trở thành mẩu tin đầu tiên trong khối $B_i$. Nếu $B_i$ không còn chỗ trống để xen thì ta phải xét khối $B_{i+1}$ để có thể chuyển mẩu tin cuối cùng trong khối $B_i$ thành mẩu tin đầu tiên của khối $B_{i+1}$ và xen mẩu tin r vào đúng vị trí của nó trong khối $B_i$ . Điều chỉnh lại tập tin chỉ mục cho phù hợp với trạng thái mới của $B_{i+1}$. Quá trình này có thể dẫn đến việc ta phải xét khối $B_m$, nếu $B_m$ đã hết chỗ thì yêu cầu hệ thống cấp thêm một khối mới $B_{m+1}$, chuyển mẩu tin cuối cùng của $B_m$ sang $B_{m+1}$, mẩu tin cuối cùng của $B_{m-1}$ sang $B_m$… Xen mẩu tin r vào khối $B_i$ và cập nhật lại tập tin chỉ mục. Việc cấp phát thêm khối mới $B_{m+1}$ 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 4-7**: Chẳng hạn ta cần xen mẩu tin r với khóa x = 24 vào trong tập tin được biểu diễn trong hình 4-3. Thủ tục tìm x trong tập tin chỉ mục xác định được khối cần xen r là khối $B_3$. Vì khối $B_3$ đã có đủ 3 mẩu tin nên phải xét khối $B_4$. Khối B4 cũng đã có đủ 3 mẩu tin nên ta lại xét khối $B_5$. Vì $B_5$ còn chỗ trống nên ta chuyển mẩu tin có khoá 38 từ $B_4$ sang $B_5$ và chuyển mẩu tin có khóa 27 từ $B_3$ sang $B_4$ và xen r vào khối $B_3$. Vì mẩu tin đầu tiên của khối $B_4$ bây giờ có khóa 27 nên ta phải sửa lại giá trị này trong cặp của tập tin chỉ mục tương ứng với khối $B_4$. Ta cũng phải làm tương tự đối với khối $B_5$. Cấu trúc của tập tin sau khi thêm mẩu tin r có khóa 24 như sau:




*Hình 4-4: Xen mẩu tin vào tập tin chỉ mục*

#### 4.5.4.4 Xoá mẩu tin

Để xoá mẩu tin r có khoá x, trước hết ta cần 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.

#### 4.5.4.5 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.

### 4.5.5 Tập tin B-cây

#### 4.5.5.1 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.

Chúng ta có thể sử dụng cây m-phân để lưu trữ các mẩu tin trong tập tin trên bộ nhớ ngoài. Mỗi một nút biểu diễn cho một khối vật lý trong bộ nhớ ngoài. Trong đó các nút lá lưu trữ các mẩu tin của tập tin. Các nút trong lưu trữ m con trỏ, trỏ tới m nút con.

Nếu ta dùng cây tìm kiếm nhị phân n nút để lưu trữ một tập tin thì cần trung bình logn phép truy xuất khối để tìm kiếm một mẩu tin. Nếu ta dùng cây tìm kiếm m-phân để lưu trữ một tập tin thì chỉ cần $\log_m n$ phép truy xuất khối để tìm kiếm một mẩu tin. Sau đây chúng ta sẽ nghiên cứu một trường hợp đặc biệt của cây tìm kiếm m-phân là B-cây.

#### 4.5.5.2 B-cây (B-tree)

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.

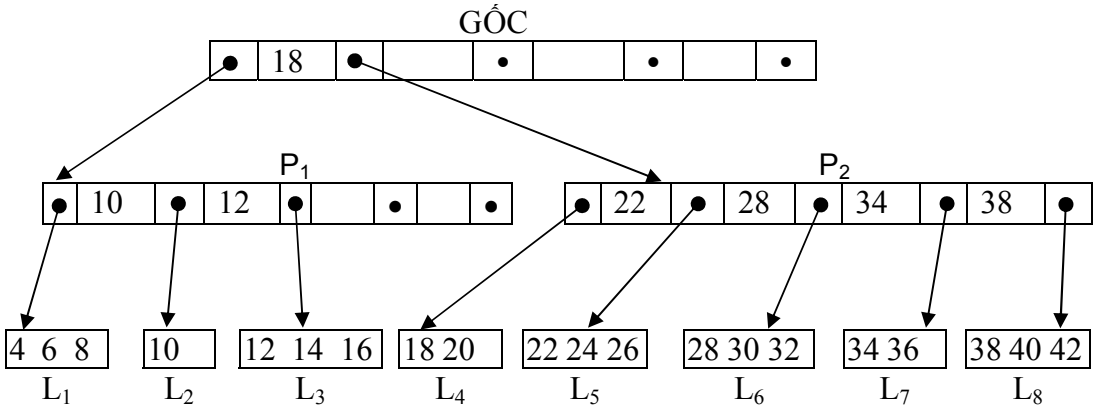
Tổ chức: Ta có thể sử dụng B-cây bậc m để lưu trữ tập tin như sau:

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 $(p_0,k_1,p_1,k_2,p_2,...,k_n,p_n)$, với $p_i$ $(0 \le i \le n)$ là con trỏ, trỏ tới nút con thứ i của nút đó và $k_i$ là các giá trị khóa. Các khoá trong một nút được sắp thứ tự, tức là $k_1 < k_2 < ... < k_n$.

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 $k_n$.

**Ví dụ 4-8**: 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.




*Hình 4-5: Tập tin B-cây bậc 5*

#### 4.5.5.3 Tìm kiếm

Để tìm kiếm một mẩu tin r có khoá là x chúng ta sẽ lần từ nút gốc đến nút lá chứa r (nếu r tồn tại trong tập tin). Tại mỗi bước ta đưa nút trong $(p_0, k_1, p_1, ..., k_n, p_n)$ vào bộ nhớ trong và xác định mối quan hệ giữa x với các giá trị khóa $k_i$.

- Nếu $k_i \le x < k_{i+1}$ $(0 < i < n)$ chúng ta sẽ xét tiếp nút được trỏ bởi $p_i$,
- Nếu $x < k_1$ ta sẽ xét tiếp nút được trỏ bởi $p_0$ và
- Nếu $x \ge k_n$ ta sẽ xét tiếp nút được trỏ bởi $p_n$.

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.

#### 4.5.5.4 Thêm mẩu tin

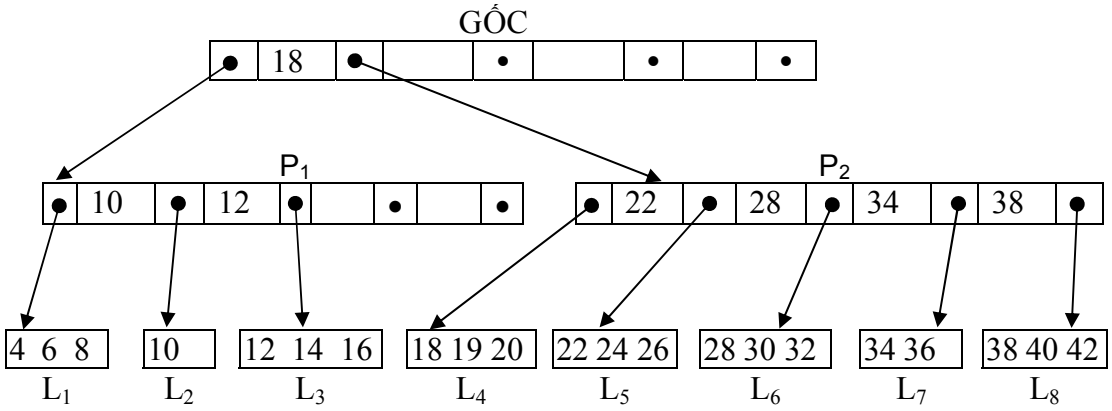
Để thêm một mẩu tin r có khoá là x vào trong B-cây, trước hết ta áp dụng thủ tục tìm kiếm nói trên để tìm r. Việc tìm kiếm này sẽ dẫn đến nút lá L. Nếu tìm thấy thì thông báo "Mẩu tin đã tồn tại", ngược lại thì L là nút lá mà ta có thể xen r vào trong đó. Nếu khối L này còn đủ chỗ cho r thì ta thêm r vào sao cho đúng thứ tự của nó trong khối L và giải thuật kết thúc. Nếu L không còn chỗ cho r thì ta yêu cầu hệ thống cấp phát một khối mới L'. Dời $\lceil b/2 \rceil$ (b là số mẩu tin nhiều nhất có thể lưu

trong một khối) mẩu tin nằm ở phân nửa 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 (P phải được biết vì thủ tục tìm đi từ gốc đến L phải thông qua P). Bây giờ ta áp dụng thủ tục xen đệ quy để xen vào P một khóa k' và con trỏ p' tương ứng của nút lá L' (k' là khoá của mẩu tin đầu tiên trong 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 này đòi hỏi 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.

**Ví dụ 4-9**: Thêm mẩu tin r có khoá 19 vào tập tin được biểu diễn bởi B-cây trong ví dụ 4-8 (hình 4-5)

- Quá trình tìm kiếm sẽ xuất phát từ GỐC đi qua $P_2$ và dẫn tới nút lá $L_4$
- Trong nút lá $L_4$ còn đủ chỗ để xen r vào đúng vị trí và giải thuật kết thúc.

Kết quả việc xen ta được B-cây trong hình 4-6:

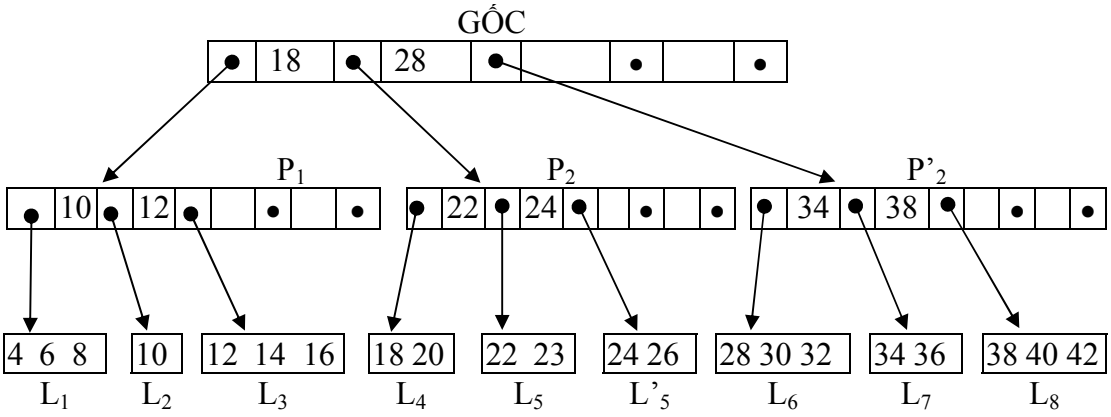



*Hình 4-6: Xen thêm mẩu tin r có khoá 19 vào trong B-cây hình 4-5*

**Ví dụ 4-10**: Thêm mẩu tin r có khoá 23 vào trong tập tin biêu diễn bởi B-cây trong ví dụ 4-8 (hình 4-5)

- Quá trình tìm kiếm đi từ nút GỐC, qua $P_2$ và tới nút lá $L_5$.
- Vì $L_5$ đã đủ 3 mẩu tin nên phải tạo ra một nút lá mới $L'_5$ và chuyển 2 mẩu tin có khóa 24, 26 sang $L'_5$ sau đó xen r vào $L_5$.
- Giá trị khóa của mẩu tin đầu tiên trong $L'_5$ là 24, ta phải xen 24 và con trỏ của $L'_5$ vào $P_2$, nhưng $P_2$ đã có đủ 5 con, vậy cần tạo ra một nút mới $P'_2$, chuyển các cặp khóa, con trỏ tương ứng với 34 và 38 sang $P'_2$ và xen cặp con trỏ, khóa 24 vào $P_2$.
- Do có một nút mới $P'_2$ nên phải xen vào cha của P2 (Ở đây là nút GỐC) một cặp khóa, con trỏ trỏ tới $P'_2$. Con trỏ $p_0$ của nút $P'_2$ trỏ tới nút lá $L_6$,

giá trị khóa đầu tiên của $L_6$ là 28. Giá trị này phải được xen vào nút GỐC cùng với con trỏ của $P'_2$.




*Hình 4-7: Xen thêm mẩu tin r có khoá 23 vào trong B-cây hình 4-5*

#### 4.5.5.5 Xóa một mẩu tin

Để xóa mẩu tin r có khóa x, trước hết ta tìm mẩu tin r. Nếu không tìm thấy thì thông báo « Mẩu tin không tồn tại », ngược lại ta sẽ xác đinh được mẩu tin r nằm trong nút lá L và xóa r khỏi L. Nếu r là mẩu tin đầu tiên của L, thì ta phải quay lui lên nút P là cha của L để đặt lại giá trị khóa của L trong P, giá trị mới này bằng giá trị khóa của mẩu tin mới đầu tiên của L. Trong trường hợp L là con đầu tiên của P, thì khóa của L không nằm trong P mà nằm trong tổ tiên của P, chúng ta phải quay lui lên mà sửa đổi.

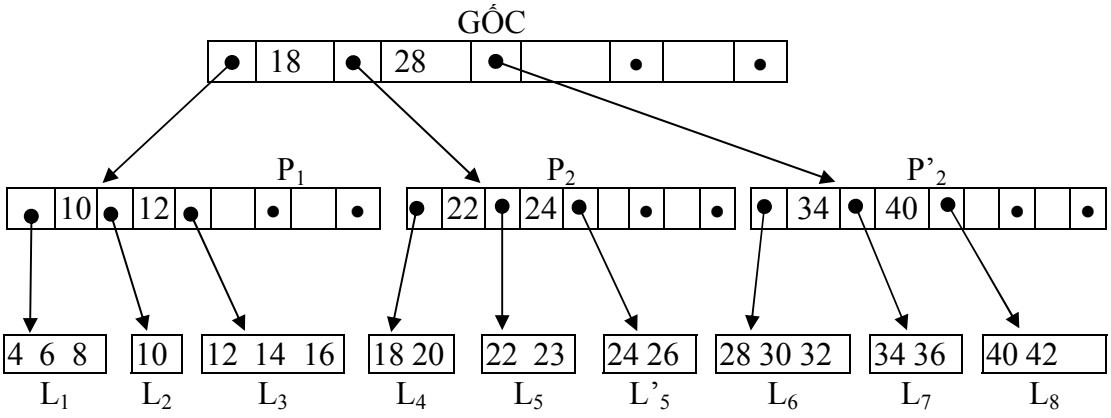
Nếu sau khi xóa mẩu tin r mà L trở nên rỗng thì giải phóng L và quay lui lên nút P là cha của L để xoá cặp khoá-con trỏ của L trong P. Nếu số con của P bây giờ (sau khi xoá khoa-con trỏ của L) nhỏ hơn $\lceil m/2 \rceil$ thì kiểm tra nút P' ngay bên trái hoặc bên phải và cùng mức với P. Nếu P' có ít nhất $\lceil m/2 \rceil+1$ con, chúng ta chuyển một con của P' sang P. Lúc này cả P và P' có ít nhất $\lceil m/2 \rceil$. Sau đó ta phải cập nhật lại giá trị khóa của P hoặc P' trong cha của chúng, và nếu cần chúng ta phải sửa cả trong tổ tiên của chúng.

Nếu P' có đúng $\lceil m/2 \rceil$ con, ta nối P và P' thành một nút có đúng m con. Sau đó ta phải xóa khóa và con trỏ của P' trong nút cha của P'. Việc xóa này có thể phải quay lui lên tổ tiên của P'. Kết quả của quá trình xóa đệ quy này có thể dẫn tới việc nối hai con của nút gốc, tạo nên một gốc mới và giải phóng nút gốc cũ, độ cao của cây khi đó sẽ giảm đi 1.

**Ví dụ 4-11**: Xóa mẩu tin r có khóa 38 trong tập tin biểu diễn bởi B-cây kết quả của ví dụ 4-10 (hình 4-7).

- Quá trình tìm kiếm, xuất phát từ nút GỐC, đi qua $P'_2$ và đến nút lá $L_8$,
- Xóa mẩu tin r khỏi $L_8$.
- Mẩu tin đầu tiên của $L_8$ bây giờ có khóa 40,

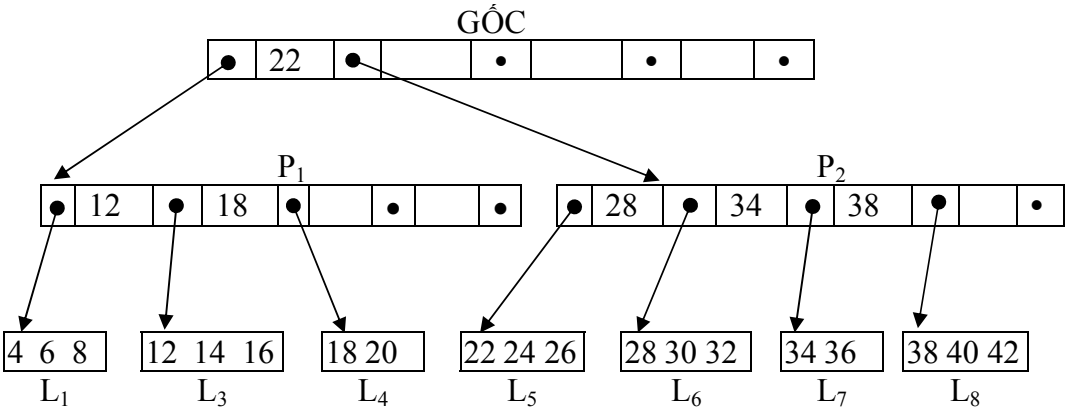
- Sửa lại giá trị khóa của $L_8$ trong $P'_2$ (thay 38 bởi 40) ta được kết quả là B-cây sau:




*Hình 4-8: Xoá mẩu tin r có khoá 38 vào trong B-cây hình 4-7*

**Ví dụ 4-12 :** Xoá mẩu tin r có khoá 10 trong tập tin biểu diễn bởi B-cây trong ví dụ 4-8 (hình 4-5).

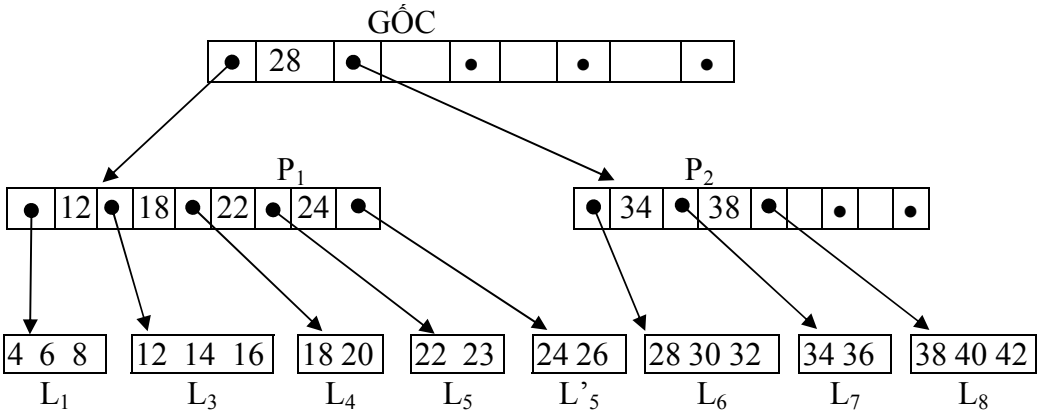
- Quá trình tìm kiếm, xuất phát từ nút GỐC, đi qua $P_1$ và đến nút lá $L_2$.
- Xóa mẩu tin r khỏi $L_2$.
- $L_2$ bây giờ trở nên rỗng, giải phóng $L_2$.
- Xóa giá trị khóa 10 và con trỏ của $L_2$ trong $P_1$, $P_1$ bây giờ chỉ có 2 con (Thiếu con do $2 < \lceil 5/2 \rceil$).
- Xét nút $P_2$, bên phải và cùng cấp với $P_1$, $P_2$ có 5 con nên ta chuyển một con từ $P_2$ sang $P_1$.
- Cập nhật lại khoá của $P_2$ trong nút GỐC, ta được B-cây kết quả như sau:




*Hình 4-9: Xoá mẩu tin có khoá 10 trong B-cây hình 4-5*

**Ví dụ 4-13:** Xóa mẩu tin r có khóa 10 trong tập tin biểu diễn bởi B-cây kết quả của ví dụ 4-10 (hình 4-7).

- Quá trình tìm kiếm, xuất phát từ nút GỐC, đi qua $P_1$ và lần đến nút lá $L_2$.
- Xóa mẩu tin r khỏi $L_2$.
- $L_2$ bây giờ trở nên rỗng, giải phóng $L_2$.
- Xóa giá trị khóa 10 và con trỏ của $L_2$ trong $P_1$, $P_1$ bây giờ chỉ có 2 con (Thiếu con do $2 < \lceil 5/2 \rceil$).
- Xét nút $P_2$, bên phải và cùng cấp với $P_1$, $P_2$ có đúng $\lceil 5/2 \rceil = 3$ con nên ta nối $P_1$ và $P_2$ để $P_1$ có đúng 5 con, giải phóng $P_2$.
- Xóa khóa và con trỏ của $P_2$ trong nút GỐC, ta được B-cây kết quả như sau:




*Hình 4-10: Xoá mẩu tin r có khoá 10 trong B-cây hình 4-7*

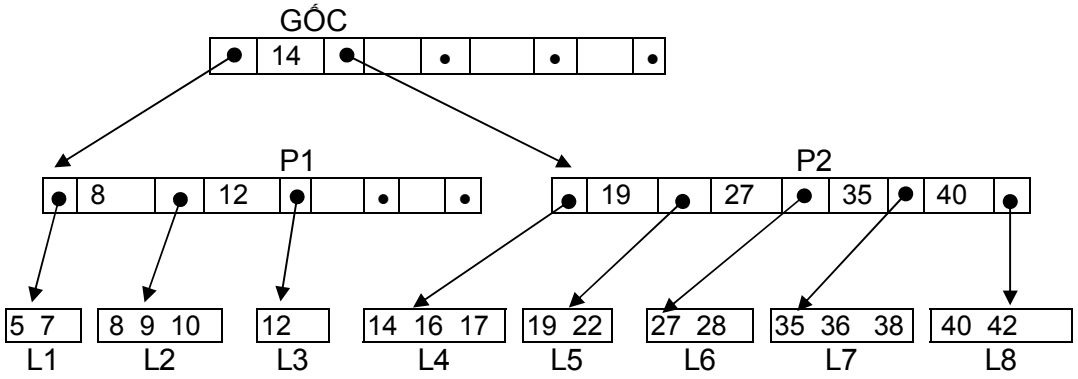
## 4.6 TỔNG KẾT CHƯƠNG 4

Để đánh giá các giải thuật xử lí ngoài, cần phải xác định số phép truy xuất khối (đọc và ghi khối) mà giải thuật đó thực hiện. Theo đó, một giải thuật được xem là tốt nếu số lượng phép truy xuất khối nhỏ và để cải tiến giải thuật, ta cần tìm cách giảm số phép truy xuất khối. Các giải thuật sắp xếp trộn minh hoạ khá rõ ràng cho việc cải tiến giải thuật xử lí ngoài.

Đối với việc tổ chức lưu trữ thông tin trong tập tin, chúng ta cần chú ý đến các loại tập tin bảng băm và tập tin B-cây, đây là hai loại tập tin rất hiệu quả.

## BÀI TẬP CHƯƠNG 4

**Bài 1:** Cho tập tin bao gồm các 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** như sau.




a) Xen mẩu tin R có giá trị khóa là 37 vào tập tin được biểu diễn bởi B-cây nói trên.
b) Xóa mẩu tin R có giá trị khóa là 12 của tập tin được biểu diễn bởi B-cây nói trên.
c) Xóa mẩu tin R có giá trị khóa là 12 của tập tin được biểu diễn bởi B-cây là kết quả của câu a).

**Bài 2**: Giả sử ta dùng B-cây **bậc 3** với các nút lá chứa được nhiều nhất **2 mẩu tin** để tổ chức tập tin. Khởi đầu tập tin rỗng, hãy mô tả quá trình hình thành tập tin B-cây (bằng hình vẽ) khi thực hiện tuần tự các thao tác sau:

1. Xen mẩu tin R có khóa 8
2. Xen mẩu tin R có khóa 2
3. Xen mẩu tin R có khóa 10
4. Xen mẩu tin R có khóa 1
5. Xen mẩu tin R có khóa 12
6. Xen mẩu tin R có khóa 3
7. Xen mẩu tin R có khóa 5
8. Xóa mẩu tin R có khóa 8
9. Xóa mẩu tin R có khóa 1