Các cấu trúc dữ liệu nâng cao

(Advanced Data Structures)

B-Cây

- Đặt vấn đề
- Truy xuất dữ liệu trên bộ nhớ ngoài
- m-way search tree
- Định nghĩa B-cây
- Lưu trữ B-cây trên bộ nhớ ngoài
- Khai báo cấu trúc B-cây
- Các thao tác cơ bản B-cây

Đặt vấn đề

- Các ứng dụng database
- Cần lưu trữ dữ liệu lớn (vd. 1,000,000 – 1,000,000,000 phần tử)
- Lưu trữ trên bộ nhớ ngoài
- Tốc độ tìm kiếm nhanh

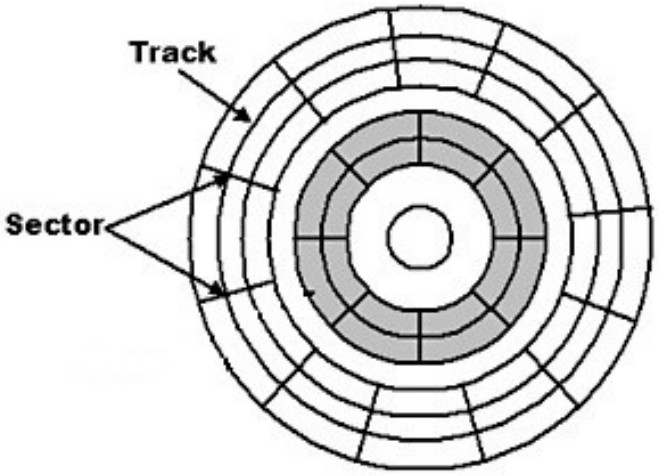
Truy xuất dữ liệu trên bộ nhớ ngoài (1)

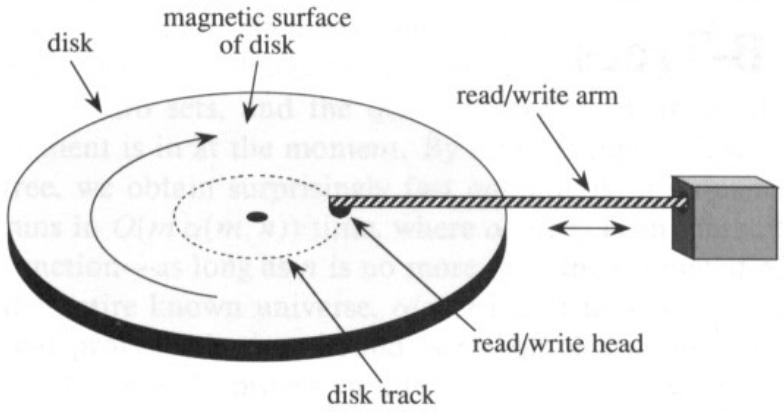
- Bộ nhớ ngoài: HDD, DVD, tape,…
- Đơn vị truy xuất tối thiểu ?
- Thời gian truy xuất ?

Truy xuất dữ liệu trên bộ nhớ ngoài (2)

- Thời gian để đọc/ghi một block

```
t = thời gian dịch chuyển đầu đọc đến block +
    thời gian đọc/ghi block vào bộ nhớ
```





Truy xuất dữ liệu trên bộ nhớ ngoài (3)

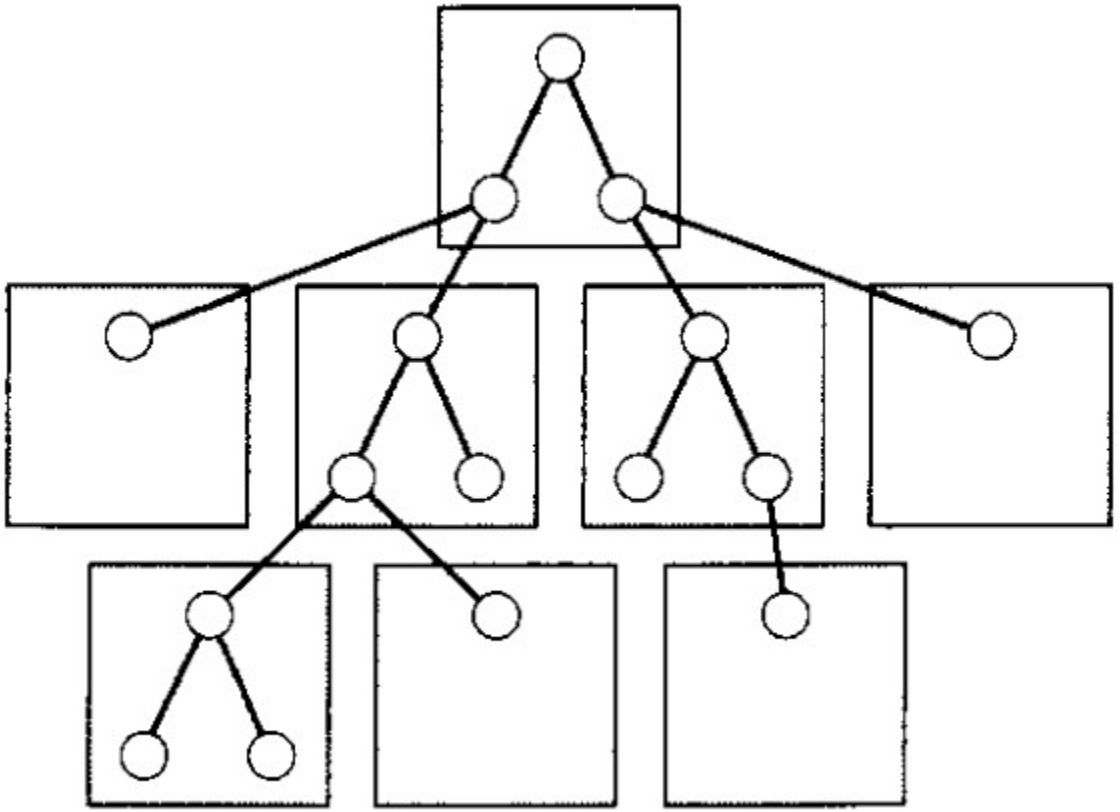
- Vd 1. thời gian để đọc 2 block liên tiếp, mỗi block=1KB

```
t1 = 20ms + (5ms + 5ms) = 30ms
```

- Vd 2. thời gian để đọc 2 block xa nhau, mỗi block=1KB

```
t2 = 2 * (20ms + 5ms) = 50ms
```

m-way Search Tree (1)

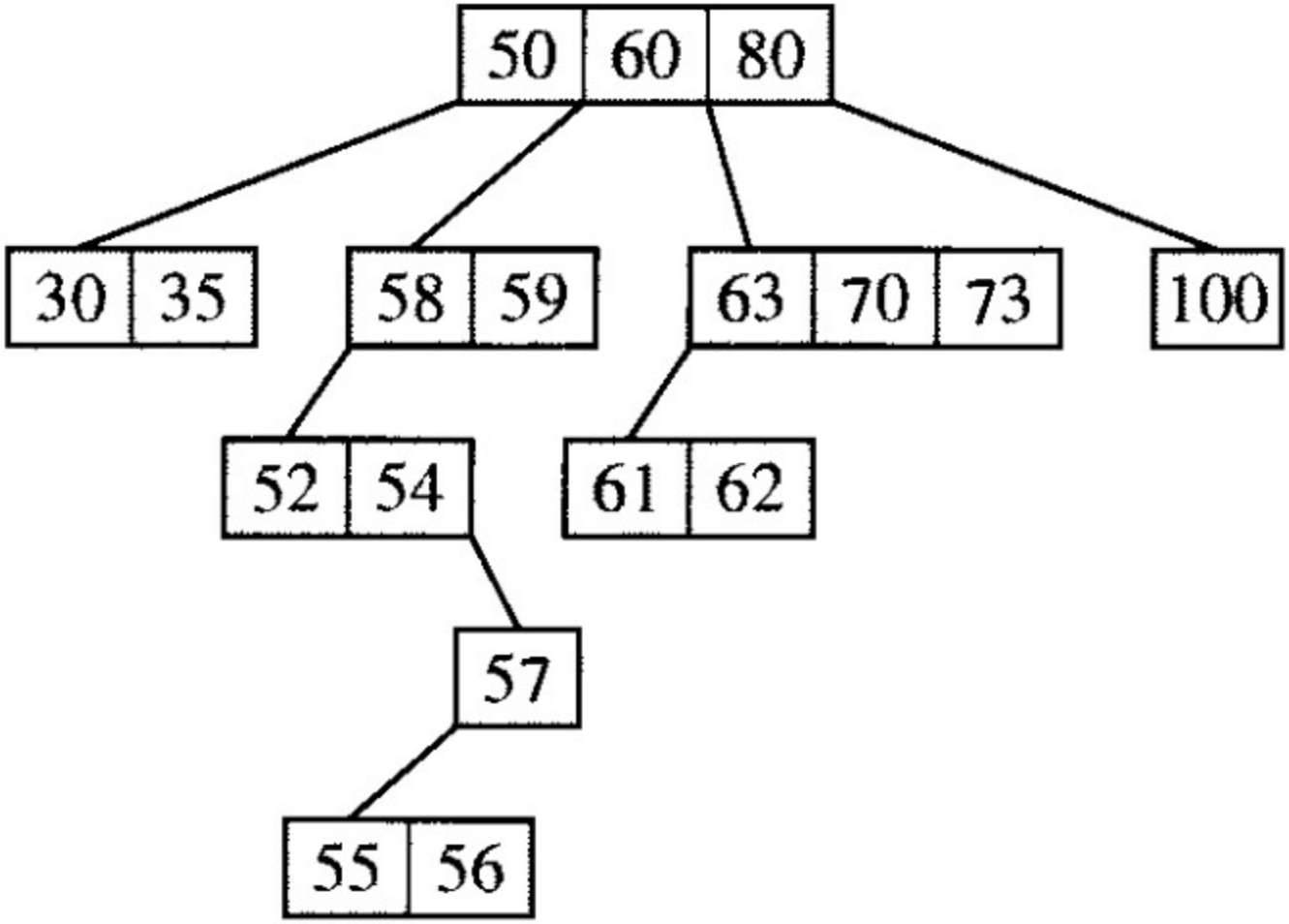


Cây nhị phân với các phần tử được gom thành từng block (trên đĩa)

m-way Search Tree (2)

- Định nghĩa: m-way search tree là cây thỏa
  - Mỗi node có tối đa m cây con và (m-1) khóa
  - Các khóa trong mỗi node sắp tăng dần
  - Các khóa trong cây con thứ i đều nhỏ hơn khóa i
  - Các khóa trong cây con thứ (i+1) đều lớn hơn khóa i

m-way Search Tree (3)



Cây tìm kiếm m-way, thao tác tìm kiếm hoạt động tương tự BST

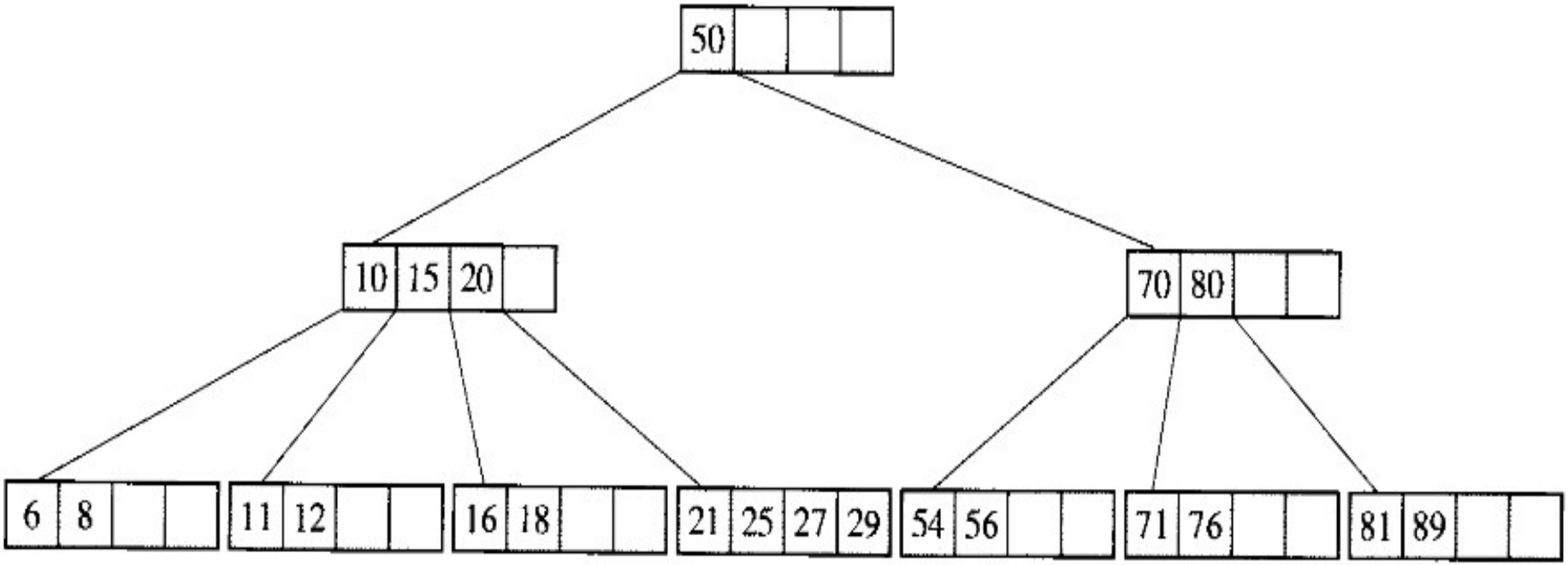
Định nghĩa B-cây (1)

- Định nghĩa: B-cây bậc m (m>2) là m-way search tree thỏa
  - Node gốc có ít nhất là 1 khóa và 2 cây con, ngoại trừ khi nó là node lá
  - Mỗi node lá có ít nhất (m-1)/2 khóa
  - Mỗi *node trong* có ít nhất (m-1)/2 khóa và ít nhất (m-1)/2+1 cây con
  - Tất cả node lá có cùng mức

** Node trong (internal node): là node không phải gốc và không phải lá*

** B-cây được giới thiệu vào năm 1972 bởi Bayer và McCreight*

Định nghĩa B-cây (2)



B-cây bậc 5

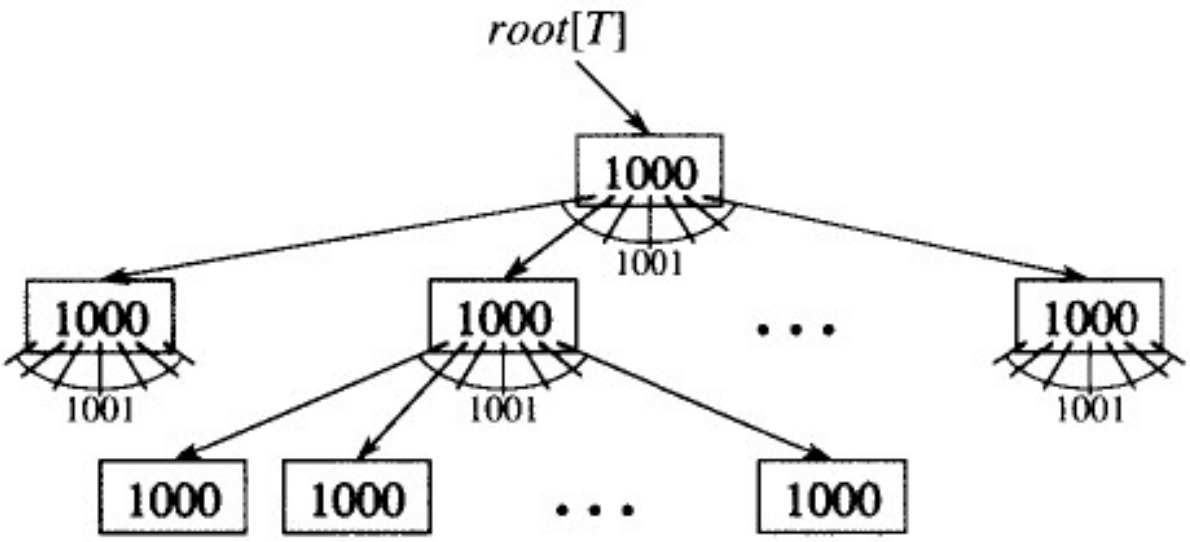
Định nghĩa B-cây (3)

- Với m=3, ta có cây 2-3 (2-3 tree)
  - Mỗi node trong có 2 hay 3 cây con
  - *Cây 2-3 được phát minh năm 1970 bởi J. E. Hopcroft*
- Với m=4, ta có cây 2-3-4 (2-3-4 tree)
  - Mỗi node trong có 2, 3 hay 4 cây con

Định nghĩa B-cây (4)

- Ý nghĩa:
  - B-cây là cây cân bằng hoàn toàn
  - Mỗi node được lấp đầy ít nhất 50%. Các phân tích và thử nghiệm thực tế cho thấy các node của B-cây trong trường hợp bình thường được lấp đầy ~70%
  - B-cây sử dụng số phép truy xuất đĩa tối thiểu cho các thao tác
  - Thích hợp với việc lưu trữ trên bộ nhớ ngoài
  - Có thể quản lý số phần tử rất lớn

Định nghĩa B-cây (5)



1 node,
1000 keys

1001 nodes,
1,001,000 keys

1,002,001 nodes,
1,002,001,000 keys

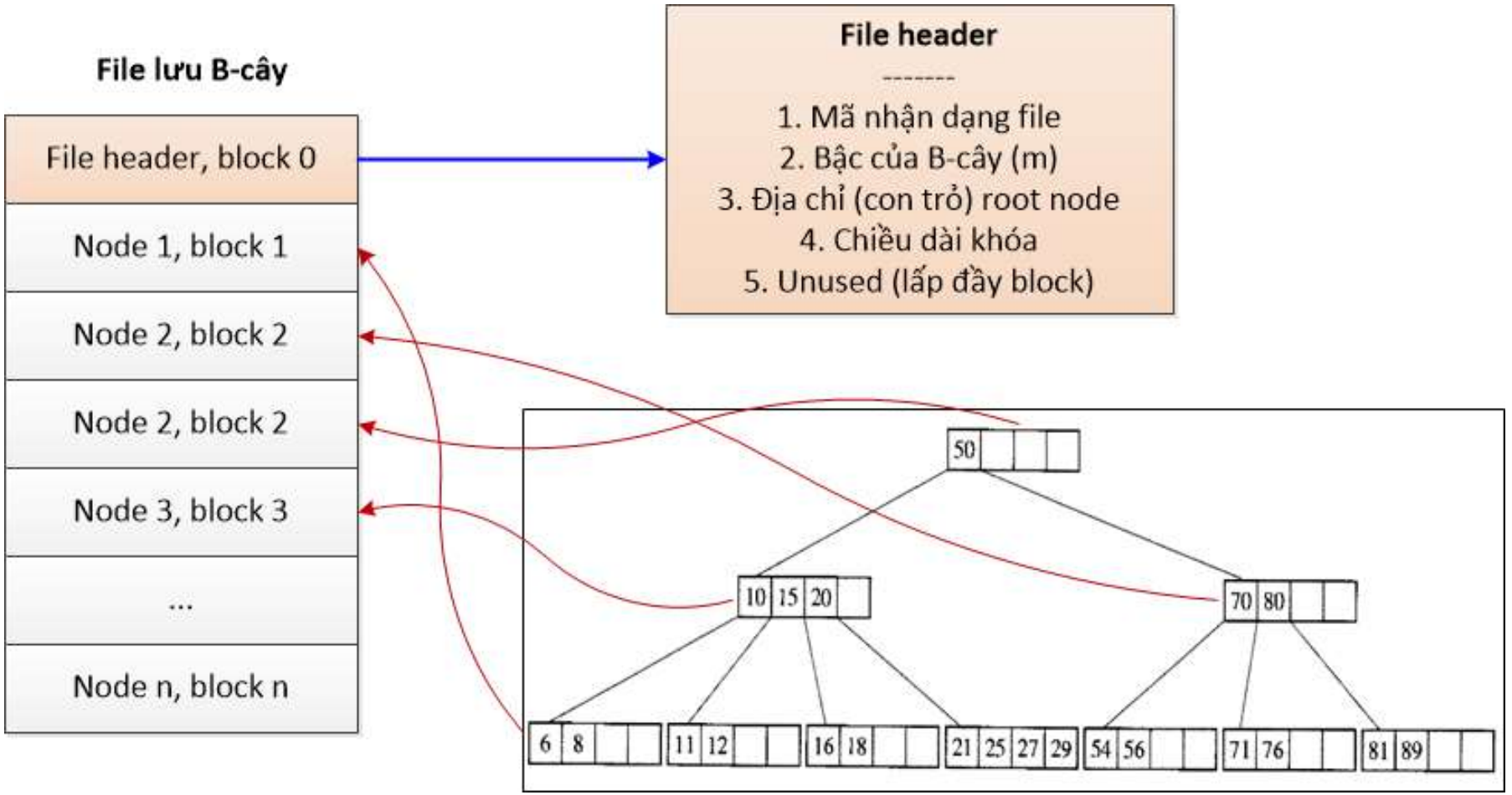
Cây 1001 nhánh, chỉ 3 mức → chứa hơn 1 tỉ phần tử

Định nghĩa B-cây (6)

- Độ cao của B-cây:
  - n: số khoá, n >= 1
  - m: bậc của cây, m > 2

$$h \leq \log_{m} \frac{n+1}{2}$$

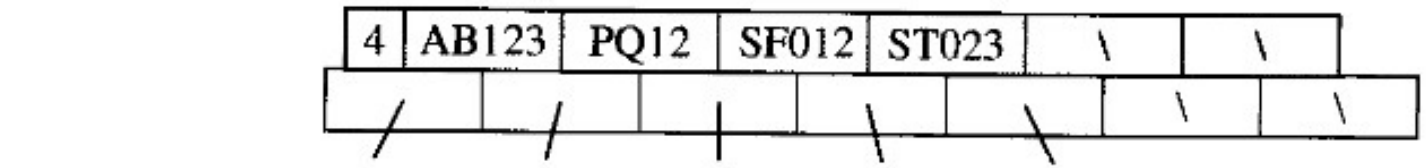
Lưu trữ B-cây trên bộ nhớ ngoài (1)



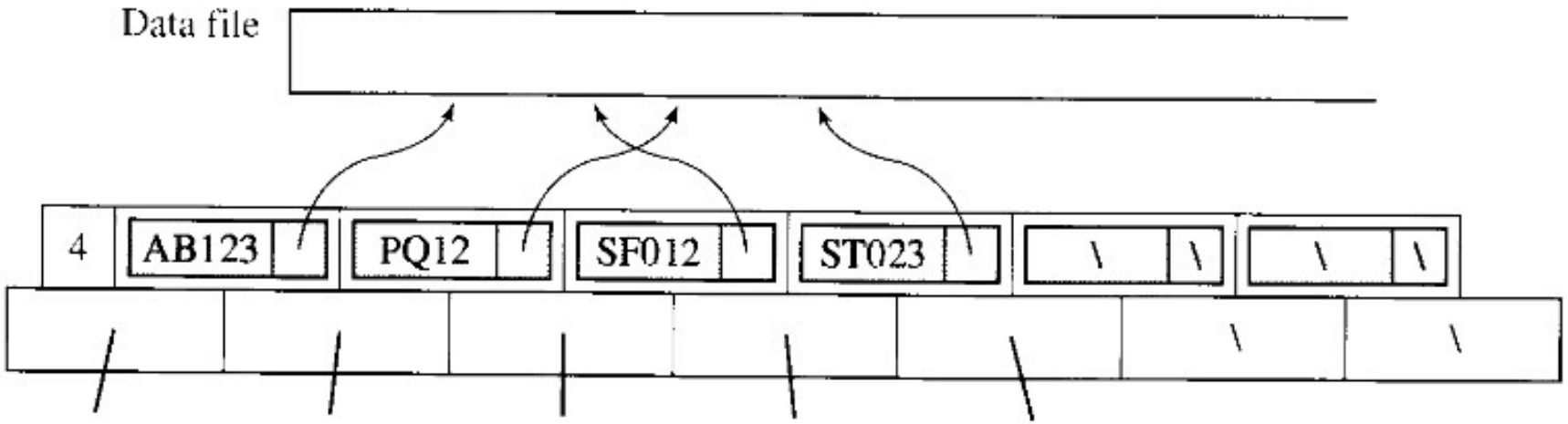
Lưu trữ B-cây trên bộ nhớ ngoài (2)

- Node gốc nên được lưu thường xuyên trong bộ nhớ
  - Không cần thực hiện thao tác READ_ROOT
  - Thao tác WRITE_ROOT được thực hiện khi node gốc thay đổi

Lưu trữ B-cây trên bộ nhớ ngoài (3)



(a)



(b)

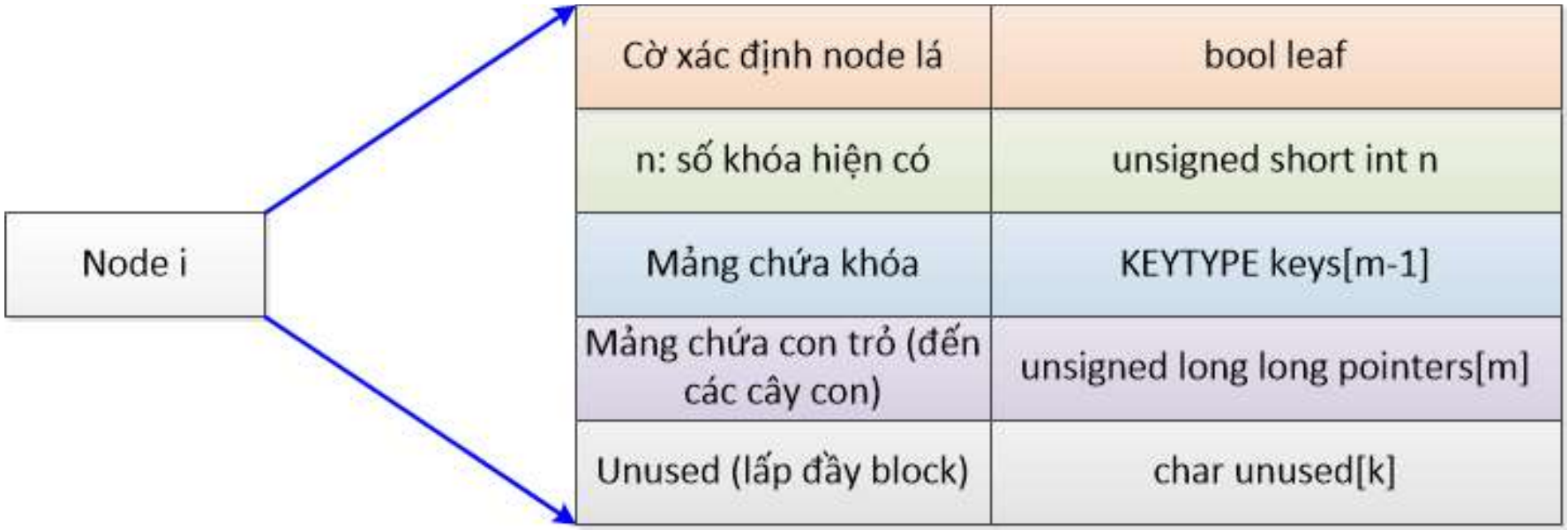
(a) B-cây với các khóa không có thông tin phụ

(b) B-cây có thêm thông tin phụ cho mỗi khóa

Khai báo cấu trúc B-cây (1)

- Hãy xây dựng cấu trúc dữ liệu và khai báo (struct/class) cho 1 node của B-cây ?
- Hãy xây dựng cấu trúc dữ liệu và khai báo (struct/class) cho header của file chứa B-cây ?

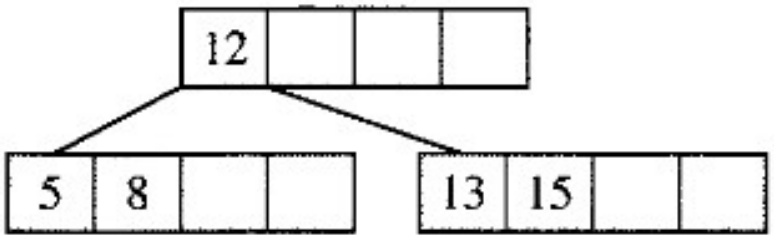
Khai báo cấu trúc B-cây (2)



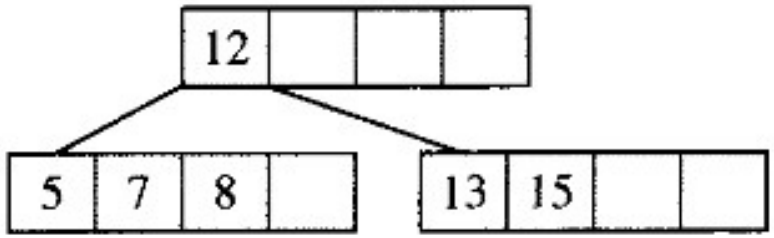
Các thao tác cơ bản trên B-cây

- Tìm kiếm một khóa
- Thêm một khóa
- Xóa một khóa

Thêm một khóa vào B-cây (1)



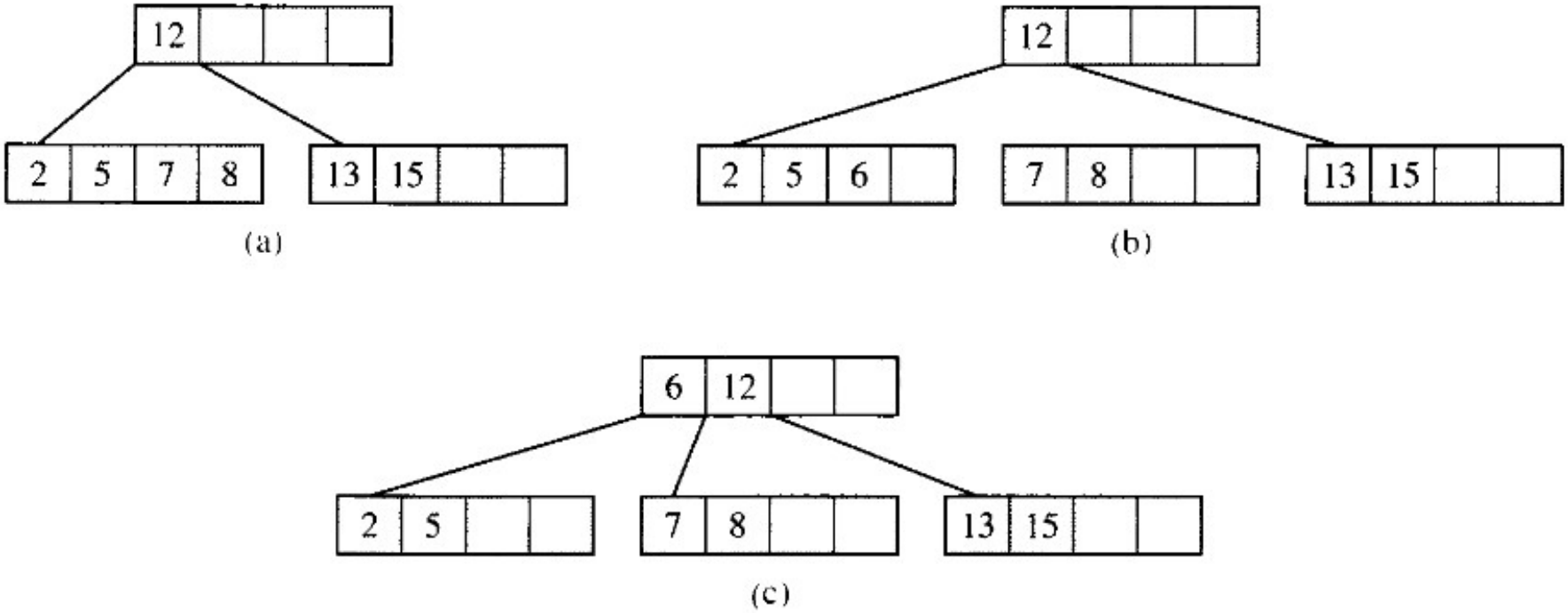
(a)



(b)

Khóa 7 được thêm vào node lá khi node này còn chỗ trống

Thêm một khóa vào B-cây (2)

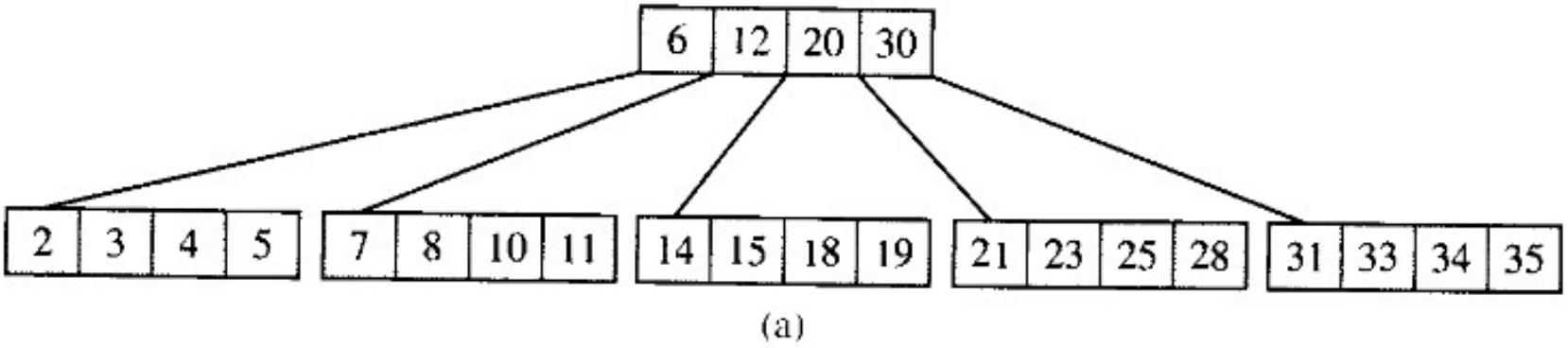


Khóa 6 được thêm vào node lá đã đầy → split node và chuyển khóa giữa lên node cha

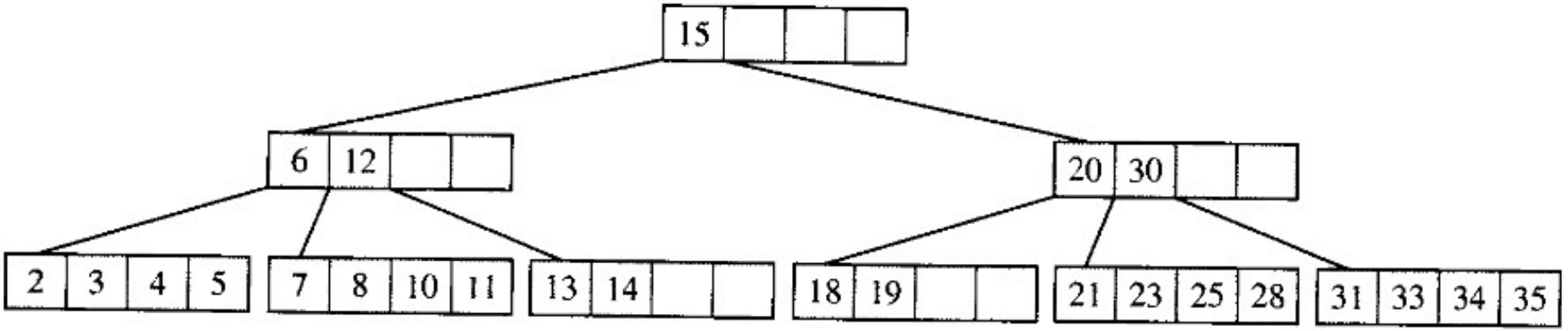
Thêm một khóa vào B-cây (3)

- Thuật toán:
  - Khóa được thêm vào node lá nếu node còn chỗ trống. Các khóa trong node sắp thứ tự tăng dần
  - Nếu node lá chứa khóa đầy → tách node (split) bằng cách tạo node mới; copy (m-1)/2 khóa sang node mới; chuyển khóa giữa lên node cha; tạo con trỏ từ node cha đến node mới. Quá trình tách node có thể thực hiện liên tiếp cho các node trong của B-cây
  - Trường hợp xấu nhất, node gốc cũng bị tách và tạo thành node gốc mới

Thêm một khóa vào B-cây (4)

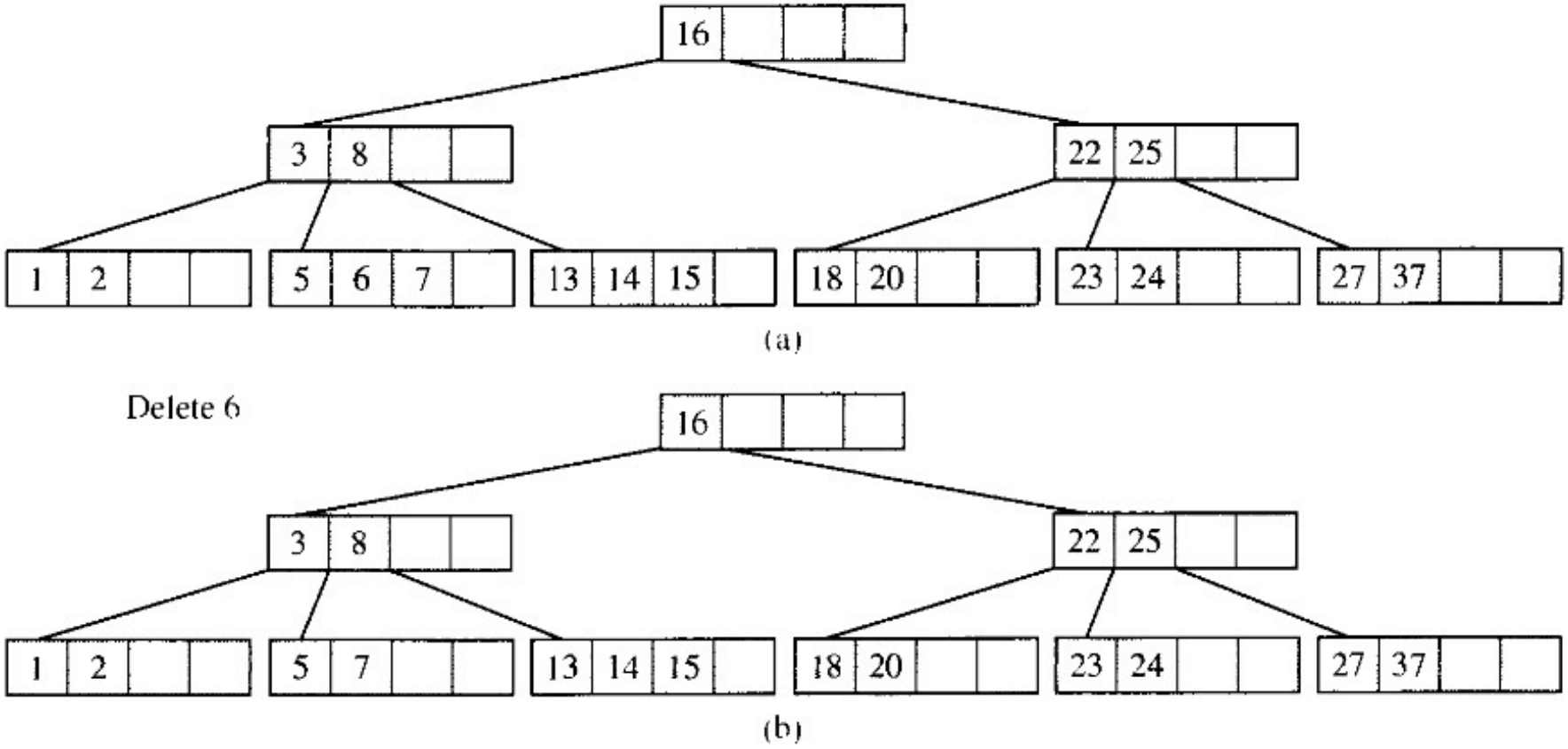


(a)



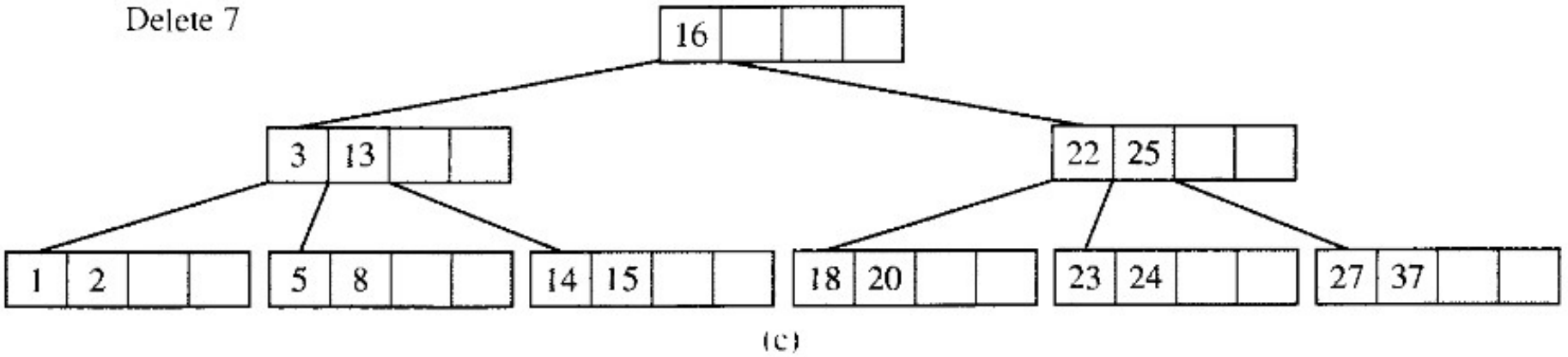
Split node nhiều lần dẫn tới split node gốc → tạo thành node gốc mới

Xóa một khóa của B-cây (1)



Xóa khóa 6 ở node lá khi node này dư khóa

Xóa một khóa của B-cây (2)

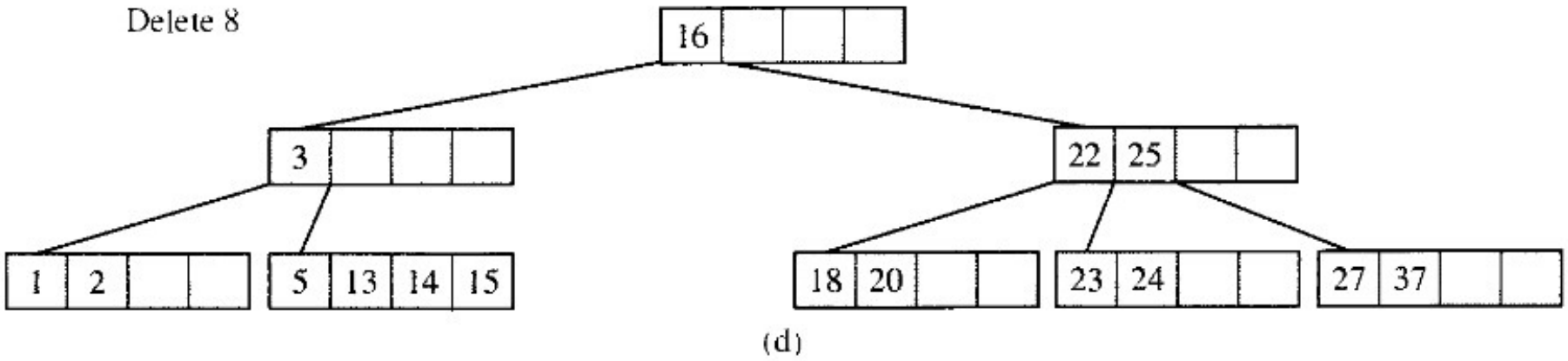


Xóa khóa 7 ở node lá → node thiếu khóa nhưng node anh/em có khóa dư → mượn khóa từ node anh/em

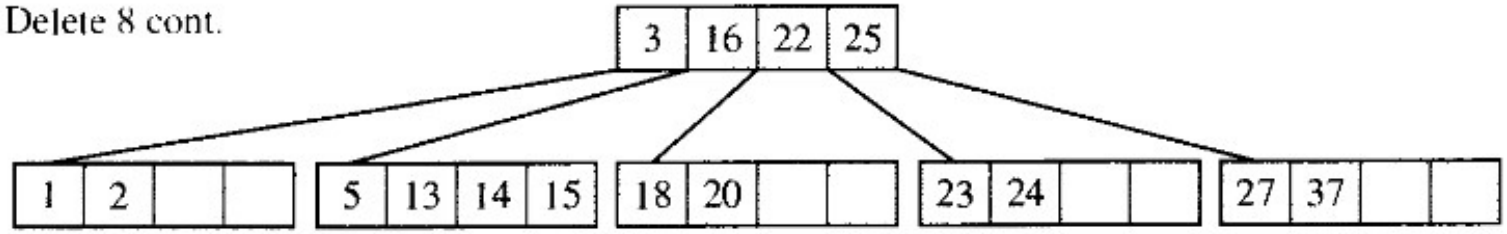
Xóa một khóa của B-cây (3)

- Thuật toán:
  - Xóa khóa ở node lá
    - Nếu sau khi xóa key, số khóa trong node >= (m-1)/2 → stop
    - Nếu sau khi xóa key, node có ít hơn (m-1)/2 khóa
      - Nếu node anh/em có > (m-1)/2 khóa → mượn khóa từ node anh/em
      - Nếu node anh/em có <= (m-1)/2 khóa → merge node và đưa khóa từ node cha xuống. Nếu node cha thiếu khóa → xử lý tương tự như node lá.
  - Xóa khóa ở node không phải là node lá
    - Áp dụng phương pháp *"tìm phần tử thay thế"* và xóa key ở node lá

Xóa một khóa của B-cây (4)



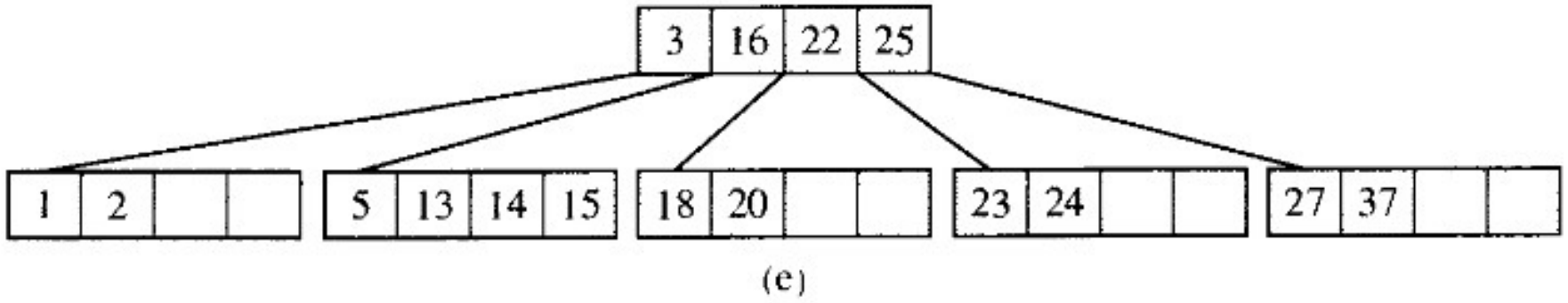
(d)



(e)

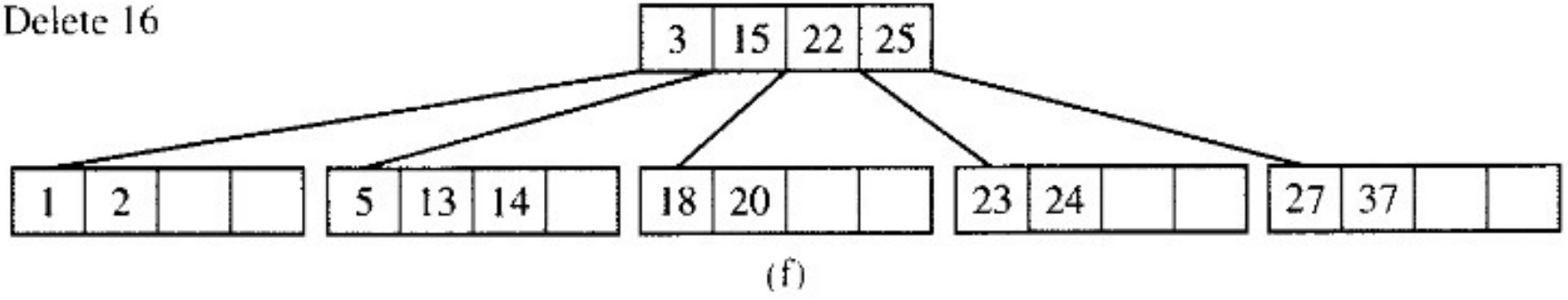
Xóa khóa 8 ở node lá → node thiếu khóa và node anh/em KHÔNG có khóa dư → gộp (merge) node và đưa khóa ở node cha xuống…

Xóa một khóa của B-cây (5)



(e)

Delete 16



(f)

Xóa khóa 16 ở node không phải là node lá → áp dụng phương pháp *"tìm phần tử thay thế"*