Trường ĐHBK Hà Nội
Viện Điện Tử - Viễn Thông

Chương 11: Cấu trúc cây

Phần 2: Cân bằng cây

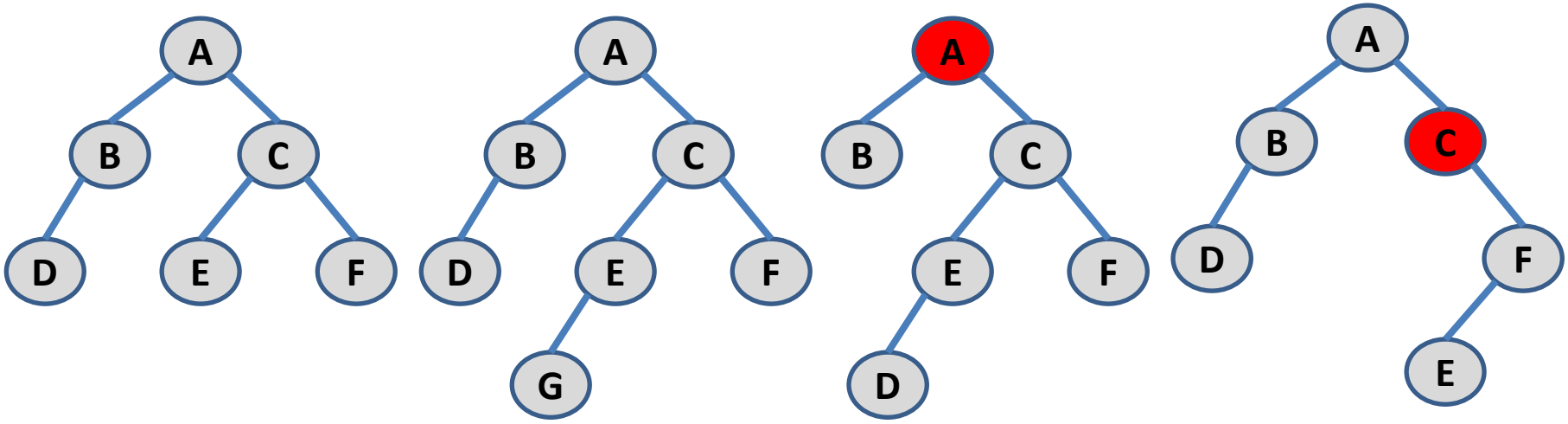
Nội dung chính

- Khái niệm cơ bản về cây cân bằng
- Vấn đề mất cân bằng cây
- Các giải thuật cân bằng cây:
  - Giải thuật đơn giản
  - Giải thuật DSW
  - Cây AVL

Khái niệm về cây cân bằng

- Cây cân bằng về chiều cao (height balanced tree), hay ngắn gọn là cây cân bằng, là cây mà tại mọi nút đều có độ chênh lệch về chiều cao của hai cây con (trái và phải) không lớn hơn 1

Ví dụ



a. **b**. **c**. **d**.

Các cây cân bằng

Các cây mất cân bằng

Vấn đề mất cân bằng

- Nhận xét: Trong số các cây có số nút bằng nhau, thì cây cân bằng có chiều cao nhỏ nhất
- Với cây mất cân bằng thì thời gian tìm kiếm có thể bị tăng lên, vì thời gian tìm kiếm thường tỉ lệ với chiều cao cây.
- Ví dụ với cây nhị phân tìm kiếm, khi bổ sung hay loại bỏ các nút có thể gây ra mất cân bằng cây

➔*Cân bằng cây sẽ giúp tối ưu thao tác tìm kiếm, nhưng lại tốn thêm chi phí cho phép cân bằng cây.*

➔*Sau này ta chỉ quan tâm cân bằng lại cây NP tìm kiếm.*

Các giải thuật cân bằng cây

- Giải thuật đơn giản (simple balance algorithm)
- Giải thuật DSW (Day, Stout, Warren algorithm)
- Cây AVL (Adelson, Velskii, Landis)

Giải thuật cân bằng đơn giản

- Ý tưởng chung của giải thuật:
  - Copy các nút của cây ra một mảng
  - Sắp xếp các phần tử của mảng theo trật tự tăng dần
  - Xóa cây ban đầu
  - Xây dựng lại cây bằng giải thuật *Cân bằng đơn giản*.

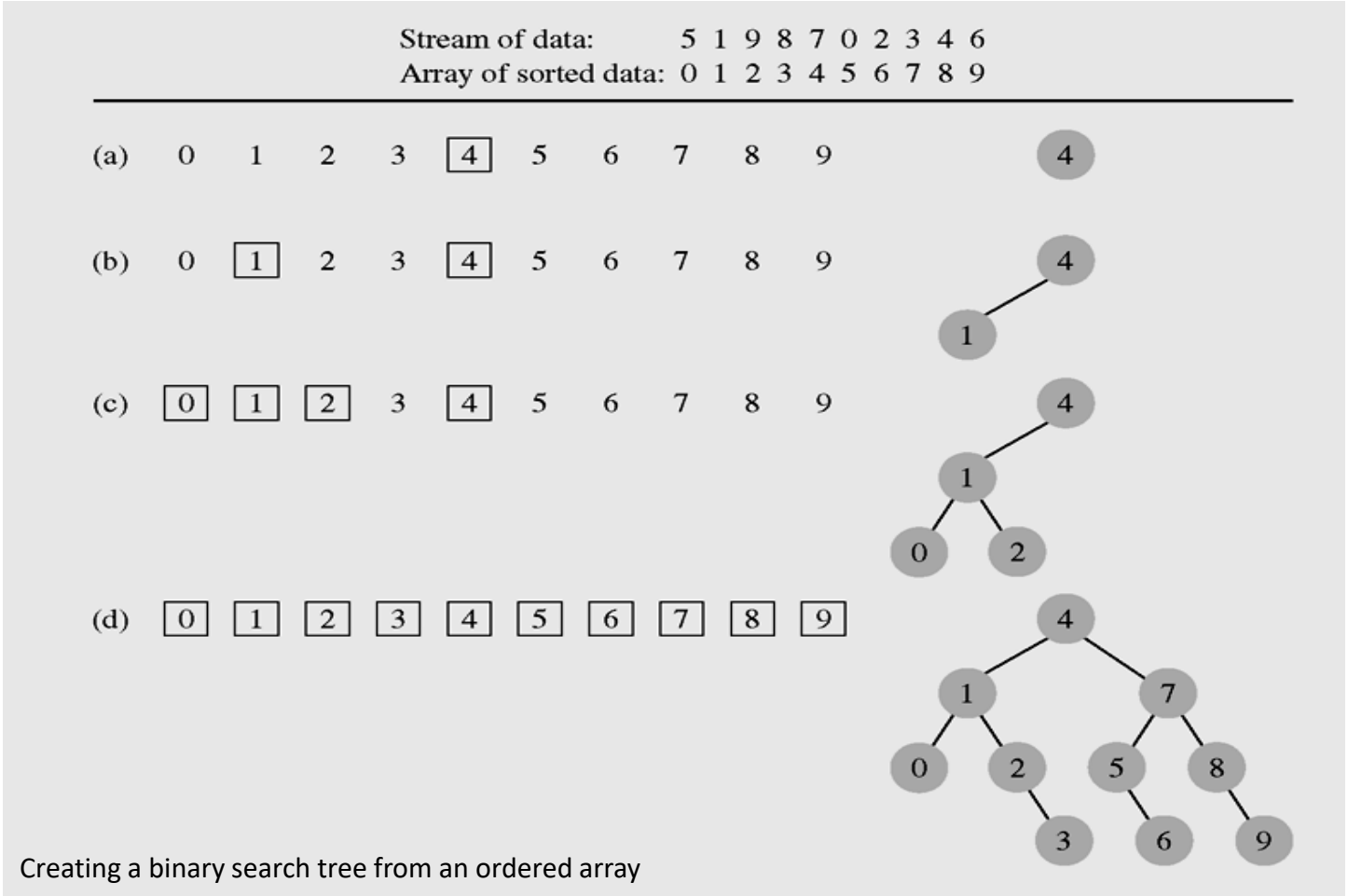
Giải thuật cân bằng đơn giản

- Đầu vào: mảng chứa n phần tử đã được sắp xếp: $a_1,a_2,\ldots,a_n$.
- Đầu ra: cây cân bằng T chứa n nút từ n phần tử của mảng đầu vào
- Giải thuật: (đệ quy)
  - Lấy phần tử ở giữa a_m của dãy rồi chèn vào cây T
  - Lặp lại giải thuật cho dãy đứng trước a_m: $a_1,a_2,\ldots,a_{m-1}$.
  - Lặp lại giải thuật cho dãy đứng sau a_m: $a_{m+1},a_{m+2},\ldots,a_n$.

Thủ tục

```
void balance(T data[], int first, int last) {
        if (first <= last) {
            int m = (first + last)/2;
            insert(data[m]);
            balance(data,first,m-1);
            balance(data,m+1,last);
        }
 }
```

Ví dụ*

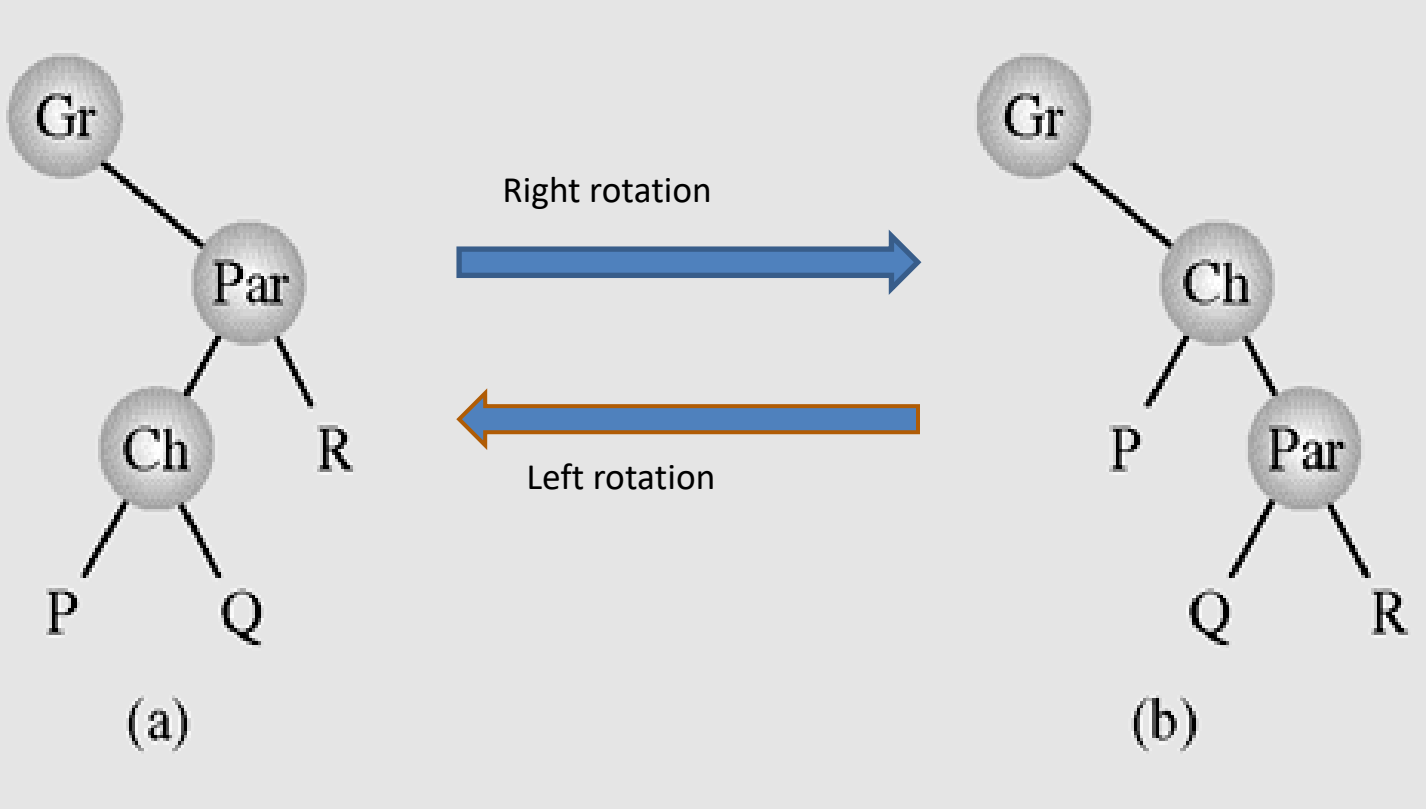


Creating a binary search tree from an ordered array

Giải thuật DSW

- Ý tưởng giải thuật:
  - Thao tác cơ bản trong giải thuật này là các phép quay trái và phải (left and right rotation)
  - Không yêu cầu phải có thao tác sắp xếp như giải thuật cân bằng đơn giản

Các phép quay



Gr: nút ông
Par: nút cha
Ch: nút con

Tác dụng của các phép quay

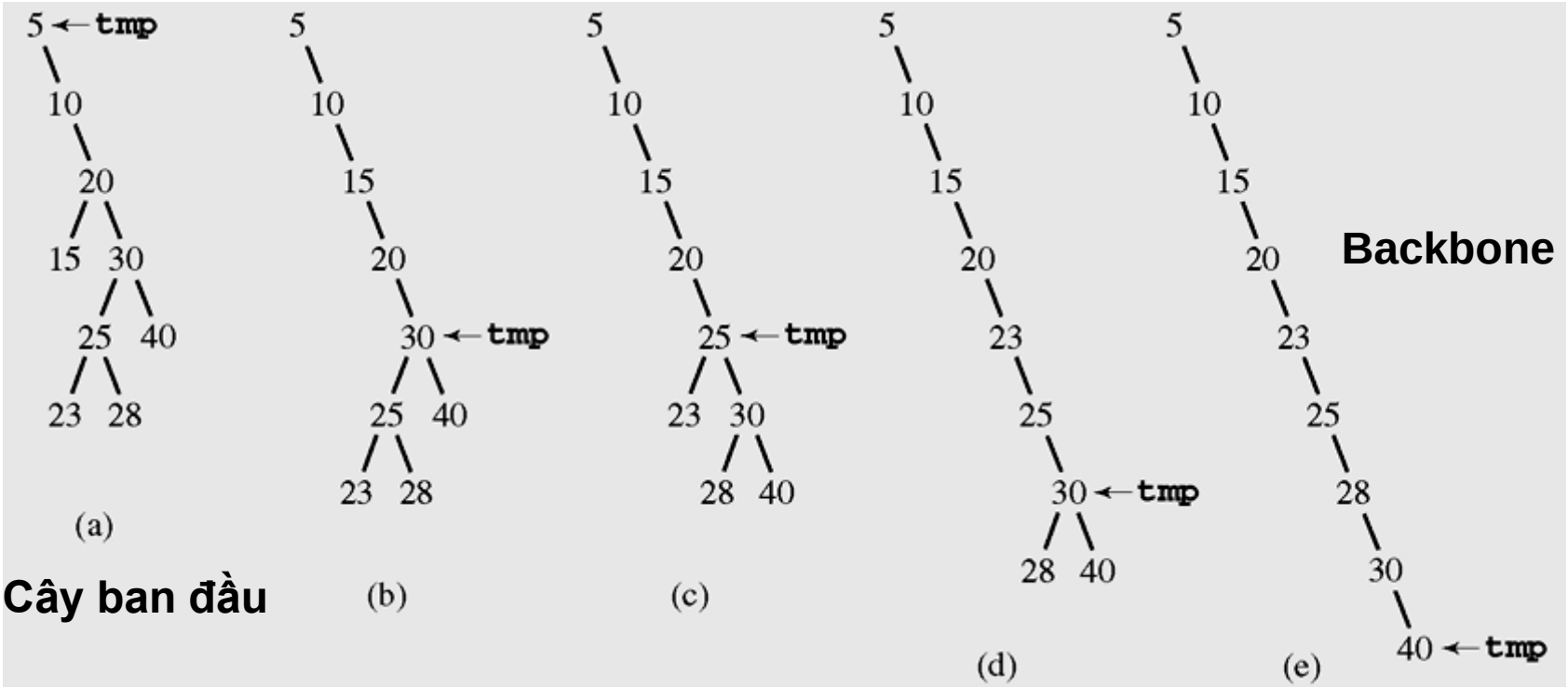
- Giúp thay đổi đồng thời chiều cao của các cây con:
  - Quay phải: tăng chiều cao con phải lên 1, giảm chiều cao con trái đi 1.
  - Quay trái: tăng chiều cao con trái lên 1, giảm chiều cao con phải đi 1.

➔ *Giúp cân bằng lại cây*

Giải thuật DSW

- Nội dung giải thuật: gồm 2 bước:
  1. Tạo ra một backbone từ cây ban đầu (CreateBackbone)
  2. Tạo ra cây cân bằng từ backbone (CreateBTree)

CreateBackbone – Khái niệm Backbone



CreateBackbone - Thủ tục

CreateBackbone(root, n)

temp = root;

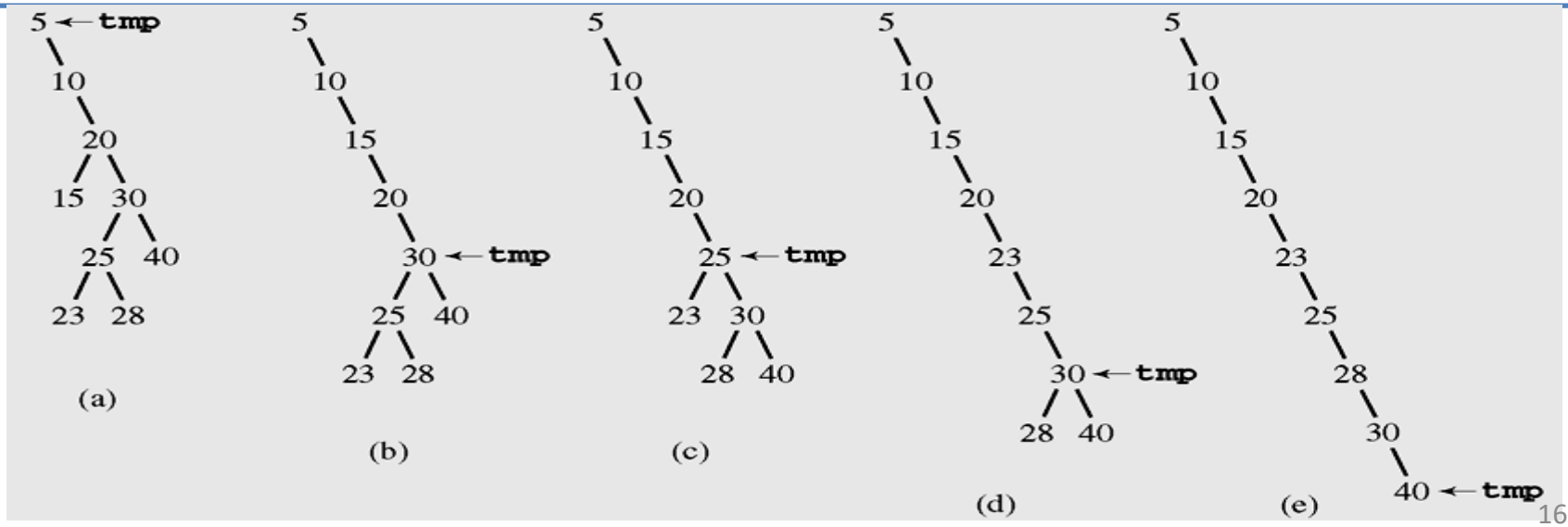
while (temp != null)

if (temp có một con trái)

Quay phải nút con trái này (temp trở thành con phải của nút đó);

temp = nút con trái này;

else temp = con phải của nó;



CreateBTree – Thủ tục

CreateBTree(n)

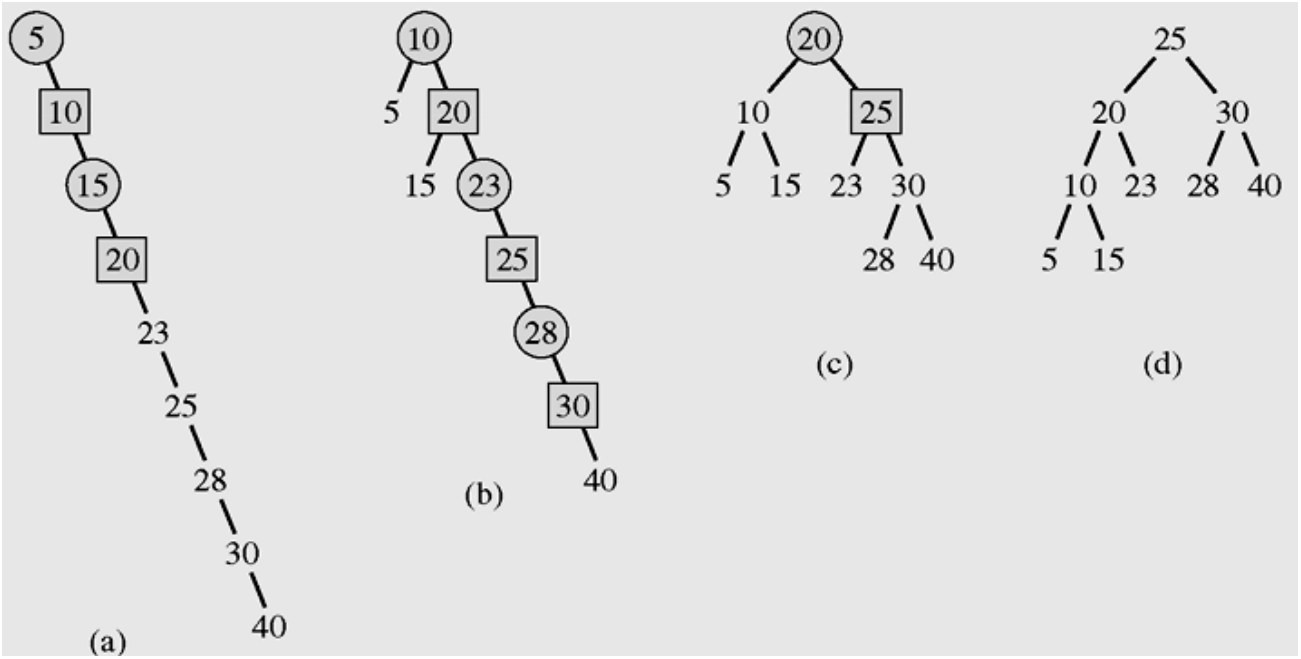
m = $2^{\lfloor \lg(n+1) \rfloor} - 1$;

Quay n-m lần, bắt đầu từ nút đỉnh của *backbone*;

while (m>1)

m = m / 2;

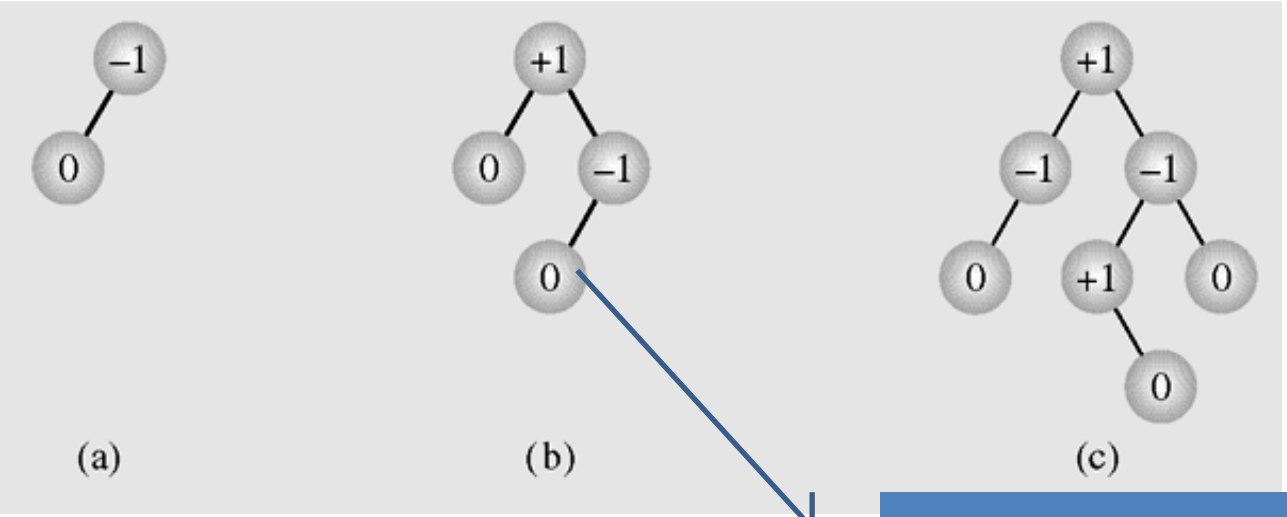
Quay m lần, bắt đầu từ nút đỉnh của *backbone*;



n=9;
m=7;

Cây AVL

- Cây **AVL** *(by Adelson, Velskii, Landis)* là cây cân bằng có độ lệch chiều cao giữa 2 cây con lớn nhất là 1.



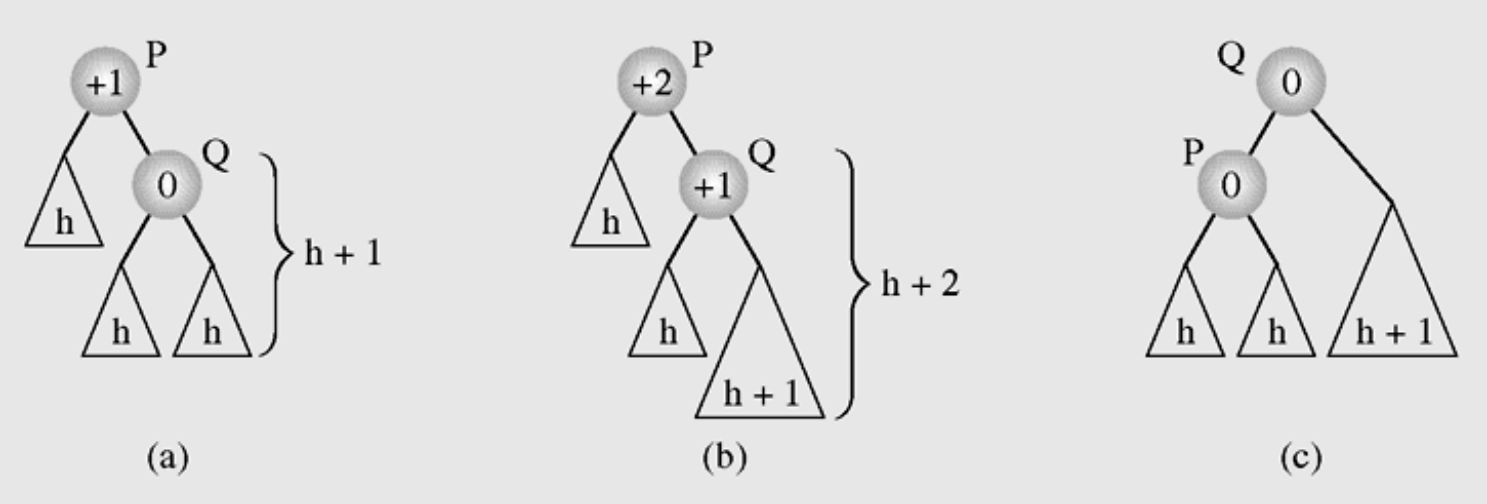
Examples of AVL trees

Balance factor = height(right)-height(left)

Cây AVL

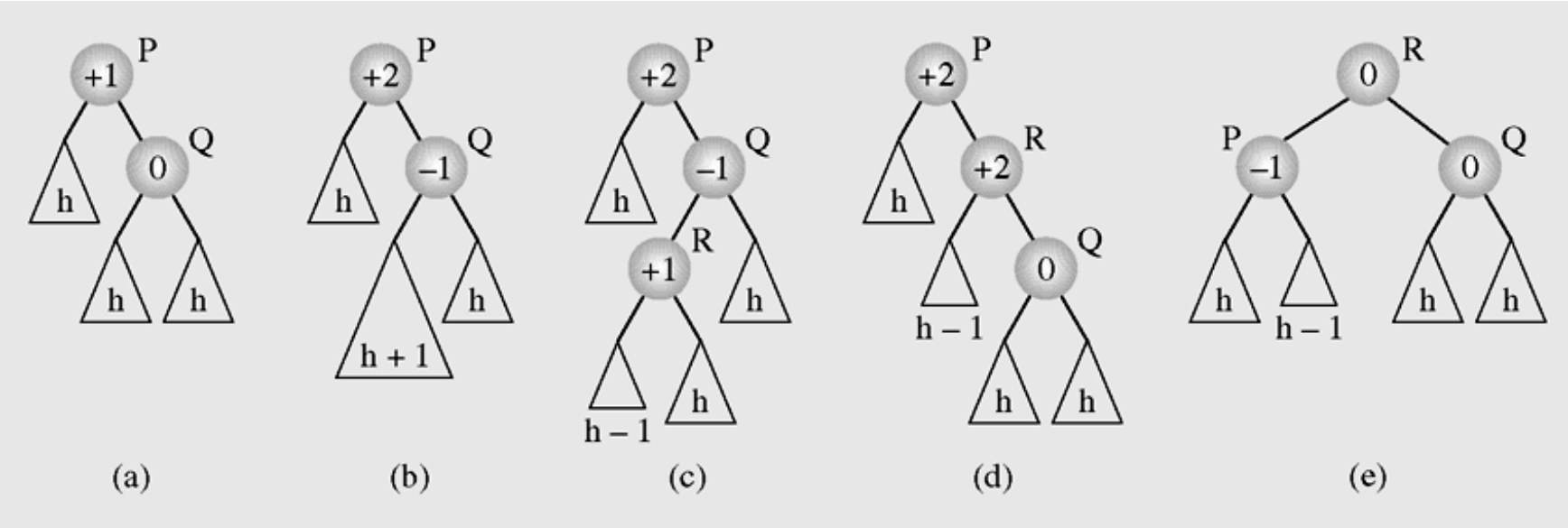
- **Chiến lược cân bằng**:
  - Trạng thái cân bằng của tất cả các nút trong cây luôn được cập nhật: để kịp thời biết được vị trí mất cân bằng trong cây.
  - Cân bằng cục bộ: chỉ cân bằng lại những chỗ mất cân bằng ngay khi phát hiện ra (khi bổ sung hoặc loại bỏ một nút).

Bổ sung 1 nút vào cây AVL - 1



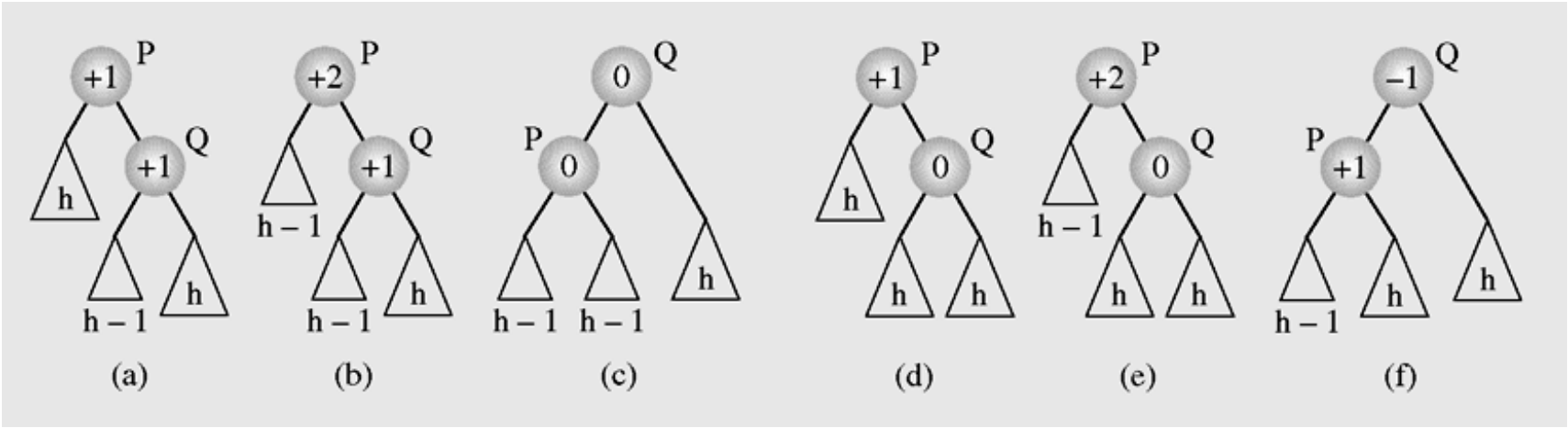
Cân bằng lại cây sau khi bổ sung một nút vào cây con phải của Q

Bổ sung 1 nút vào cây AVL - 2



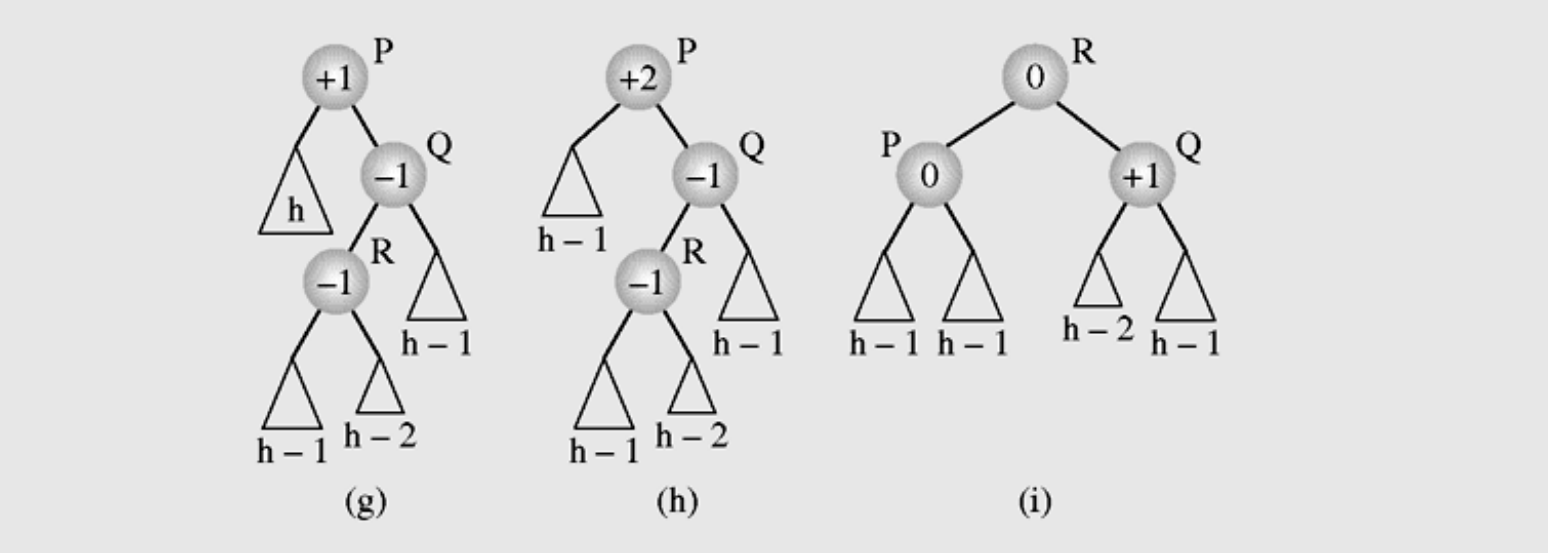
Cân bằng lại cây sau khi bổ sung một nút vào cây con trái của Q

Loại bỏ 1 nút khỏi cây AVL - 1



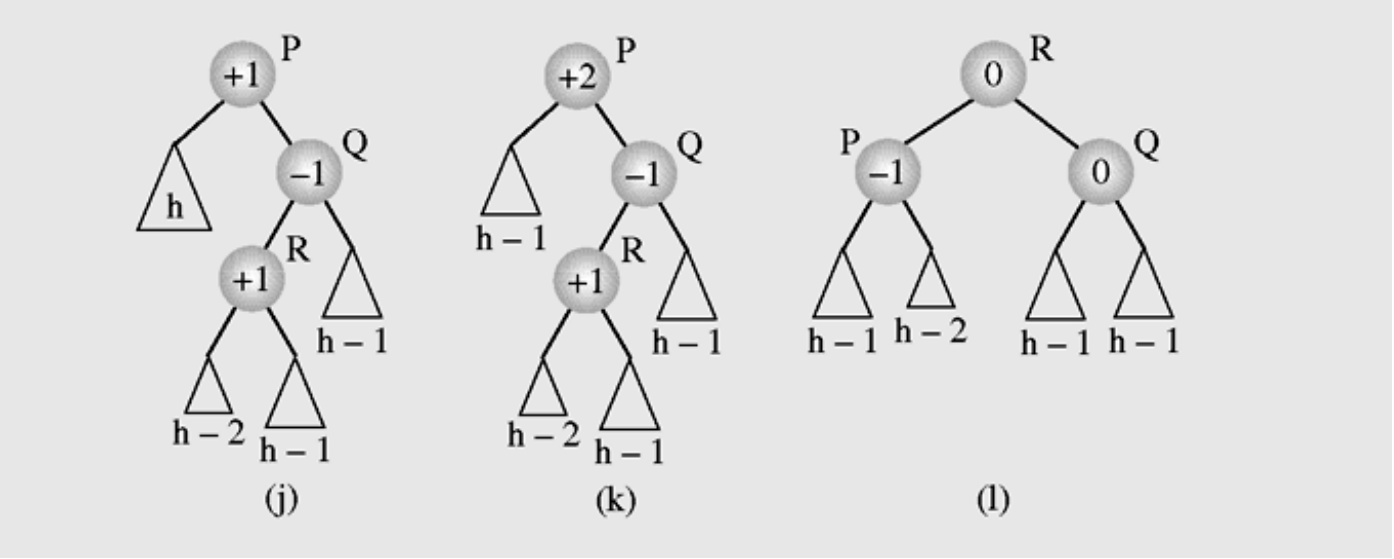
Cân bằng lại cây sau khi loại bỏ một nút ở cây con trái của P

Loại bỏ 1 nút khỏi cây AVL - 2



Cân bằng lại cây sau khi loại bỏ một nút ở cây con trái của P (tiếp)

Loại bỏ 1 nút khỏi cây AVL - 3



Cân bằng lại cây sau khi loại bỏ một nút ở cây con trái của P (tiếp)