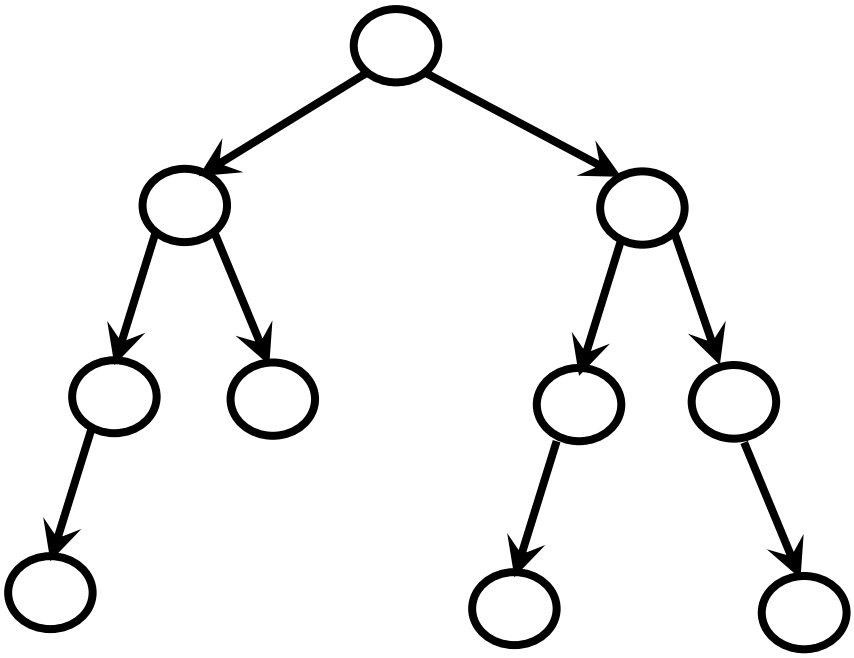
Chương 7
CẤU TRÚC DỮ LIỆU
CÂY NHỊ PHÂN CÂN BẰNG

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

1. HÌNH ẢNH CÂY NHỊ PHÂN CÂN BẰNG



TS. Nguyễn Tấn Trần Minh Khang

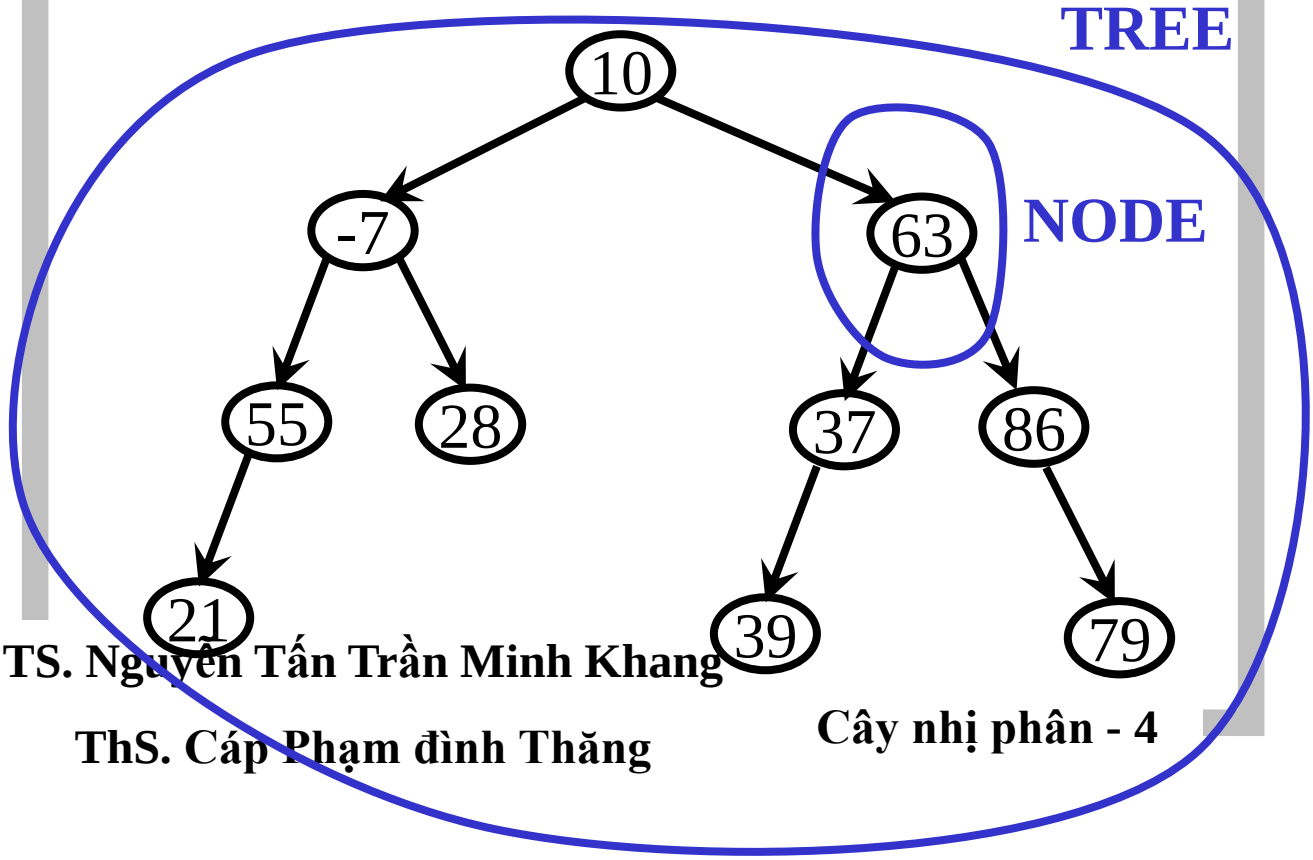
ThS. Cáp Phạm đình Thăng

2. KHÁI NIỆM CÂY NHỊ PHÂN CÂN BẰNG

- Cây nhị phân cân bằng là một cây nhị phân tìm kiếm thoả điều kiện: mọi node trong cây có độ lệch tối đa là 1.

3. CẤU TRÚC DỮ LIỆU CÂY NHỊ PHÂN CÂN BẰNG

```
1. struct node
2. {
3.     KDL info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE *TREE;
```



3. CẤU TRÚC DỮ LIỆU CÂY NHỊ PHÂN CÂN BẰNG

- Ví dụ 1: Hãy khai báo cấu trúc dữ liệu cho cây nhị phân cân bằng các số nguyên.
- Cấu trúc dữ liệu.

```
1. struct node
2. {
3.     int info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE *TREE;
```

3. CẤU TRÚC DỮ LIỆU CÂY NHỊ PHÂN CÂN BẰNG

- Ví dụ 2: Hãy khai báo cấu trúc dữ liệu cho cây nhị phân cân bằng các phân số.
- Cấu trúc dữ liệu.

```
10. struct phanso
11. {
12.     int tu;
13.     int mau;
14. };
15. typedef struct phanso PHANSO;
16. struct node
17. {
18.     PHANSO info;
19.     struct node *pLeft;
20.     struct node *pRight;
21. };
22. typedef struct node NODE;
23. typedef NODE *TREE;
```

4. KHỞI TẠO CÂY NHỊ PHÂN CÂN BẰNG

- Khái niệm: Khởi tạo cây nhị phân là tạo ra một cây nhị phân rỗng không chứa node nào hết.
- Định nghĩa hàm

```
1. void Init(TREE &t)
2. {
3.     t = NULL;
4. }
```

5. TẠO NODE CHO CÂY NHỊ PHÂN CÂN BẰNG

- Khái niệm: Tạo node cho cây nhị phân là xin cấp phát bộ nhớ có kích thước bằng kích thước của KDL NODE để chứa thông tin biết trước.
- **Định nghĩa hàm**

```
10. NODE* GetNode(KDL x)
11. {
12.     NODE *p = new NODE;
13.     if(p==NULL)
14.         return NULL;
15.     p->info = x;
16.     p->pLeft = NULL;
17.     p->pRight= NULL;
18.     return p;
19. }
```

5. TẠO NODE CHO CÂY NHỊ PHÂN CÂN BẰNG

- Ví dụ: Định nghĩa hàm tạo node cho cây nhị phân các số nguyên.
- **Định nghĩa hàm**

```
10. NODE* GetNode(int x)
11. {
12.     NODE *p = new NODE;
13.     if(p==NULL)
14.         return NULL;
15.     p->info = x;
16.     p->pLeft = NULL;
17.     p->pRight= NULL;
18.     return p;
19. }
```

6. THÊM MỘT NODE VÀO CÂY NHỊ PHÂN CÂN BẰNG

- Thêm một nút vào trong cây nhị phân cân bằng có thể làm cây mất cân bằng.

7. XÓA MỘT NODE TRONG CÂY NHỊ PHÂN CÂN BẰNG

- Xóa một nút ra khỏi cây nhị phân cân bằng có thể làm cây mất cân bằng.

8. THAO TÁC CÂN BẰNG LẠI

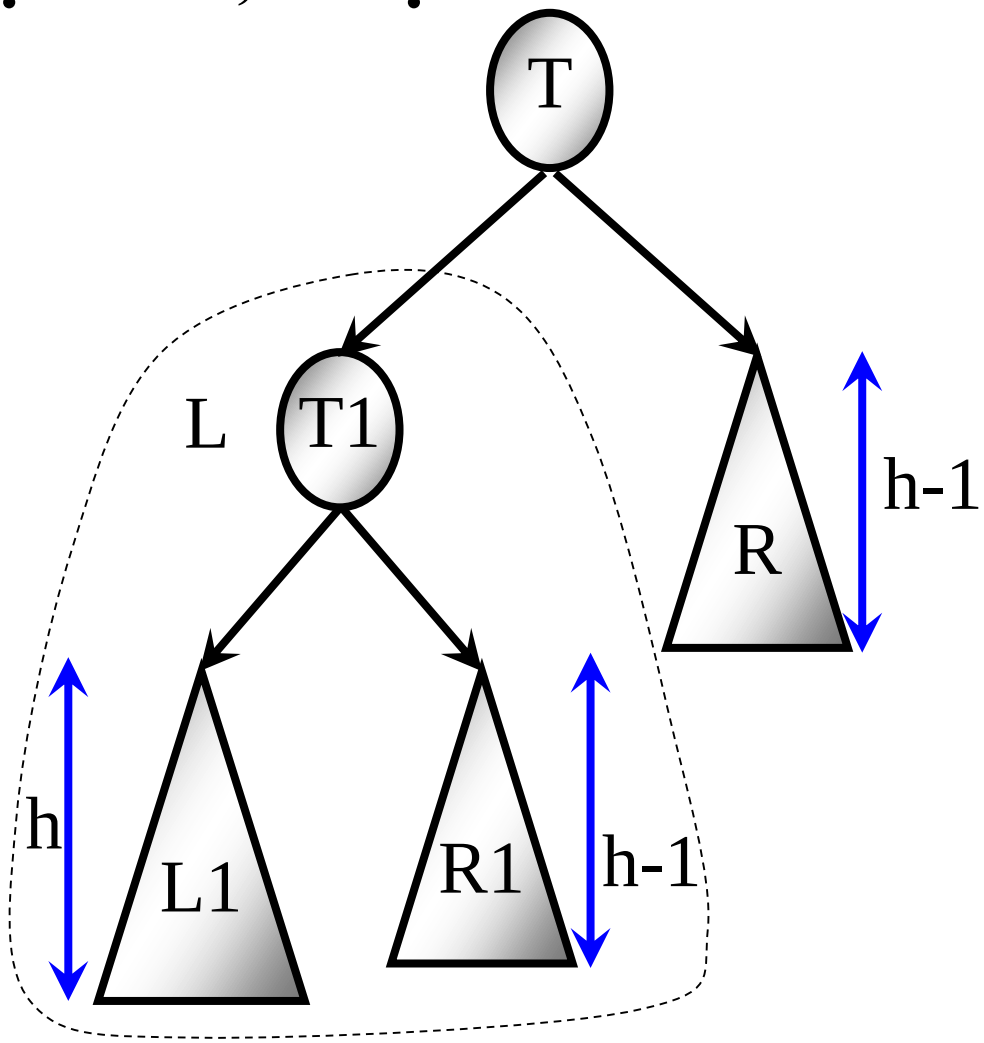
TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

T lệch trái
T1 lệch trái

8.1 T LỆCH TRÁI

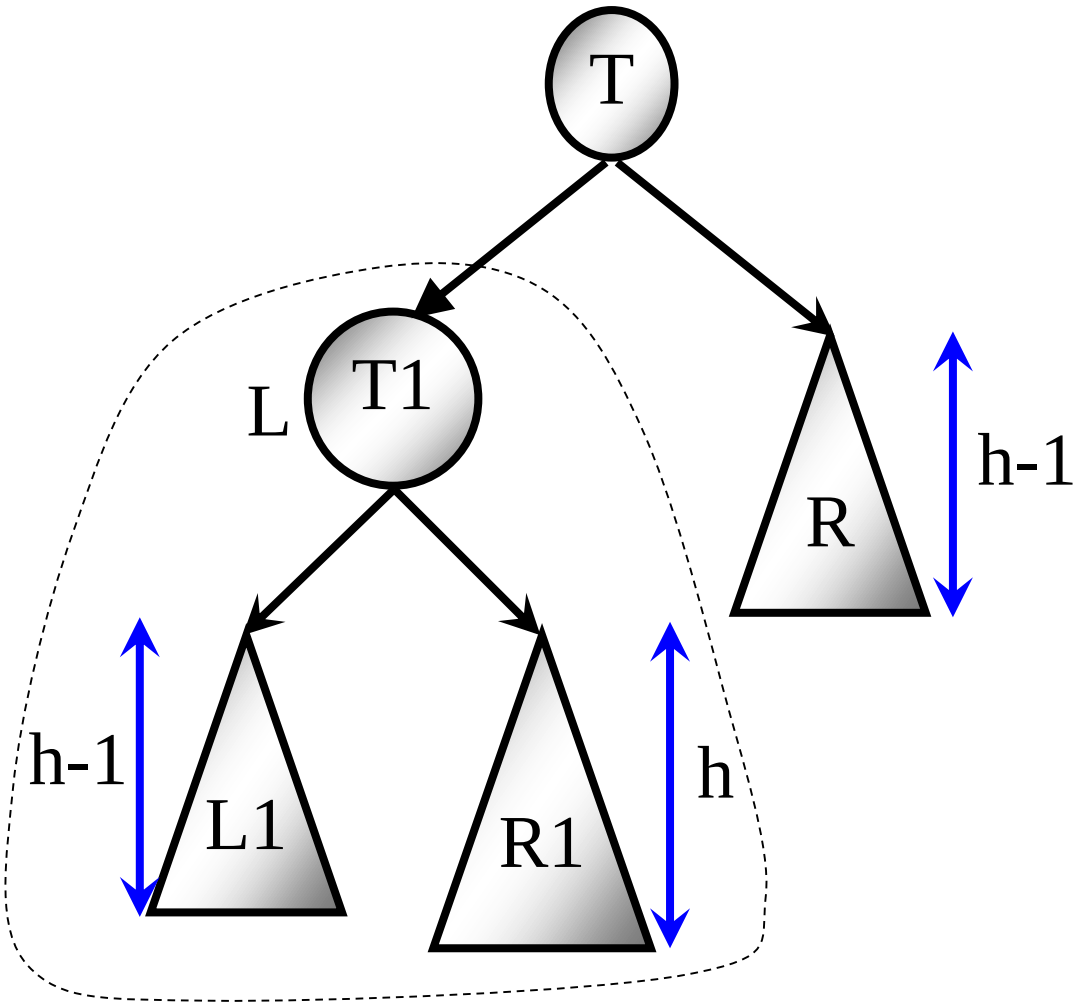
T lệch trái, T1 lệch trái



T lệch trái
T1 lệch phải

8.1 T LỆCH TRÁI

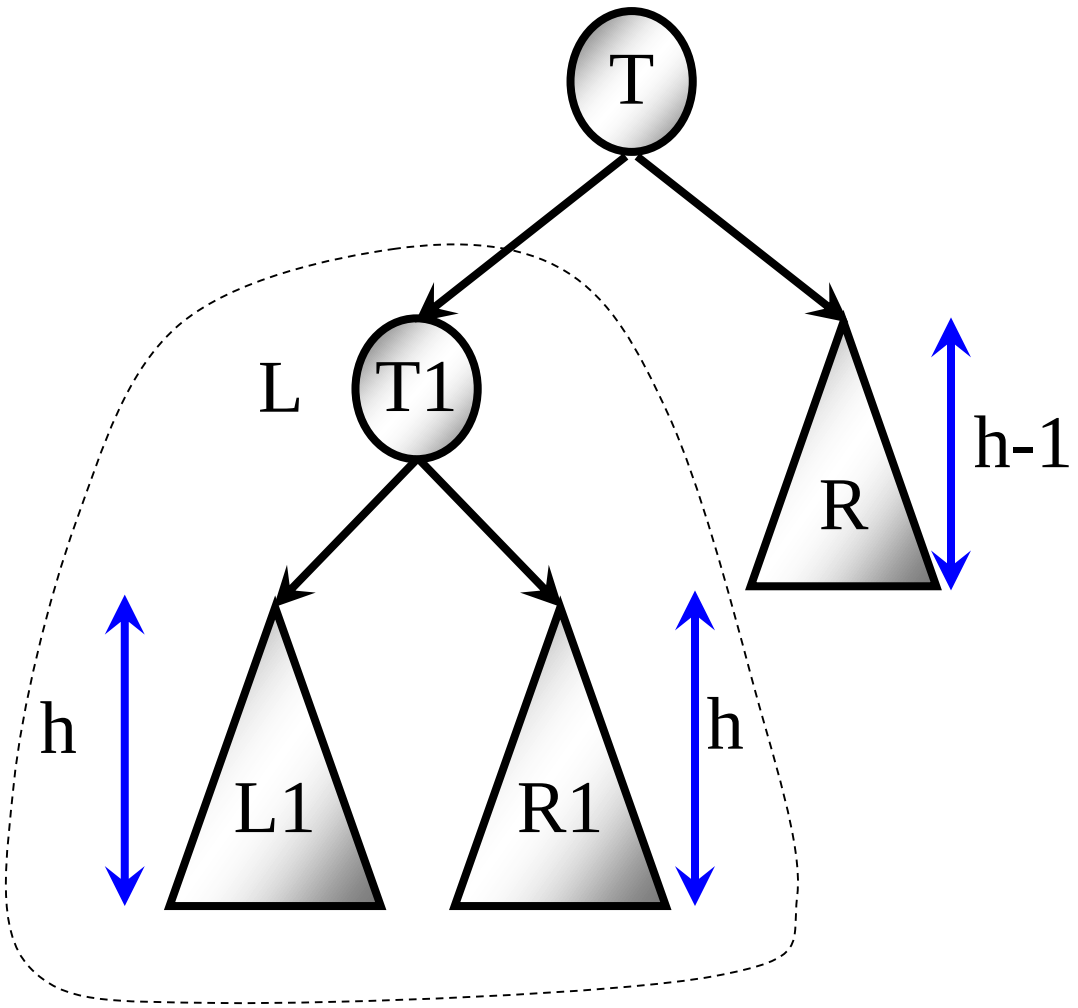
T lệch trái, T1 lệch phải



T lệch trái
T1 không lệch

8.1 T LỆCH TRÁI

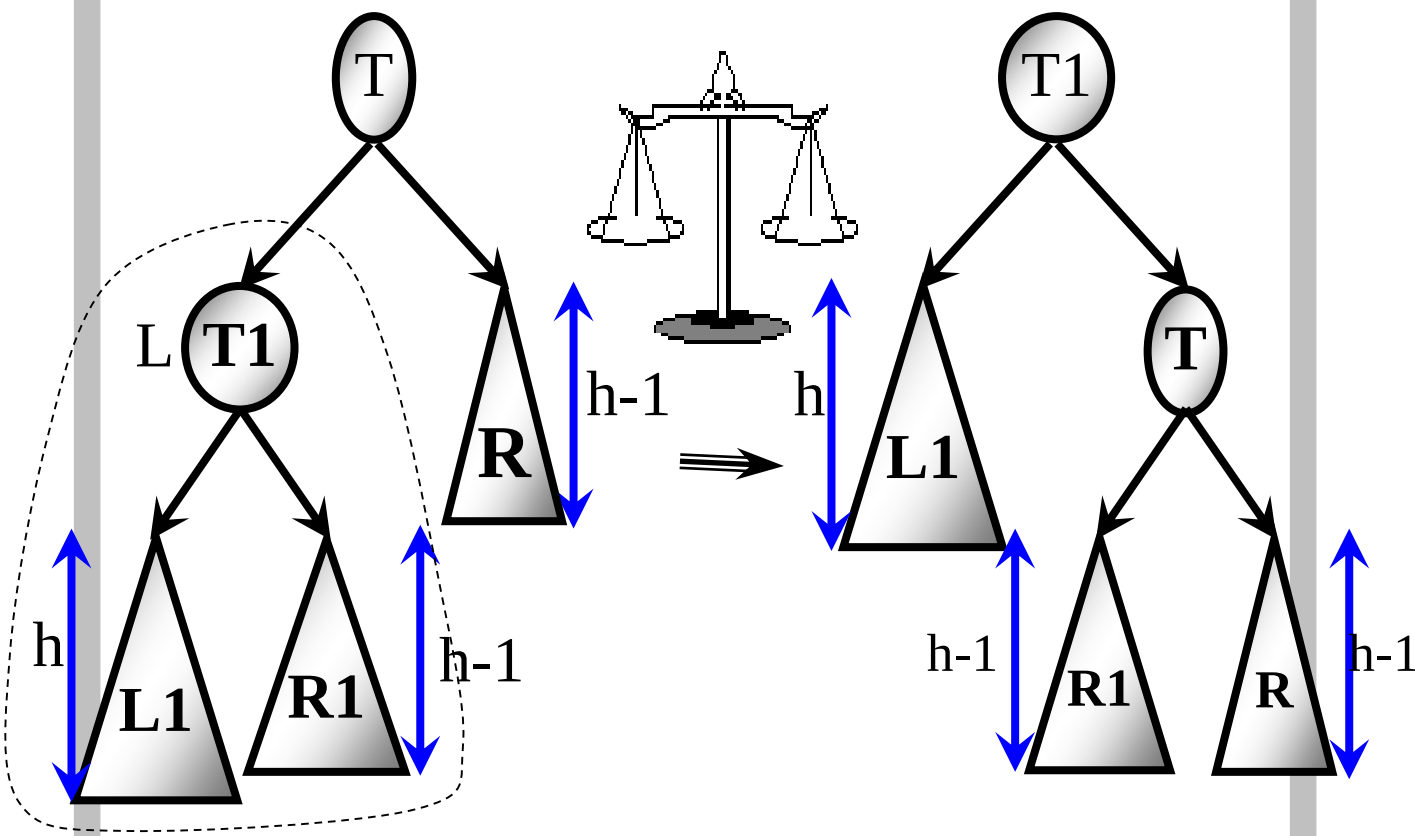
T lệch trái, T1 không lệch



T lệch trái
T1 lệch trái

8.1 T LỆCH TRÁI

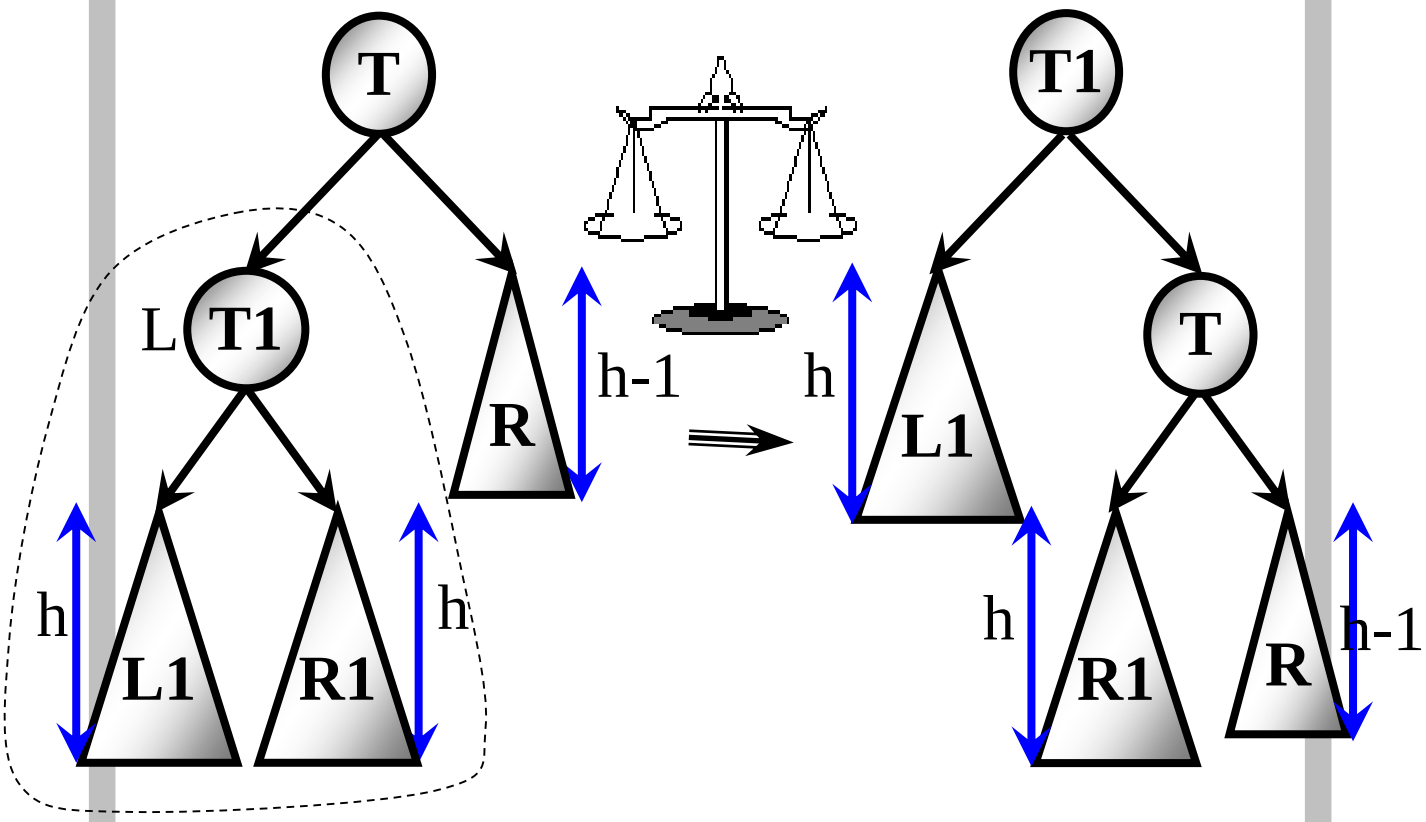
Cân bằng lại: T lệch trái, T1 lệch trái



T lệch trái
T1 không lệch

8.1 T LỆCH TRÁI

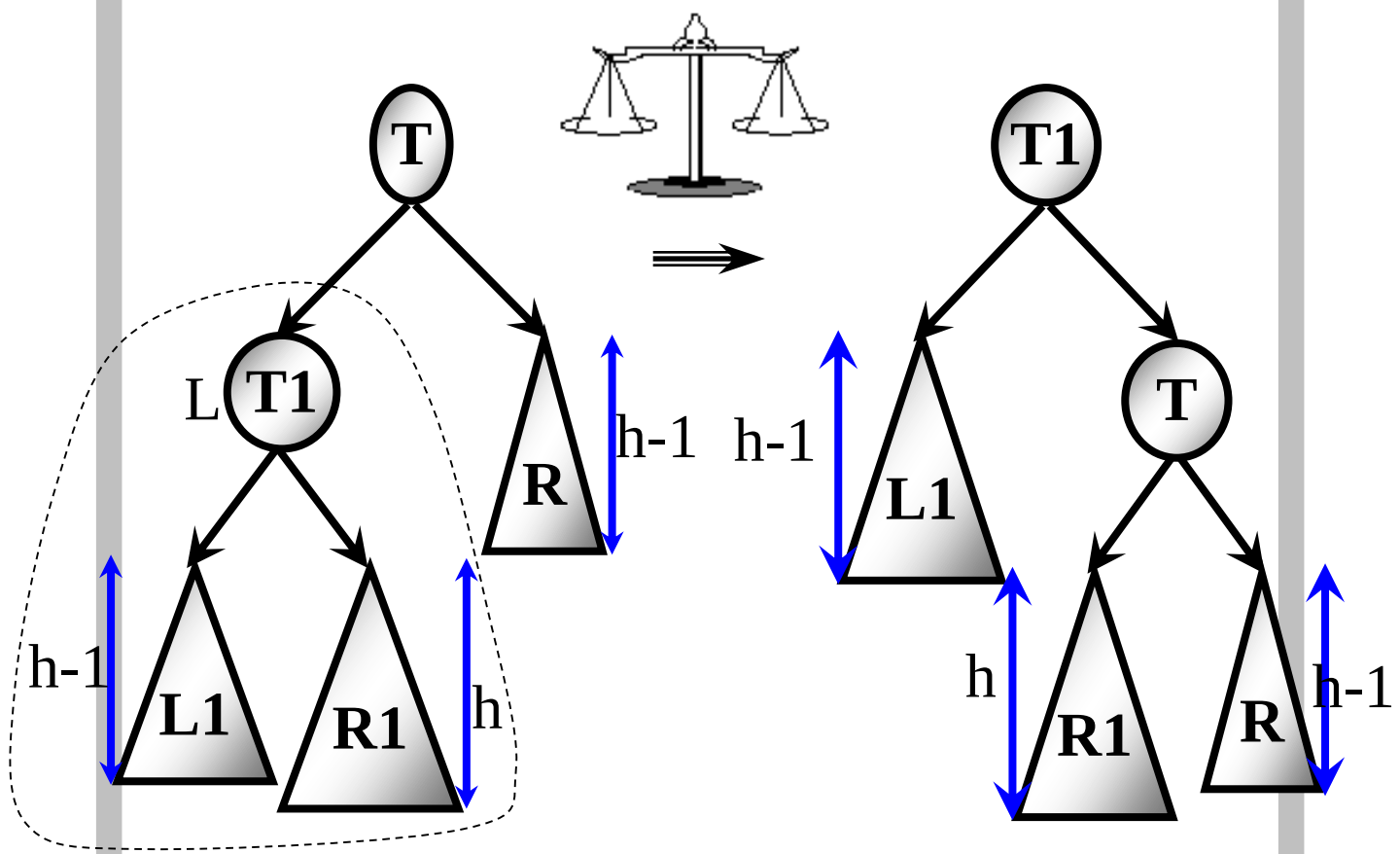
Cân bằng lại: T lệch trái, T1 không lệch



T lệch trái
T1 lệch phải

8.1 T LỆCH TRÁI

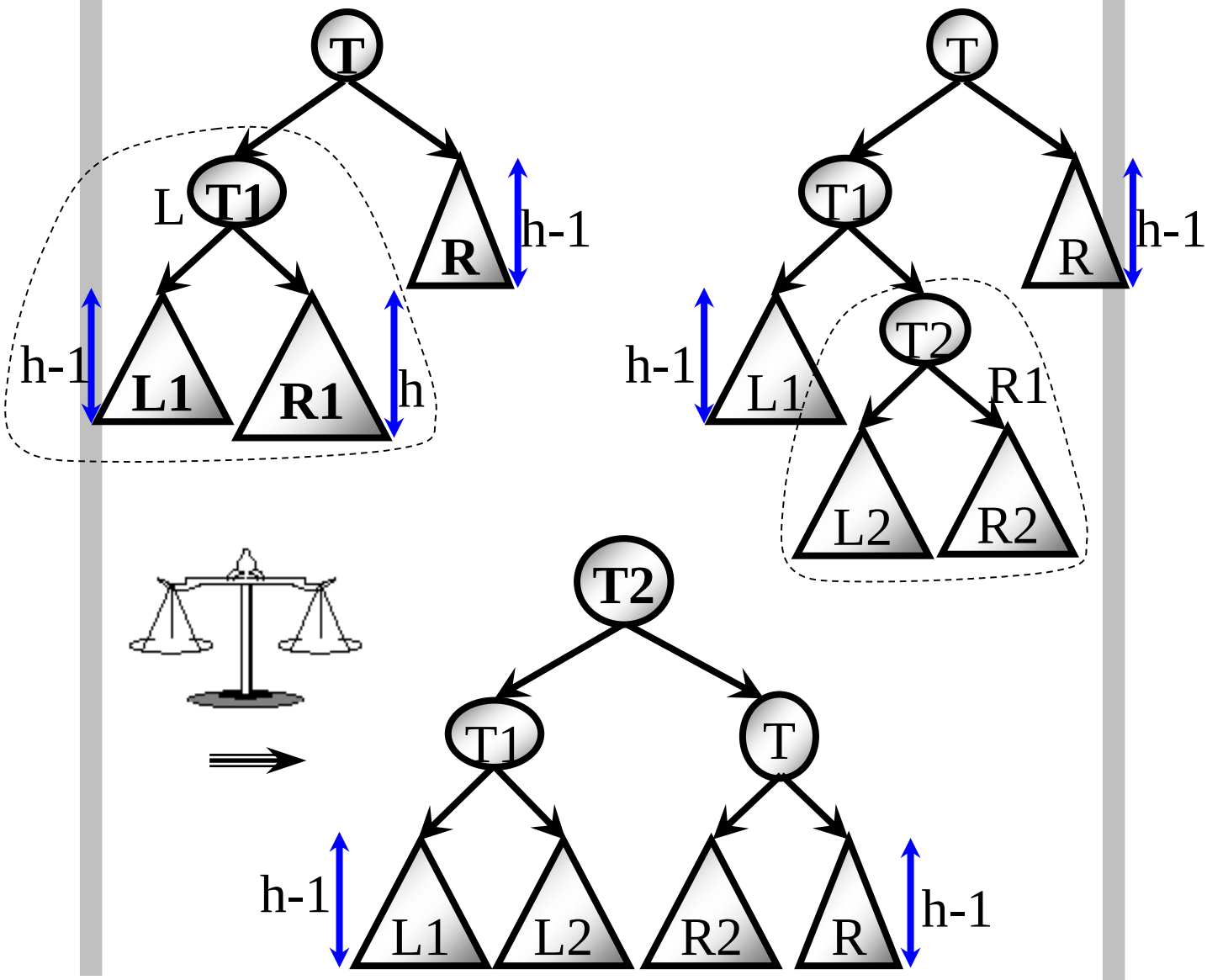
Cân bằng lại: T lệch trái, T1 lệch phải



T lệch trái
T1 lệch phải

8.1 T LỆCH TRÁI

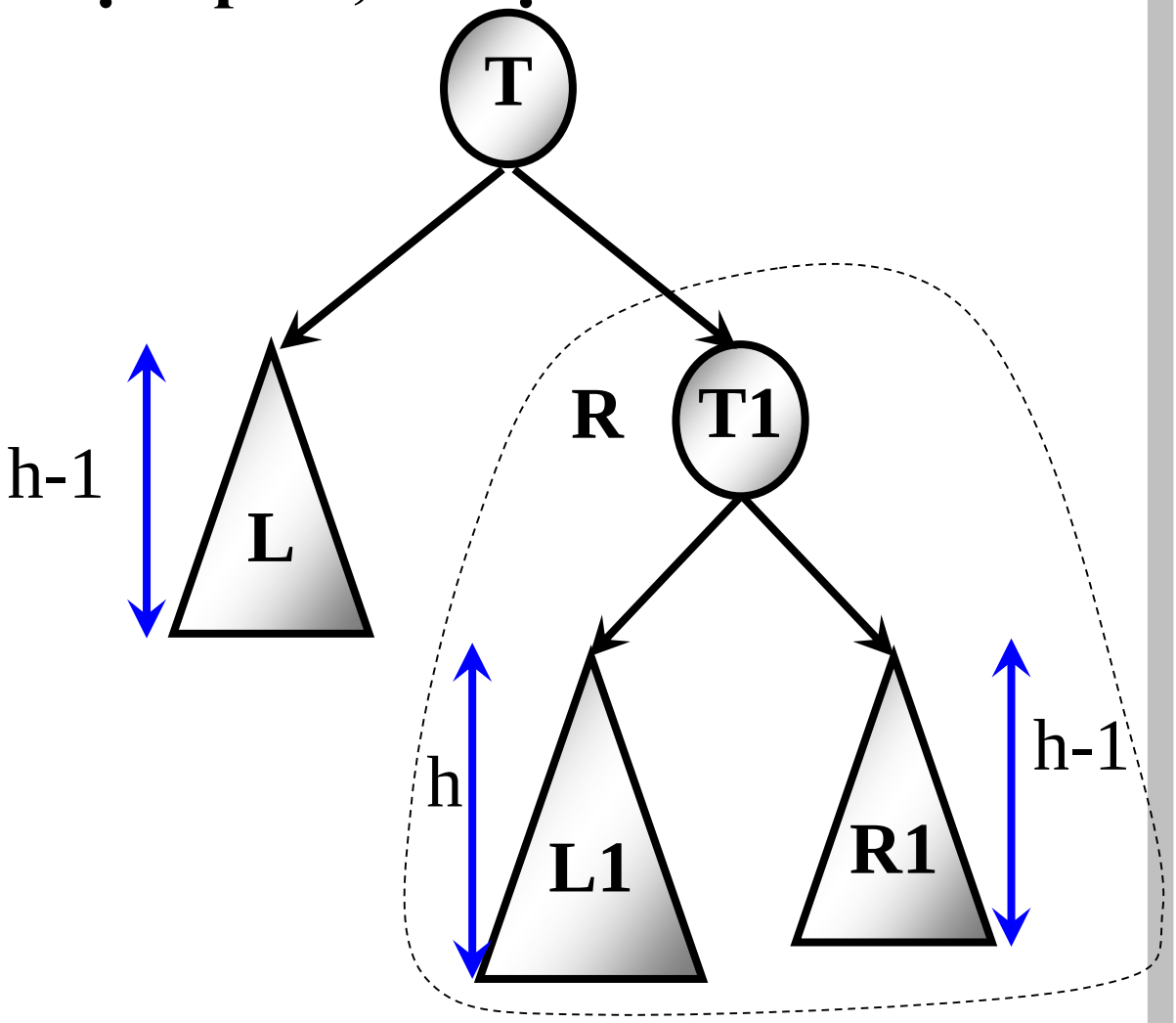
Cân bằng lại: T lệch trái, T1 lệch phải



T lệch phải
T1 lệch trái

8.2 T LỆCH PHẢI

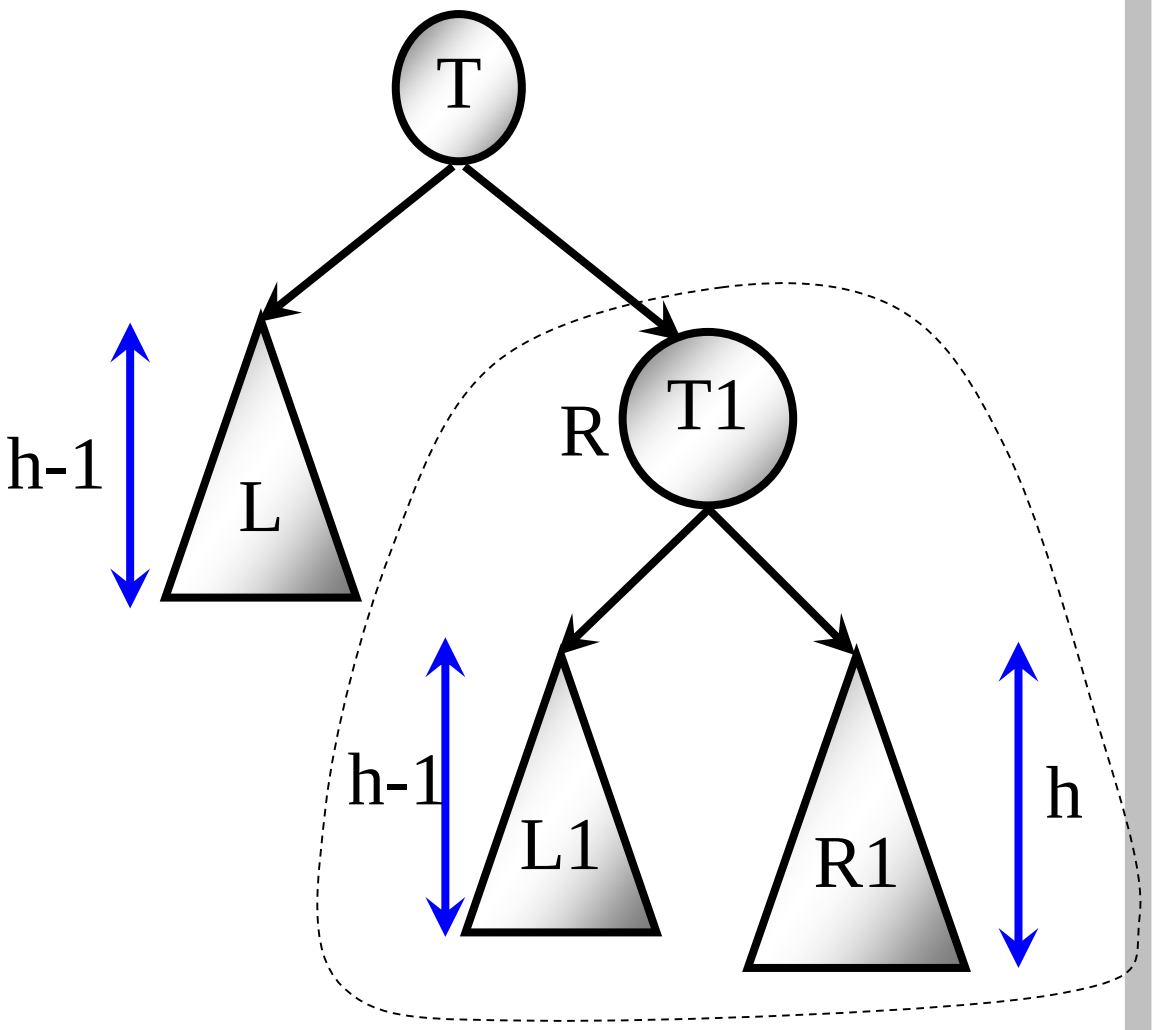
T lệch phải, T1 lệch trái



T lệch phải
T1 lệch phải

8.2 T LỆCH PHẢI

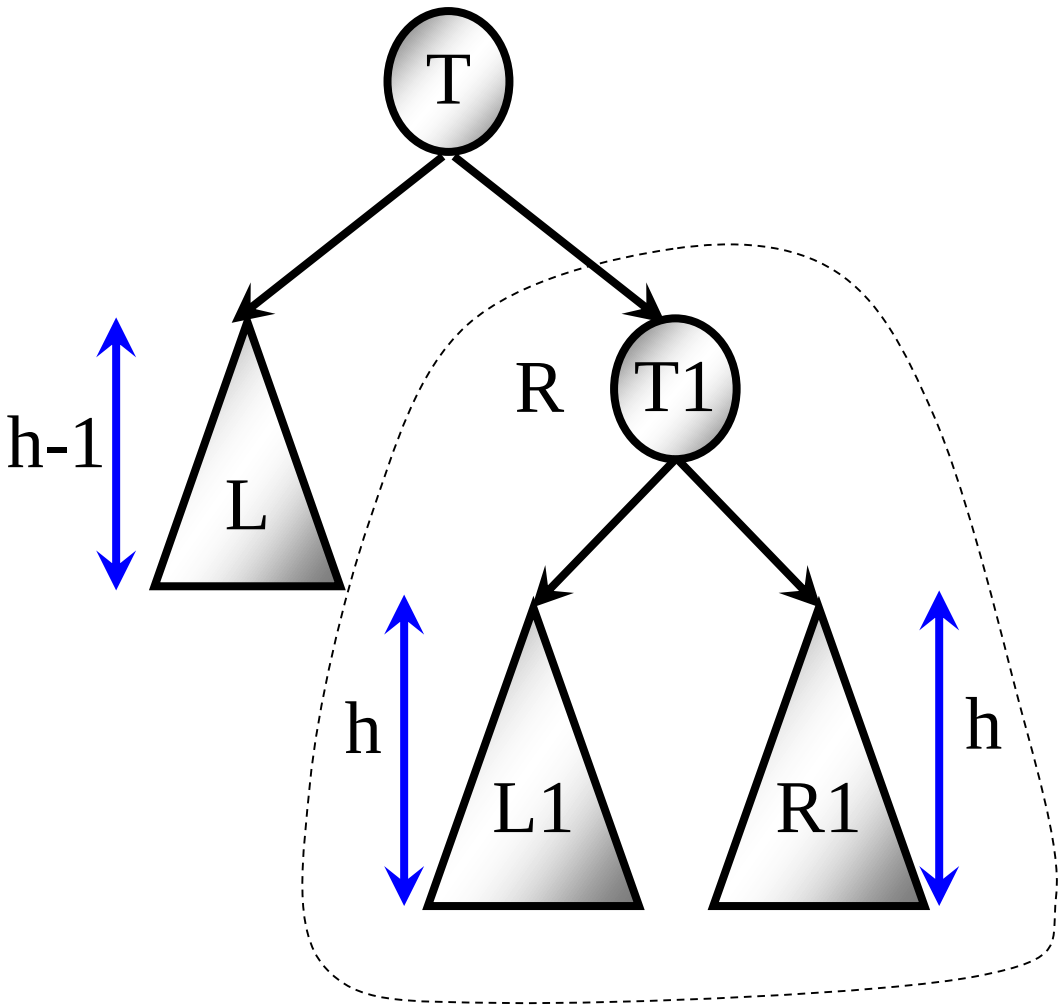
T lệch phải, T1 lệch phải



T lệch phải
T1 không lệch

8.2 T LỆCH PHẢI

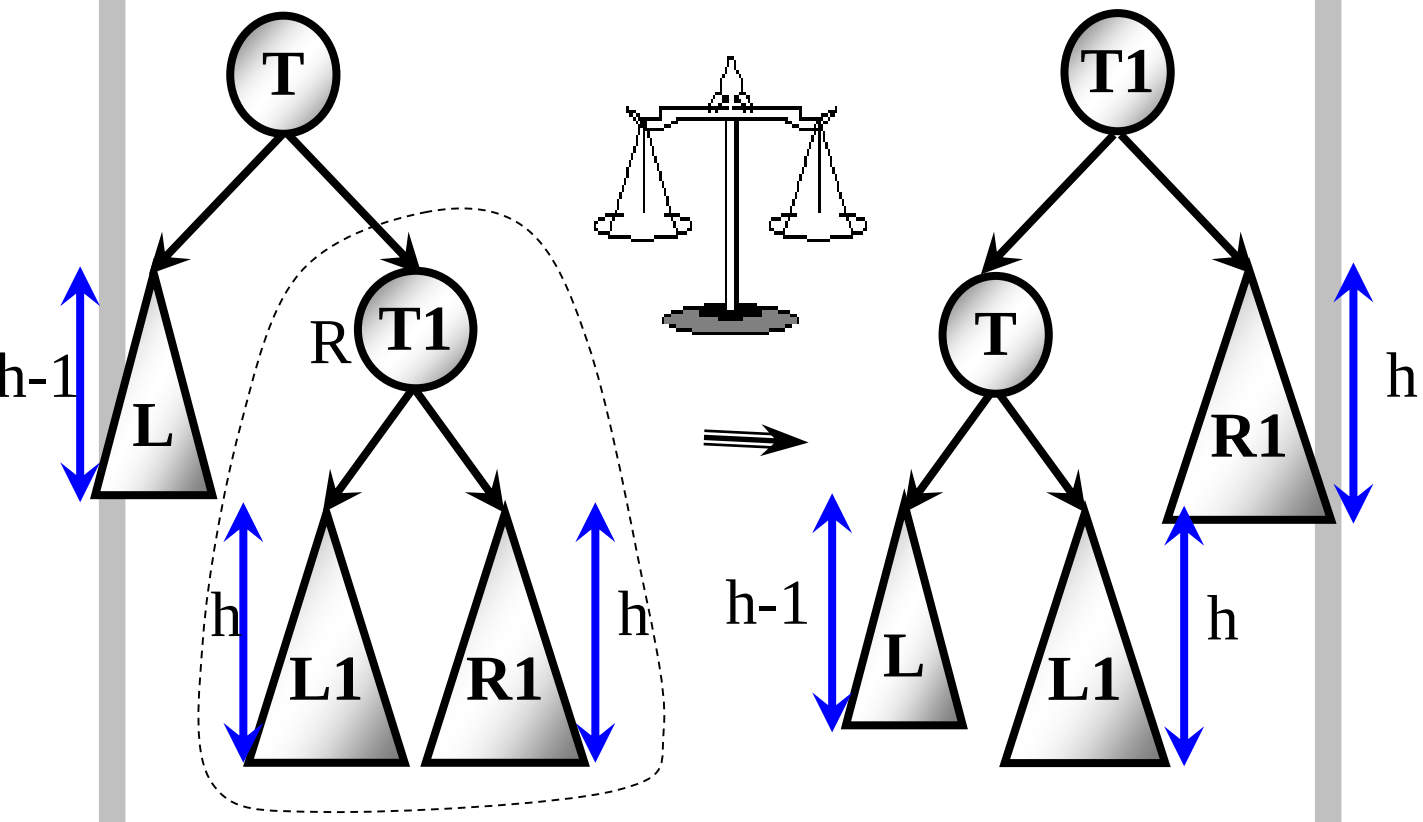
T lệch phải, T1 không lệch



T lệch phải
T1 không lệch

8.2 T LỆCH PHẢI

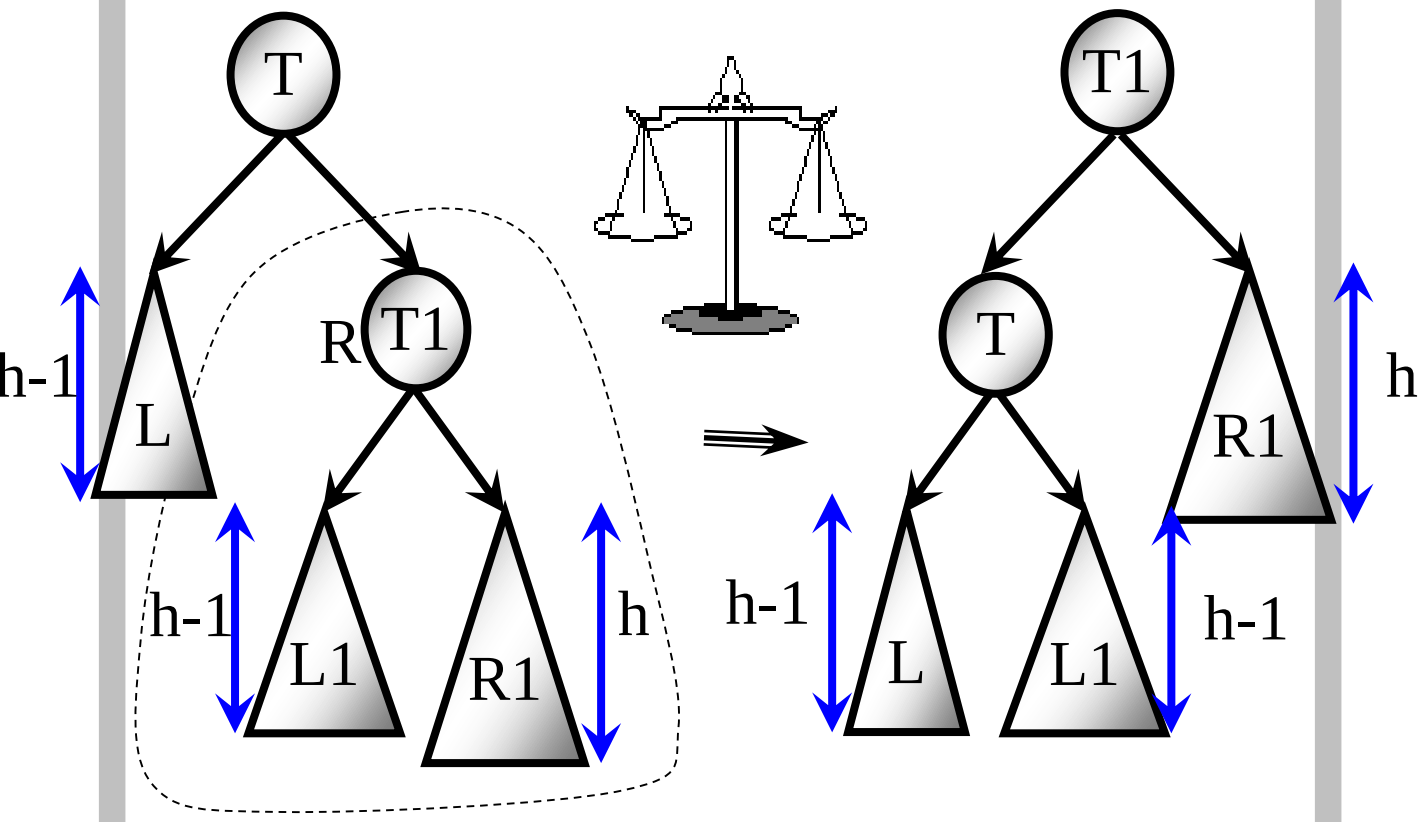
Cân bằng lại: T lệch phải, T1 không lệch



T lệch phải
T1 lệch phải

8.2 T LỆCH PHẢI

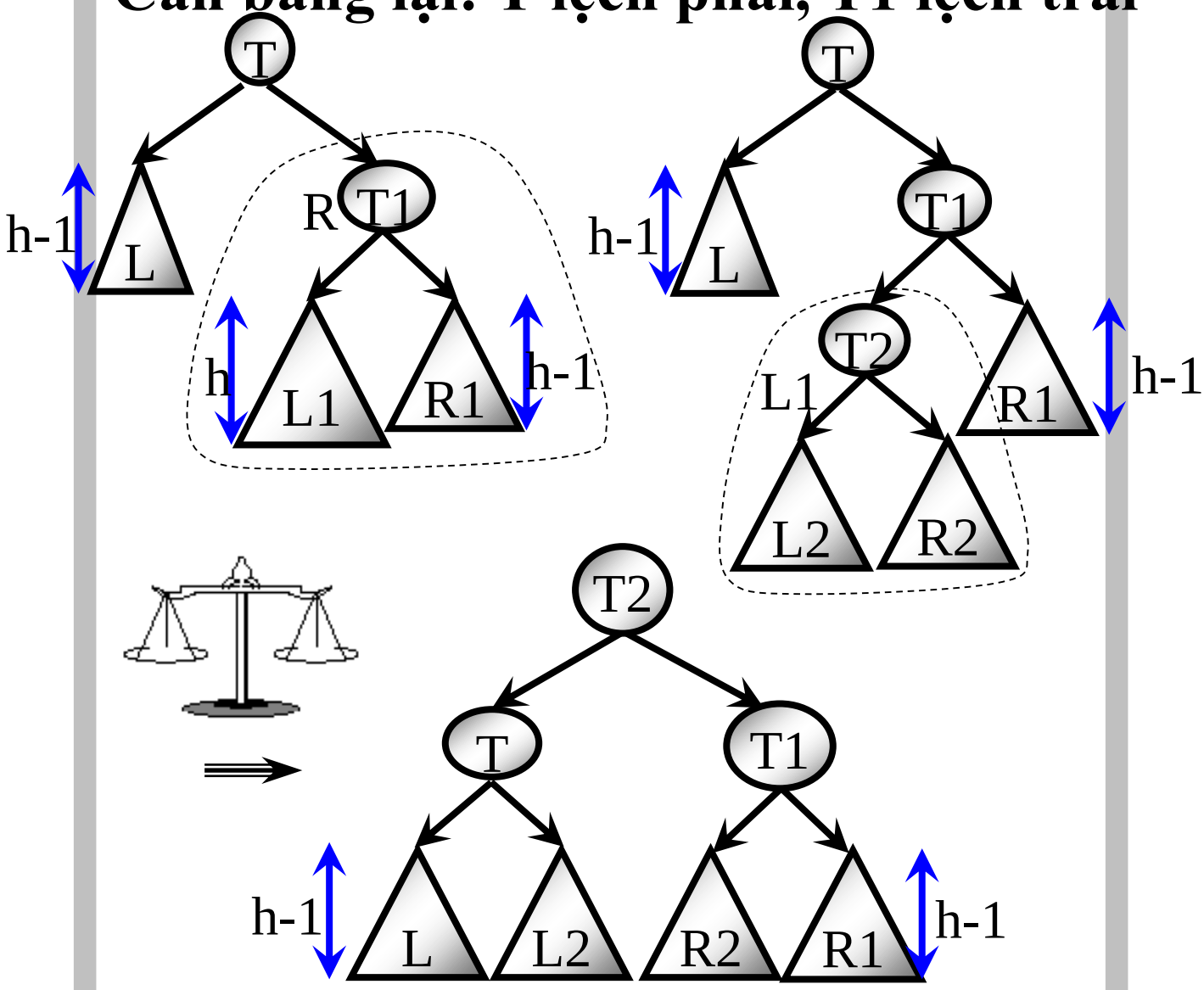
Cân bằng lại: T lệch phải, T1 lệch phải



T lệch phải
T1 lệch trái

8.2 T LỆCH PHẢI

Cân bằng lại: T lệch phải, T1 lệch trái



9. ÁP DỤNG

- Hãy vẽ cây cân bằng nếu chúng ta tiến hành tạo cây với dữ liệu các nút thêm vào theo thứ tự sau (giả sử cây ban đầu rỗng):
- M, N, O, F, C, G, H, U, A, E

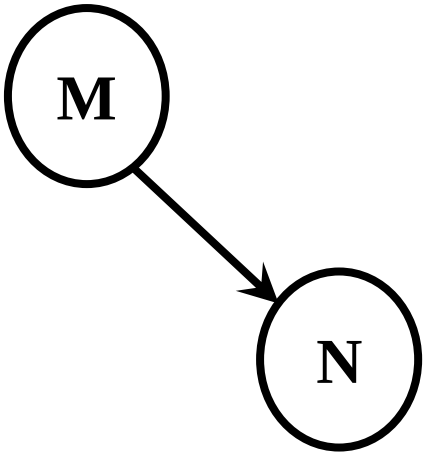
9. ÁP DỤNG

- Thêm M

- Thêm N

9. ÁP DỤNG

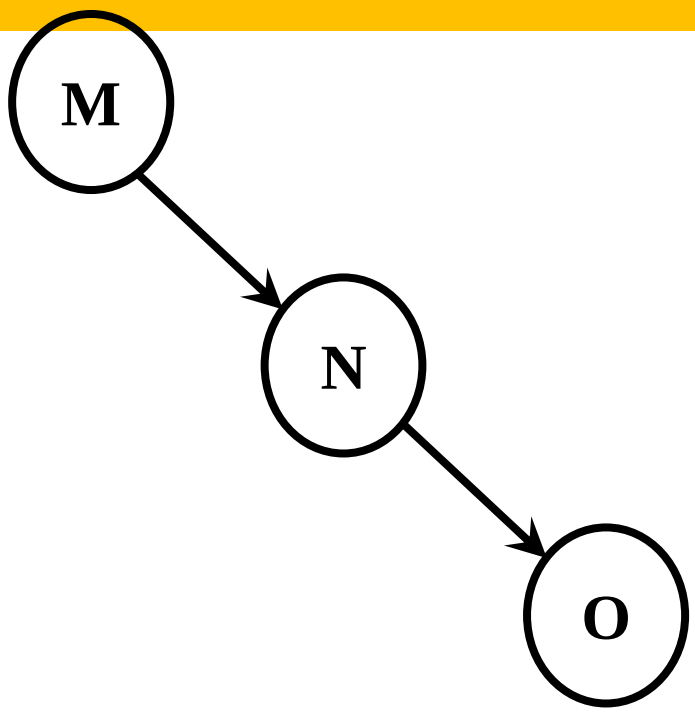
- Thêm N



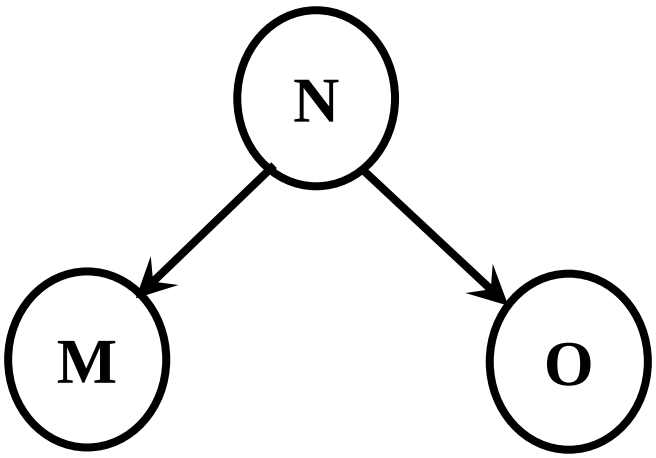
- Thêm O

9. ÁP DỤNG

- Thêm O



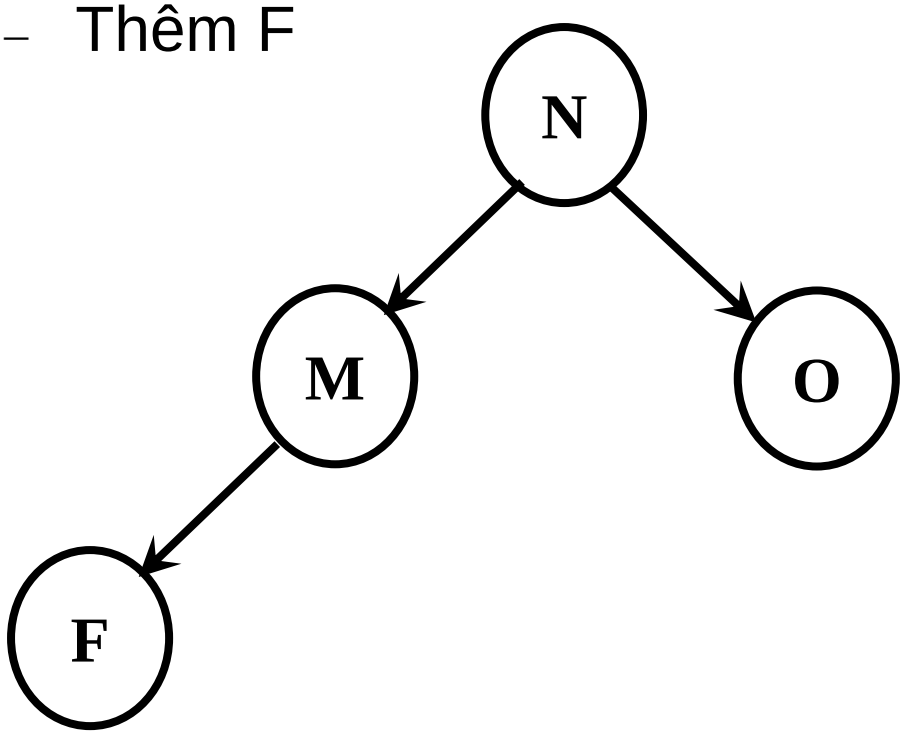
- Mất cân bằng tại M - Cân bằng lại



- Thêm F

9. ÁP DỤNG

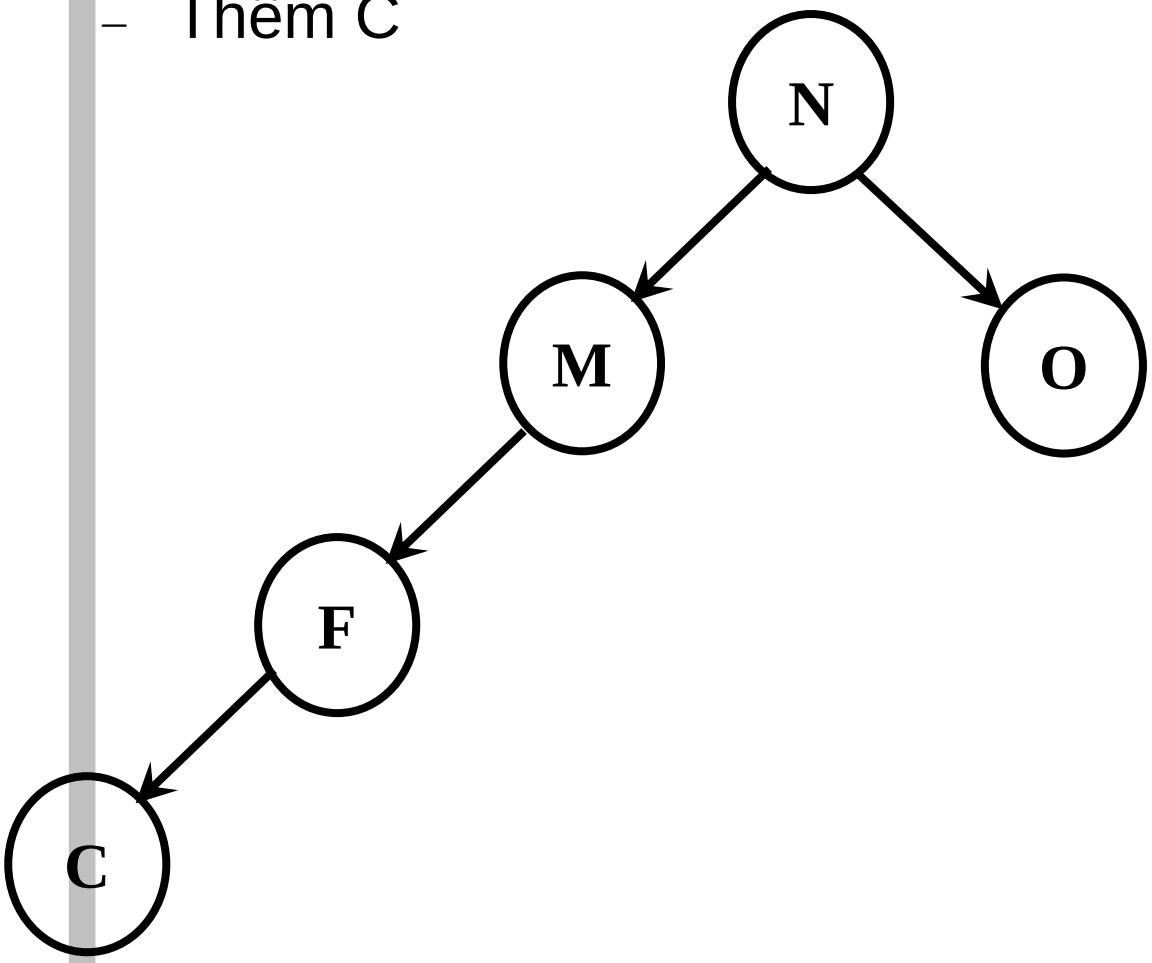
- Thêm F



- Thêm C

9. ÁP DỤNG

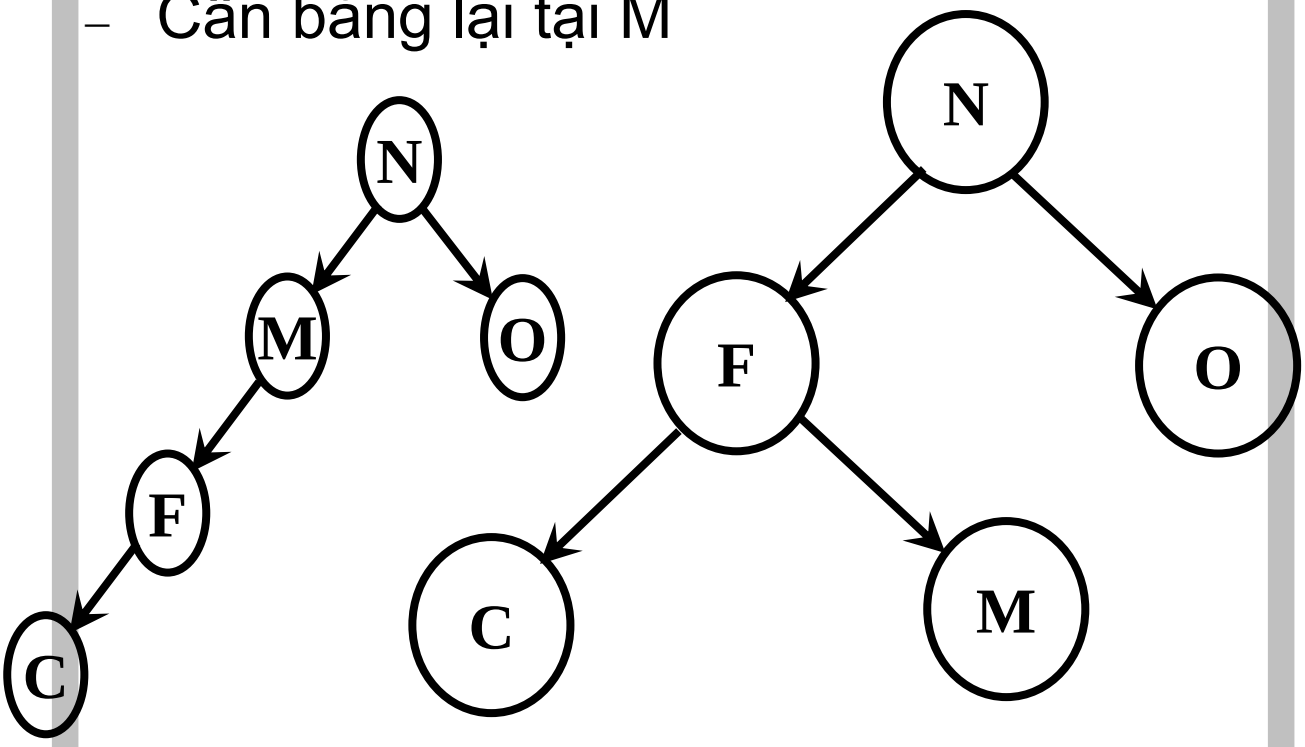
- Thêm C



- Mất cân bằng tại M

9. ÁP DỤNG

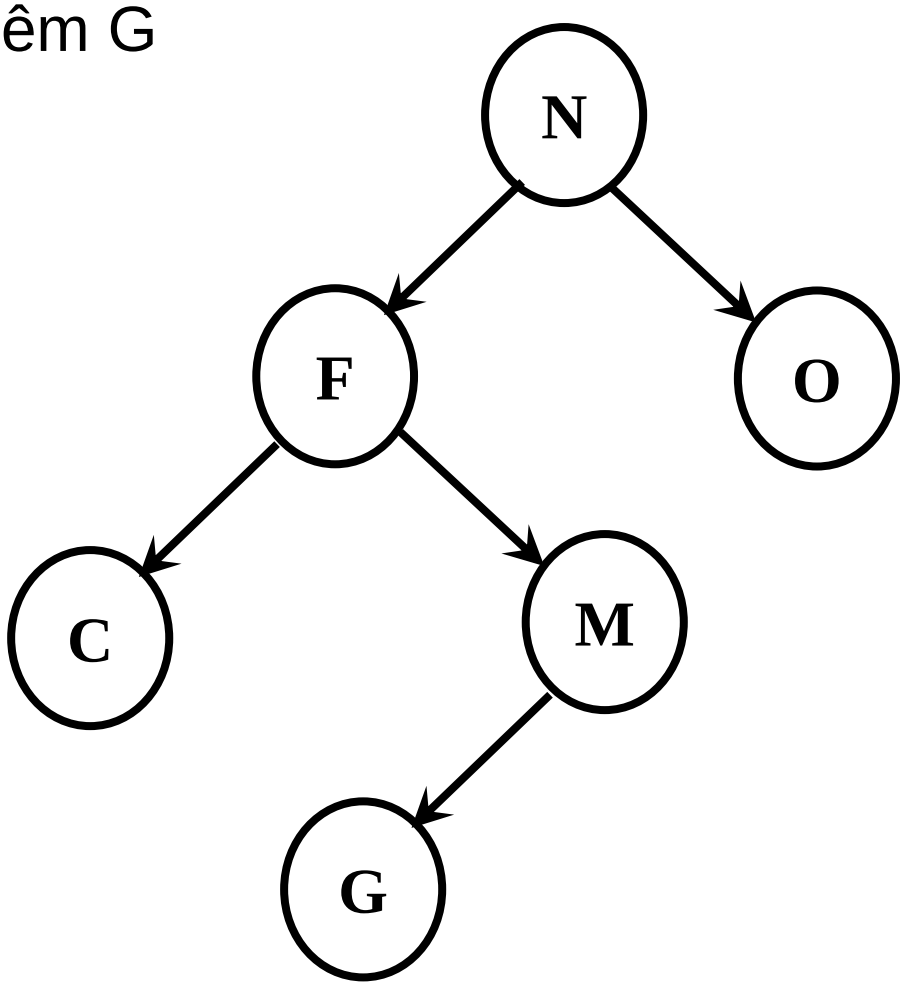
- Cân bằng lại tại M



- Thêm G

9. ÁP DỤNG

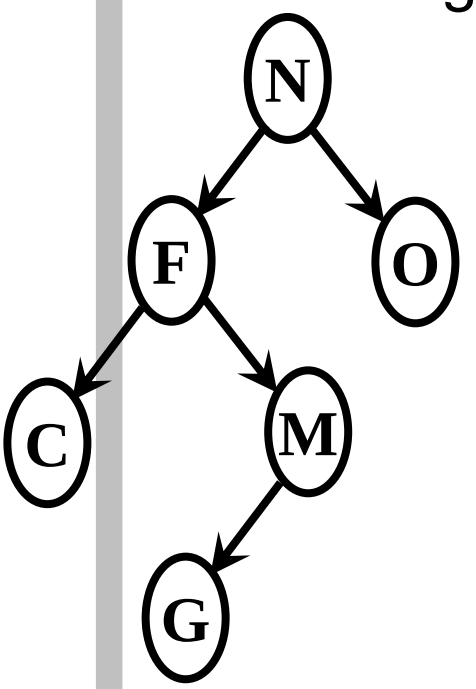
- Thêm G

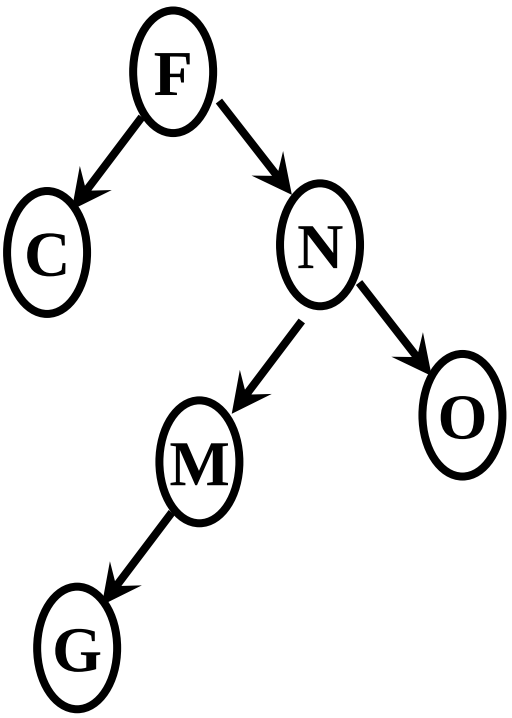


- Mất cân bằng tại N – Cân bằng lại

9. ÁP DỤNG

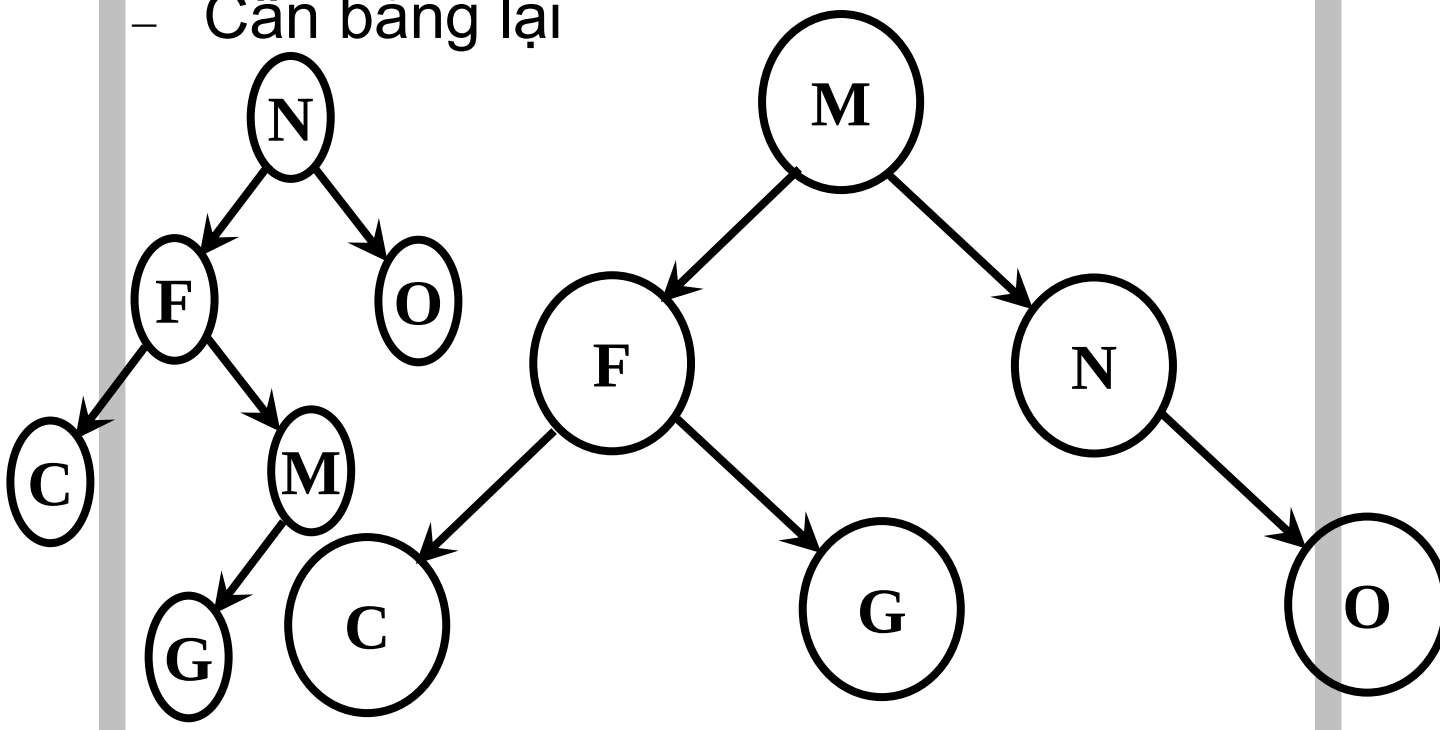
- Cân bằng lại





9. ÁP DỤNG

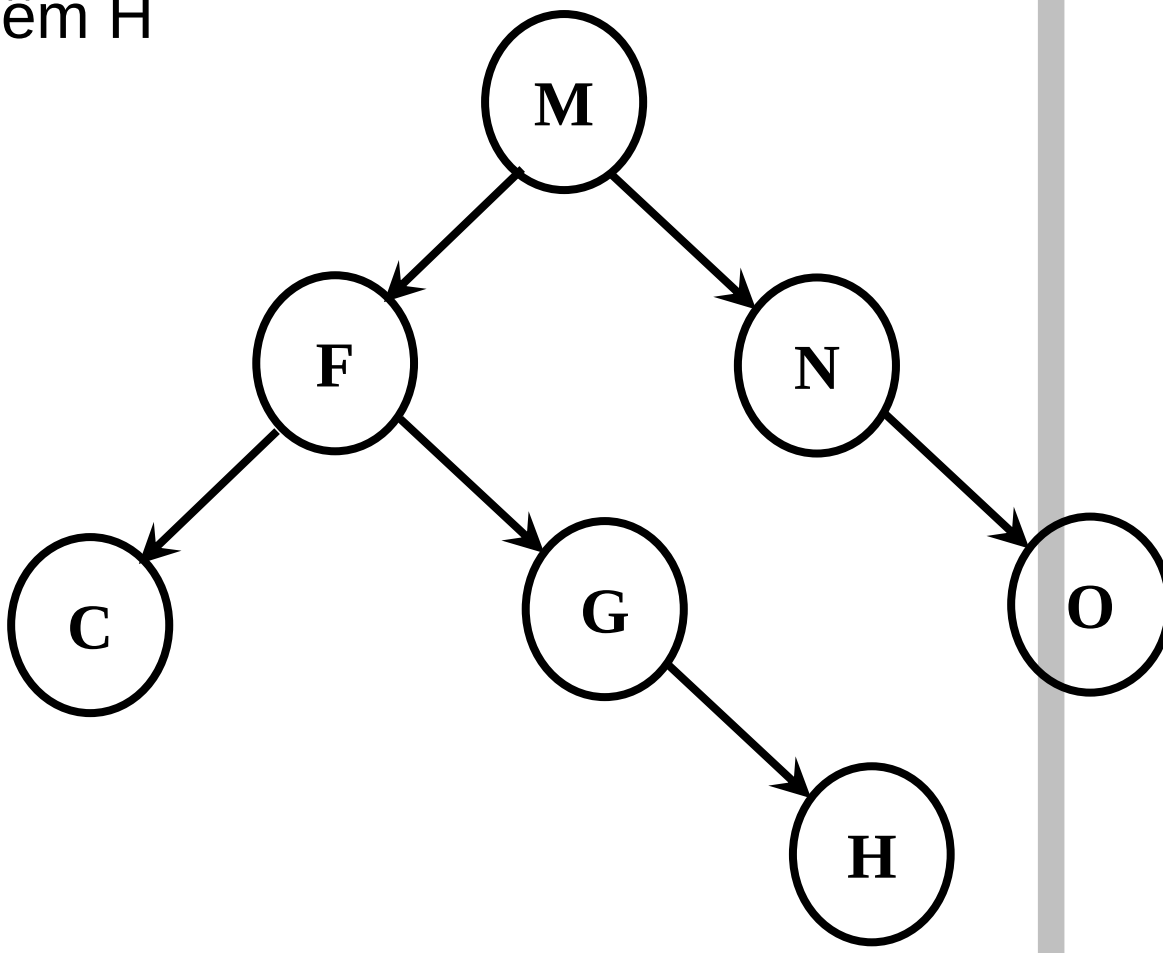
- Cân bằng lại



- Thêm H

9. ÁP DỤNG

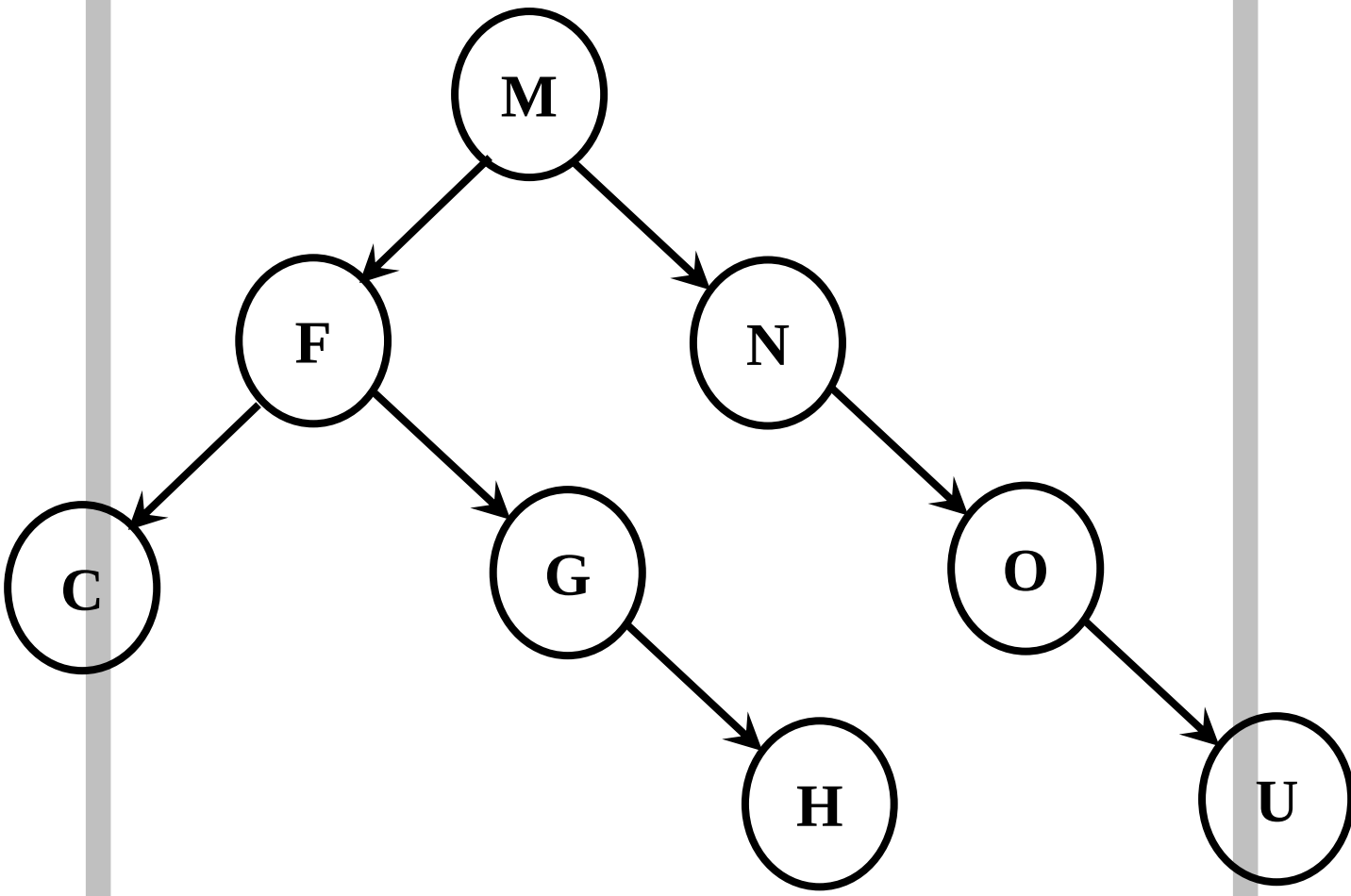
- Thêm H



- Thêm U

9. ÁP DỤNG

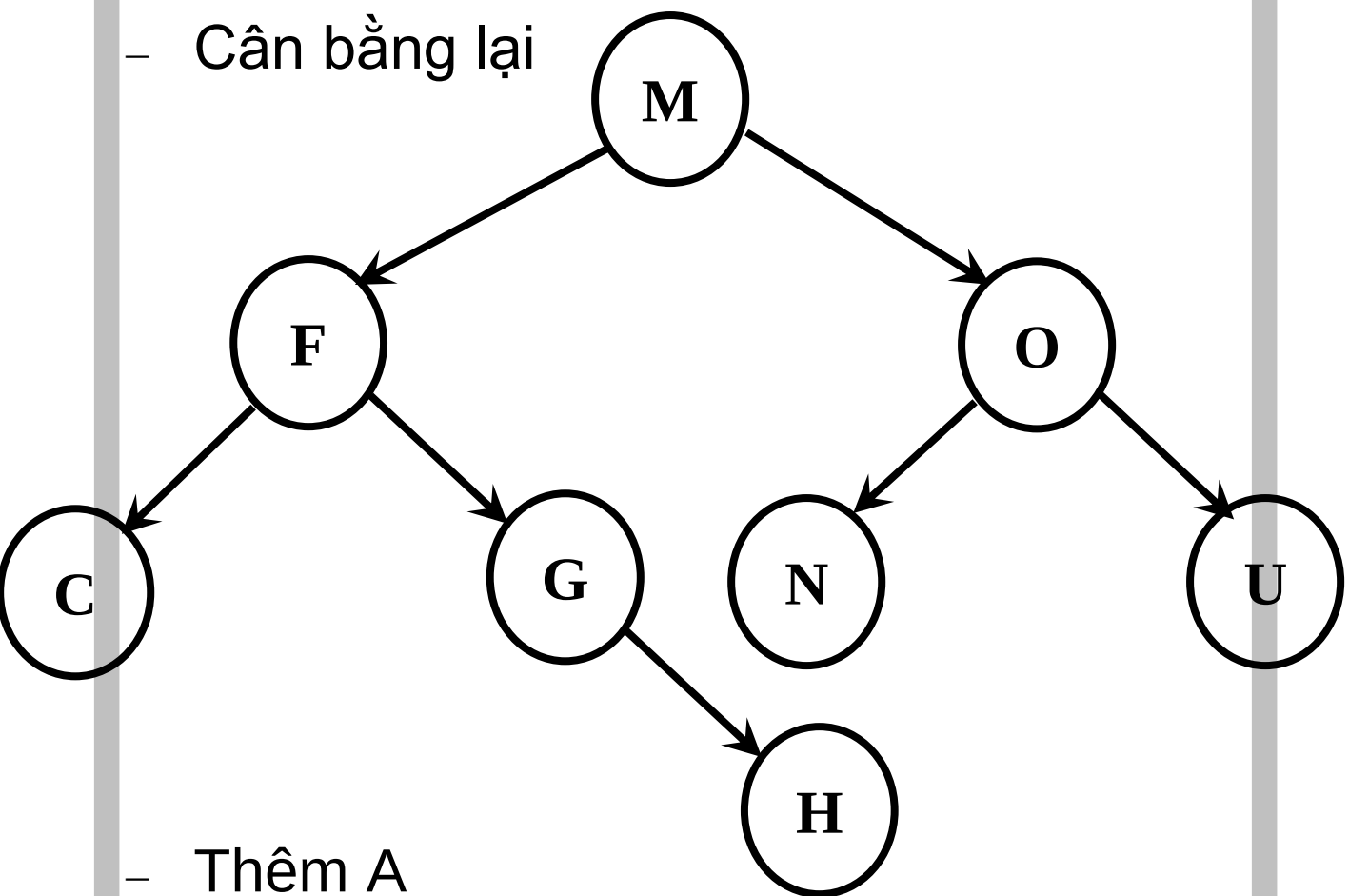
- Thêm U



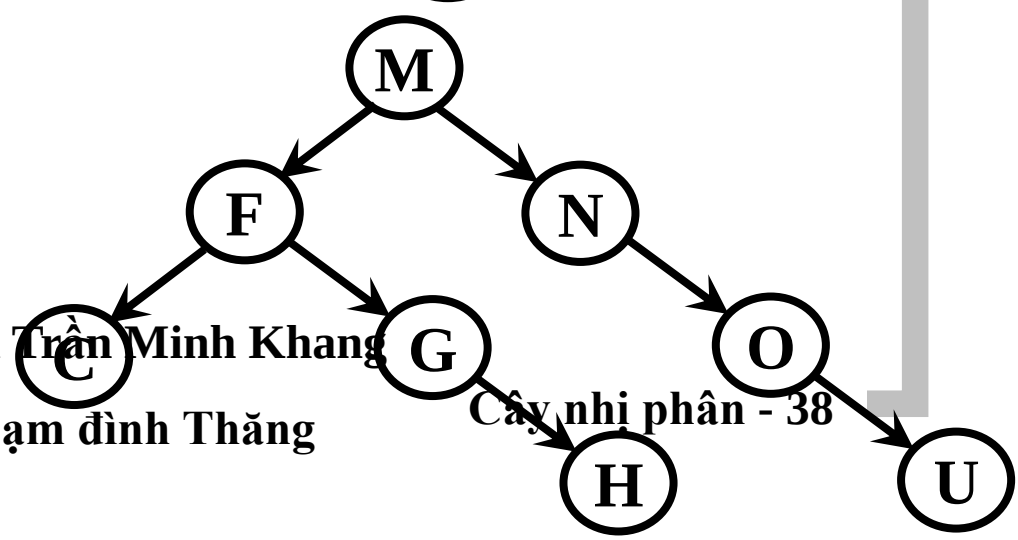
- Mất cân bằng tại N

9. ÁP DỤNG

- Cân bằng lại

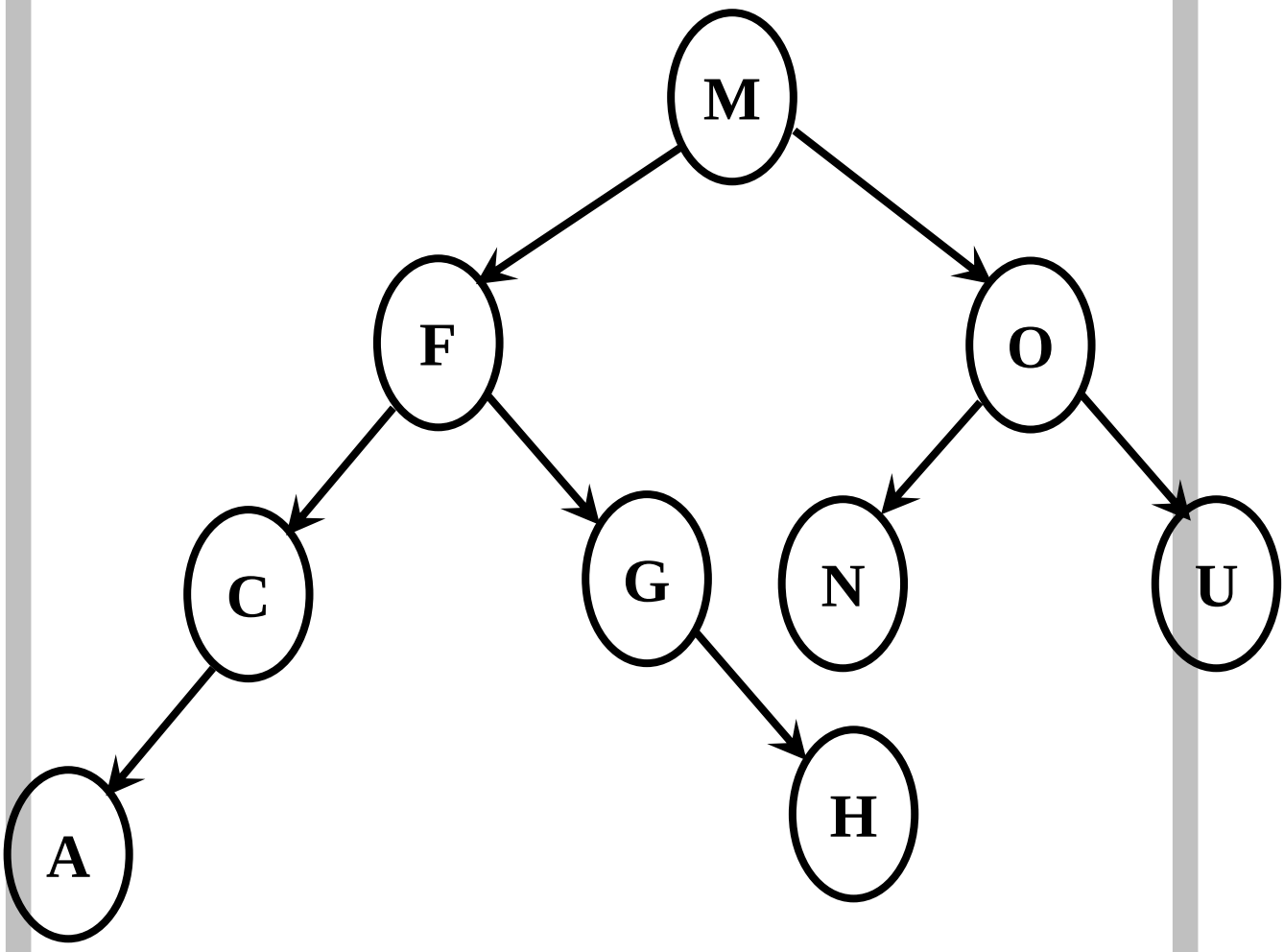


- Thêm A



9. ÁP DỤNG

- Thêm A



- Thêm E

9. ÁP DỤNG

- Thêm E