# BAN HỌC TẬP CÔNG NGHỆ PHẦN MỀM

## TRAINING CUỐI KỲ HỌC KỲ I NĂM HỌC 2022 – 2023

ĐOÀN KHOA
CÔNG NGHỆ PHẦN MỀM
BAN HỌC TẬP

Sharing is learning

### BAN HỌC TẬP


*Khoa Công nghệ Phần mềm*

*Trường Đại học Công nghệ Thông tin*

*Đại học Quốc gia thành phố Hồ Chí Minh*


### CONTACT


*bht.cnpm.uit@gmail.com*

*fb.com/bhtcnpm*

*fb.com/groups/bht.cnpm.uit*

**TRAINING**

# CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

**Thời gian:** 19:30 thứ 3 ngày 20/06/2023

**Địa điểm:** Microsoft Team

**Trainers:** Phạm Trung Nguyên – KTMP2022.2

Nguyễn Hoài Như – KTPM2022.2




Sharing is learning

# Câu 1:

a. Cho biết độ phức tạp của giải thuật Merge Sort theo định nghĩa Big-O. Trình bày các bước thực hiện giải thuật. Vẽ sơ đồ từng bước thực hiện giải thuật Merge Sort để sắp xếp dãy sau theo thứ tự tăng dần: 9 8 2 3 4 1 3 2 5 7

b. Cho biết độ phức tạp của giải thuật Binary Search theo định nghĩa Big-O. Trình bày các bước thực hiện giải thuật. Vẽ sơ đồ từng bước thực hiện giải thuật Binary Search để tìm kiếm phần tử 10 trong dãy sau: 1 2 2 3 3 4 5 7 8 9

# Câu 1a:

Độ phức tạp của giải thuật Merge Sort:

**Best case: O(nlogn)**

**Average case: O(nlogn)**

**Worst case: O(nlogn)**

# Câu 1a:

**Các bước thực hiện:**

Giả sử ta cần sắp xếp mảng A[l..r], trong đó l, r là các số nguyên không âm, l là chỉ số đầu và r là chỉ số cuối của mảng. Ta chia mảng thành hai mảng con bởi chỉ số m nằm giữa l và r ( m = ( l + r ) / 2). Các mảng con A[l..m] và A[m+1...r] được sắp xếp bằng cách gọi đệ quy thủ tục merge sort. Sau đó ta gộp hai mảng con A[l...m] và A[m+1...r] đã được sắp thành mảng A[a...b] được sắp. Giả sử merge(A,l,m,r) là hàm gộp hai mảng con đã được sắp A[l...m] và A[m+ 1...r] thành mảng A[l...r] được sắp. Thuật toán sắp xếp hoà nhập được biểu diễn bởi hàm đệ quy sau.

# Câu 1a:

```
void mergeSort(int A[], int l, int r)
{
    if (l < r)
    {
        int m = (l + r) / 2;
        mergeSort(A, l, m);
        mergeSort(A, m + 1, r);
        merge(A, l, m, r);
    }
}
```

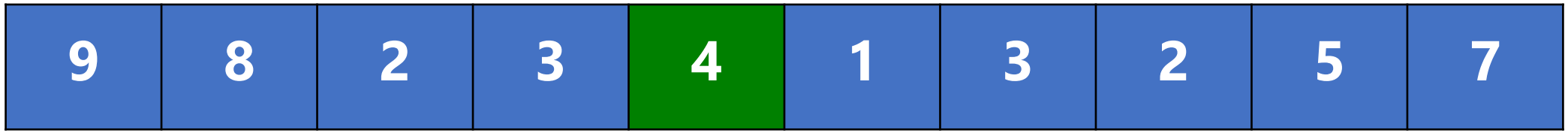
# Câu 1a:

Sơ đồ chạy từng bước giải thuật:

| 9 | 8 | 2 | 3 | 4 | 1 | 3 | 2 | 5 | 7 |
|---|---|---|---|---|---|---|---|---|---|

# Câu 1a:

Chọn mid là array[4] = 4

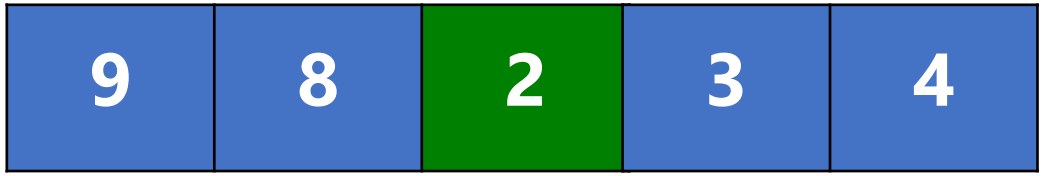


# Câu 1a:

Chọn mid là array[4] = 4

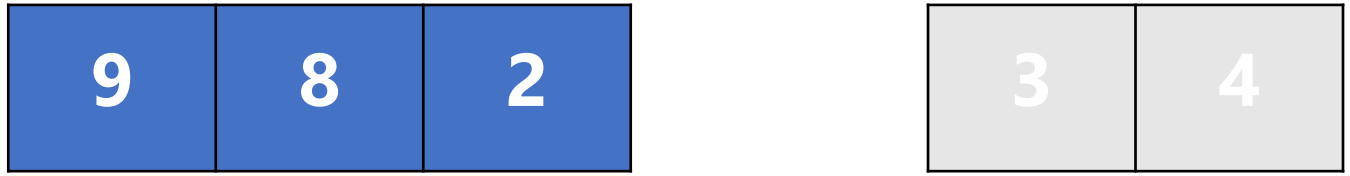

# Câu 1a:

Chọn mid là array[2] = 2 (đoạn 9, 4)




# Câu 1a:

Chọn mid là array[2] = 2 (đoạn 9, 4)

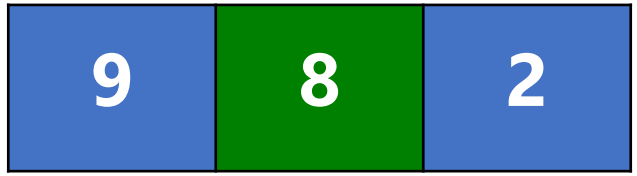


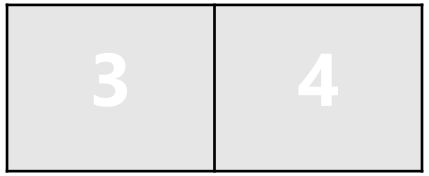




# Câu 1a:

Chọn mid là array[1] = 8 (đoạn 9, 2)




# Câu 1a:

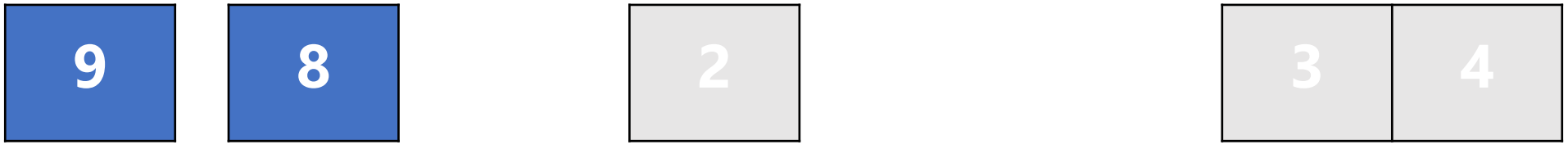
Chọn mid là array[1] = 8 (đoạn 9, 2)


# Câu 1a:

Chọn mid là array[0] = 9 (đoạn 9, 8)

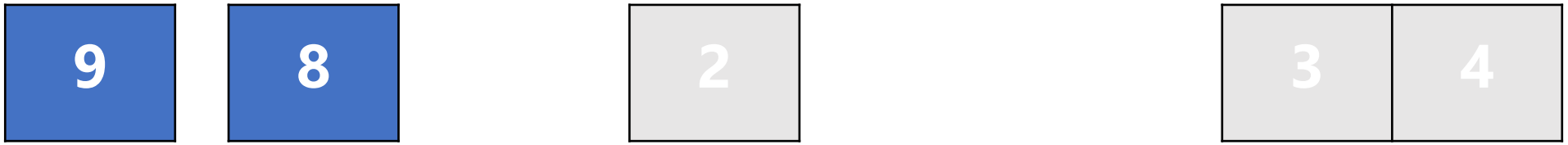

# Câu 1a:

Chọn mid là array[0] = 9 (đoạn 9, 8)




# Câu 1a:

Merge đoạn (9) và (8)

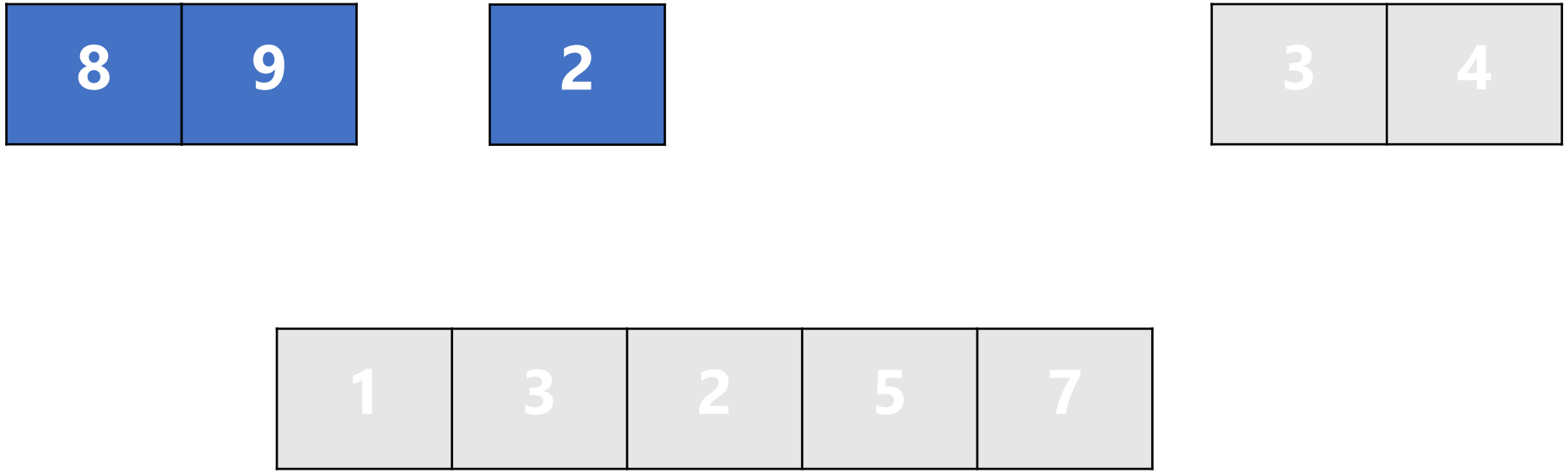



# Câu 1a:

Merge đoạn (9) và (8)

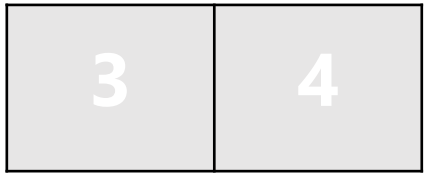

# Câu 1a:

Merge đoạn (8, 9) và (2)



# Câu 1a:

Merge đoạn (8, 9) và (2)


# Câu 1a:

Chọn mid là array[3] = 3 (đoạn 3, 4)


# Câu 1a:

Chọn mid là array[3] = 3 (đoạn 3, 4)


# Câu 1a:

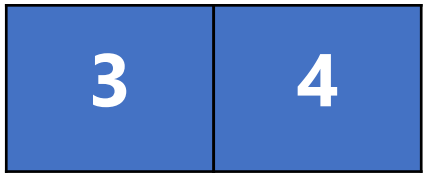
Merge đoạn (3) và (4)


# Câu 1a:

Merge đoạn (3) và (4)

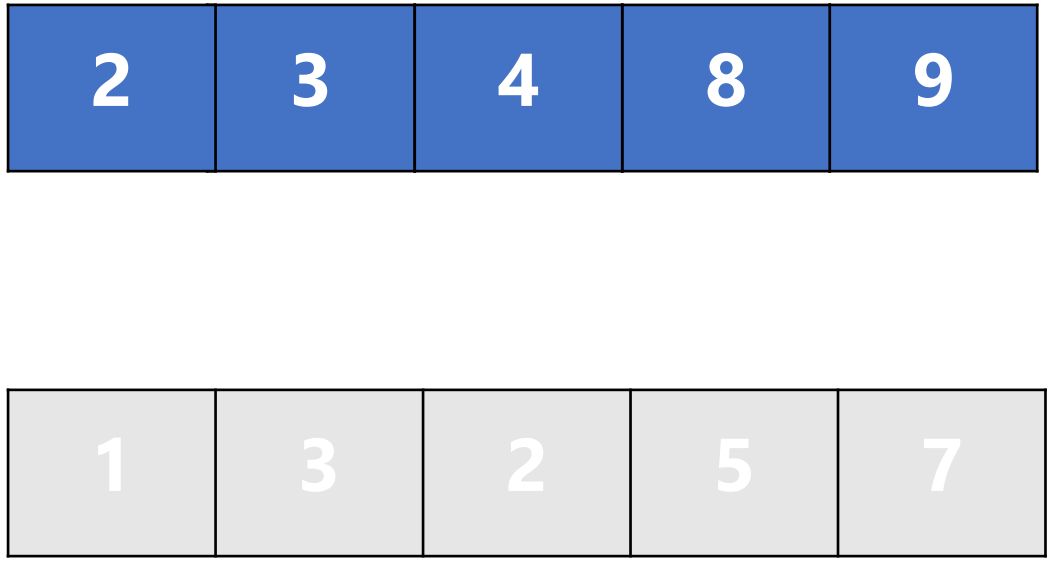

# Câu 1a:

Merge đoạn (2, 9) và (3, 4)

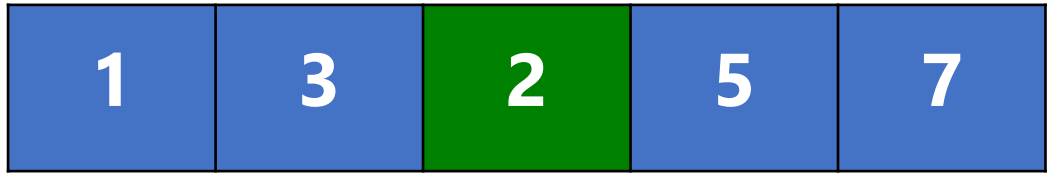

# Câu 1a:

Merge đoạn (2, 9) và (3, 4)



# Câu 1a:

Chọn mid là array[7] = 2 (đoạn 1, 7)

# Câu 1a:

Chọn mid là array[7] = 2 (đoạn 1, 7)

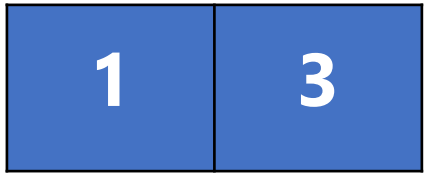


# Câu 1a:

Chọn mid là array[6] = 3 (đoạn 1, 2)


# Câu 1a:

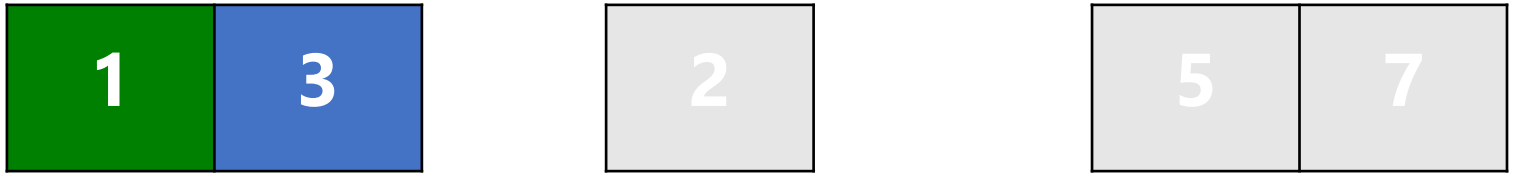
Chọn mid là array[6] = 3 (đoạn 1, 2)



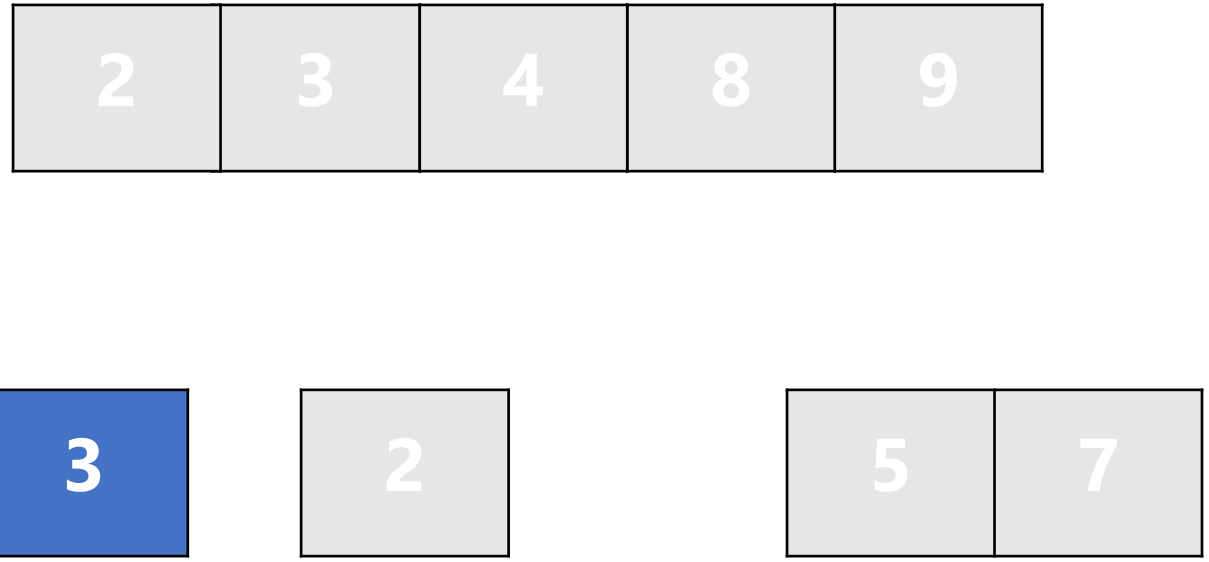
# Câu 1a:

Chọn mid là array[5] = 1 (đoạn 1, 3)

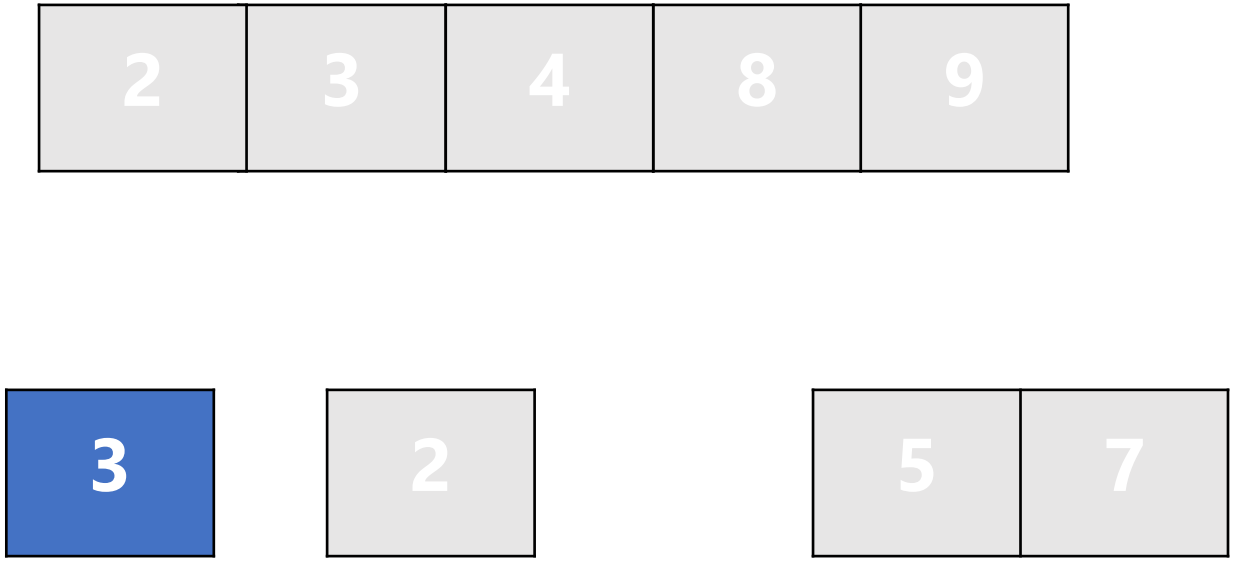
# Câu 1a:

Chọn mid là array[5] = 1 (đoạn 1, 3)



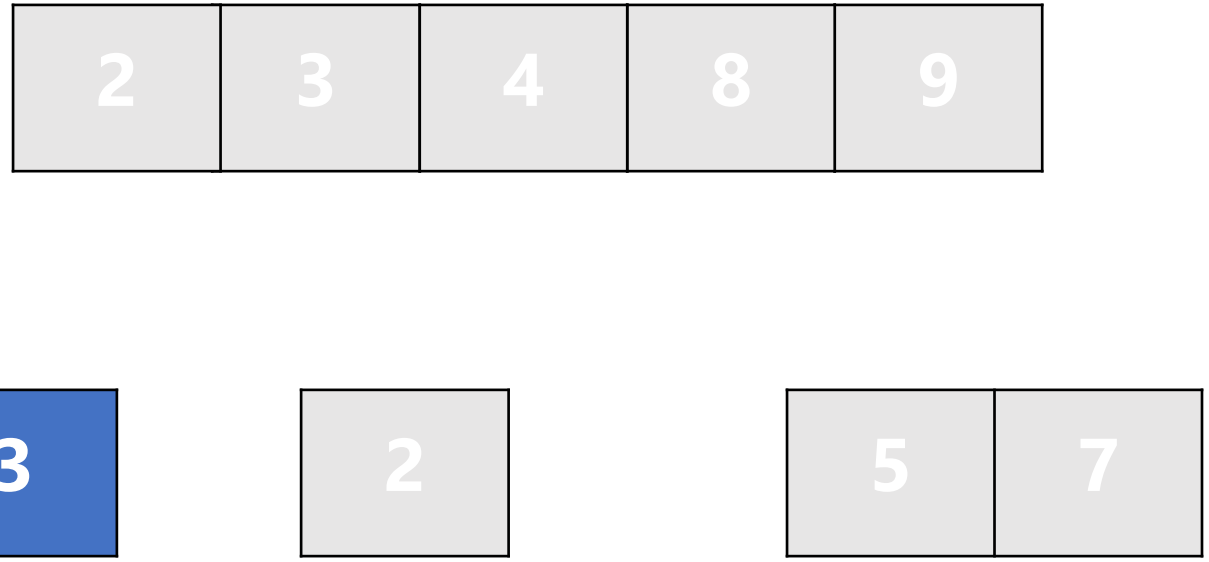
# Câu 1a:

Merge đoạn (1) và (3)



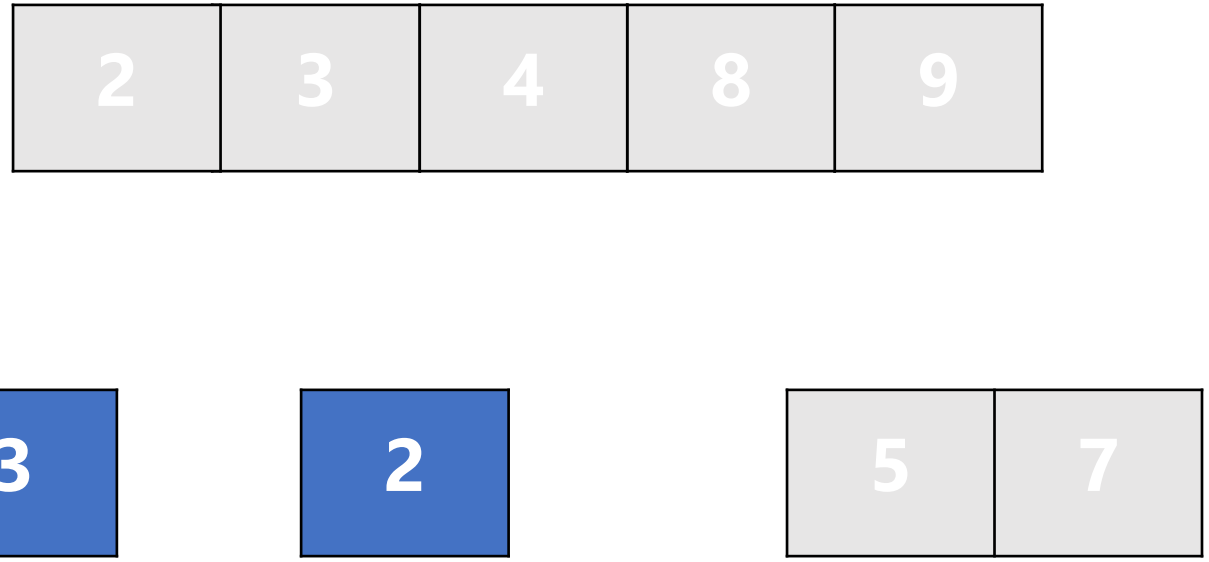
# Câu 1a:

Merge đoạn (1) và (3)



# Câu 1a:

Merge đoạn (1, 3) và (2)



# Câu 1a:

Merge đoạn (1, 3) và (2)



# Câu 1a:

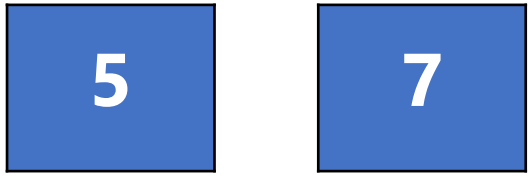
Chọn mid là array[8] = 5 (đoạn 5, 7)



# Câu 1a:

Chọn mid là array[8] = 5 (đoạn 5, 7)

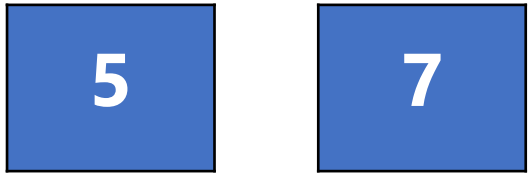




# Câu 1a:

Merge đoạn (5) và (7)

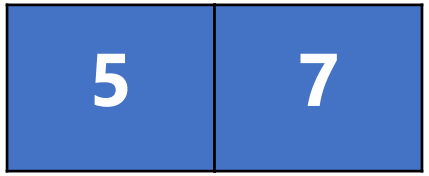




# Câu 1a:

Merge đoạn (5) và (7)

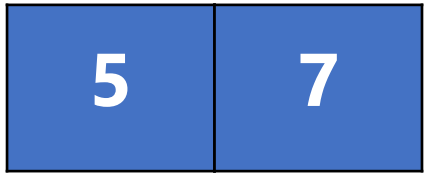




# Câu 1a:

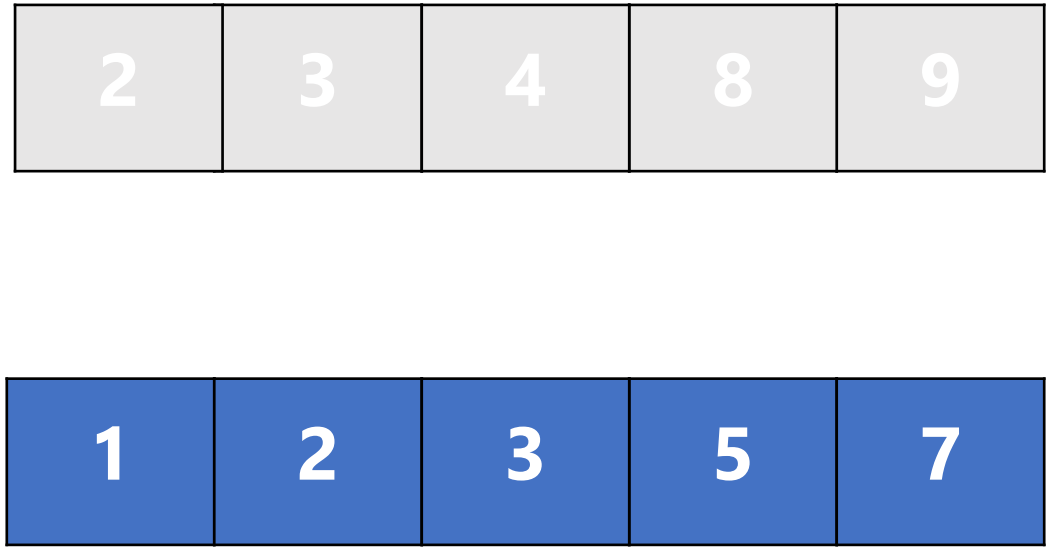
Merge đoạn (1, 3) và (5, 7)





# Câu 1a:

Merge đoạn (1, 3) và (5, 7)

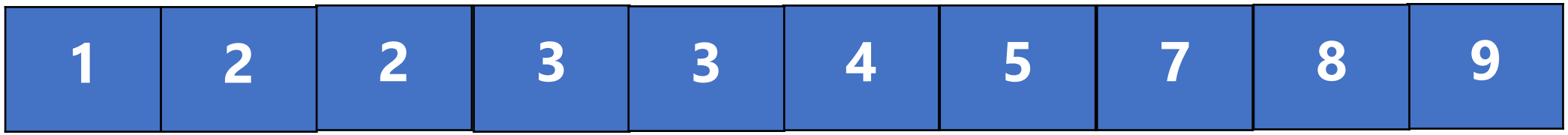


# Câu 1a:

Merge đoạn (2,9) và (1,7)

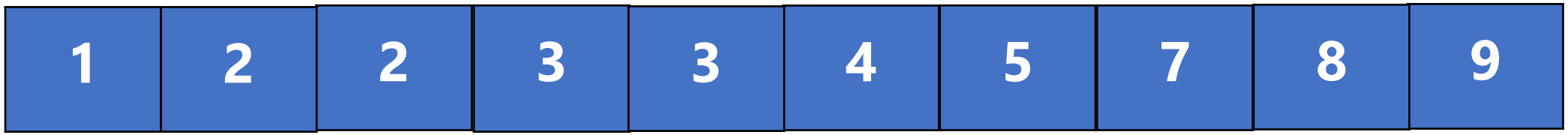

# Câu 1a:

Merge đoạn (2,9) và (1,7)



# Câu 1a:

Sau khi sắp xếp:



# Một số câu hỏi trắc nghiệm

Thuật toán quick sort có độ phức tạp xấu nhất khi nào?

a. Tất cả các phần tử trong mảng giống nhau

b. Tất cả các ý đều đúng

c. Mảng được sắp xếp đã có thứ tự theo yêu cầu sắp

d. Mảng được sắp xếp có thứ tự đảo ngược với yêu cầu sắp xếp

# Một số câu hỏi trắc nghiệm

Thuật toán sắp xếp nào sau đây cho thời gian chạy tốt nhất khi áp dụng trên một mảng đã được sắp xếp hoặc sắp xếp gần hết (tối đa 1 hoặc hai phần tử bị đặt sai vị trí)

a. Quick sort

b. Merge sort

c. Heap sort

d. Insertion sort

# Một số câu hỏi trắc nghiệm

Trong số các thuật toán sắp xếp sau, thuật toán nào có thời gian chạy ít phụ thuộc nhất vào thứ tự đầu vào ban đầu?

a. Quick sort

b. Merge sort

c. Heap sort

d. Insertion sort

# Một số câu hỏi trắc nghiệm

Thuật toán sắp xếp nào sau đây cần sử dụng thêm bộ nhớ phụ trong quá trình sắp xếp?

a. Selection sort

b. Quick sort

c. Merge sort

d. Insertion sort

# Câu 1b:

Độ phức tạp của giải thuật **Binary Search**:

**Best case: O(1)**

**Average case: O(logn)**

**Worst case: O(logn)**

# Câu 1b:

**Các bước thực hiện:**

Khởi tạo lần lượt hai biến left = 0 và right = n-1 (n là số phần tử trong dãy).

Chạy vòng lặp while với điều kiện left <= right:

- Tính chỉ số giữa middle = (left + right) / 2.

- So sánh phần tử ở vị trí middle với phần tử cần tìm.

+ Nếu phần tử cần tìm bằng phần tử ở vị trí middle, trả về vị trí middle và kết thúc thuật toán.

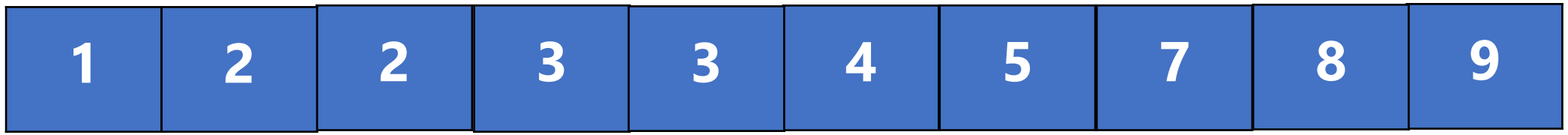
+ Nếu phần tử cần tìm lớn hơn phần tử ở vị trí middle, ta cập nhật lại left = middle + 1.

+ Nếu phần tử cần tìm nhỏ hơn phần tử ở vị trí middle, ta cập nhật lại right = middle - 1.

- Nếu không tìm thấy phần tử cần tìm trong dãy, trả về giá trị -1 hoặc thông báo không tìm thấy phần tử đó.

# Câu 1b

Sơ đồ chạy từng bước giải thuật:

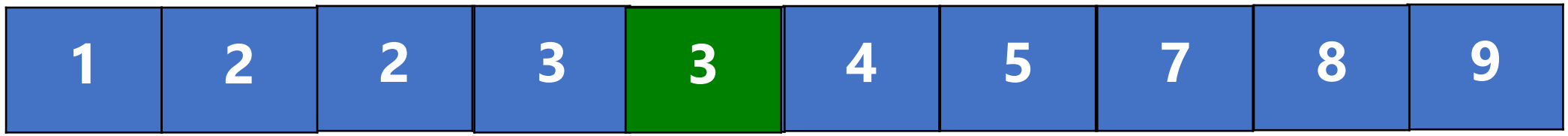


# Câu 1b

Left = 0, right = 9,

chọn mid là array[4] = 3,

x = 10 > 3 => xét đoạn (4, 9)

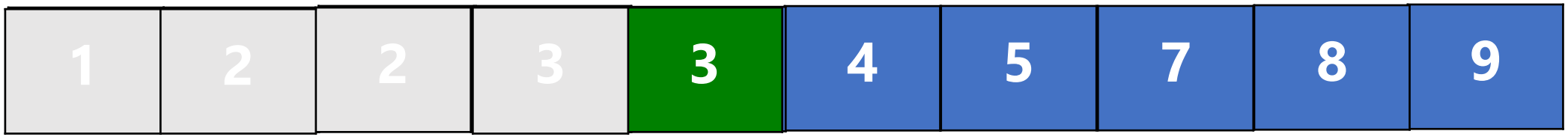


# Câu 1b

Left = 0, right = 9,

chọn mid là array[4] = 3,

x = 10 > 3 => xét đoạn (4, 9)

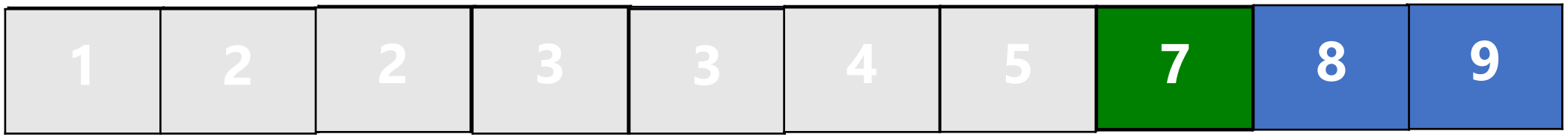


# Câu 1b

Left = 5, right = 9,

chọn mid là array[7] = 7,

x = 10 > 7 => xét đoạn (8, 9)

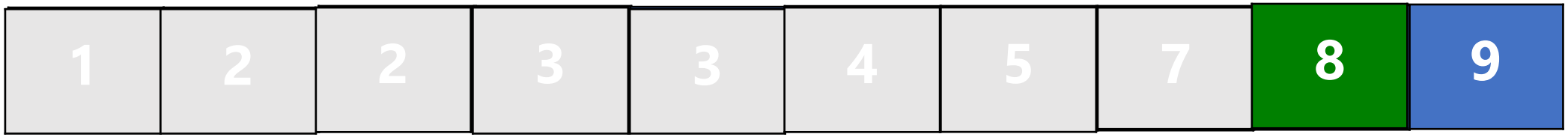


# Câu 1b

Left = 8, right = 9,

chọn mid là array[8] = 8,

x = 10 > 8 => xét đoạn (9)



# Câu 1b

Left = 9, right = 9,

chọn mid là array[9] = 9,

x = 10 > 9

# Câu 1b

Left = 10, right = 9

=> kết thúc thuật toán, trả về (-1): không tìm thấy 10

# Câu 2

Cho dãy số sau: 27, 66, 58, 23, 32, 8, 10, 21, 9, 4, 5, 99, 63, 72, 24

Hãy thực hiện các yêu cầu sau:

**a.** Xây dựng cây nhị phân tìm kiếm từ dãy số đã cho vào cây theo thứ tự thêm các số từ phải sang trái của dãy số.

**b.** Duyệt cây trong câu a theo NRL, LRN, RNL.

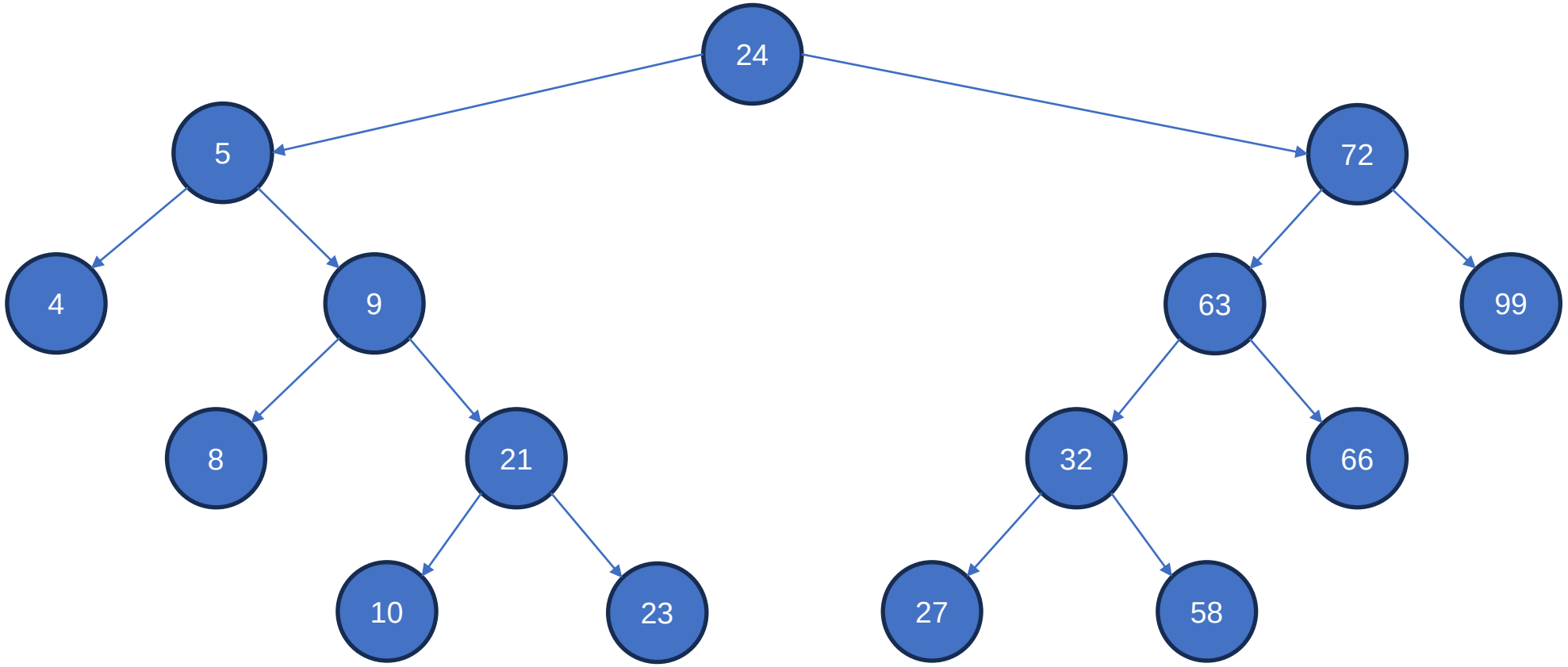
**c.** Xóa khỏi cây lần lượt các nút 10, 21, 24, 63, 27 (vẽ hình từng trường hợp) sao cho cây vẫn là cây nhị phân tìm kiếm sau khi xoá nút.

**d.** Viết hàm in ra màn hình các nút trên cây có duy nhất một nút con.

**e.** Viết hàm đếm số lượng nút lá, số nút hai con có trên cây.

# Câu 2a:

Dãy số: 27, 66, 58, 23, 32, 8, 10, 21, 9, 4, 5, 99, 63, 72, 24 <-



# Câu 2b:

LRN:

4 8 10 23 21 9 5 27 58 32 66 63 99 72 24

NRL:

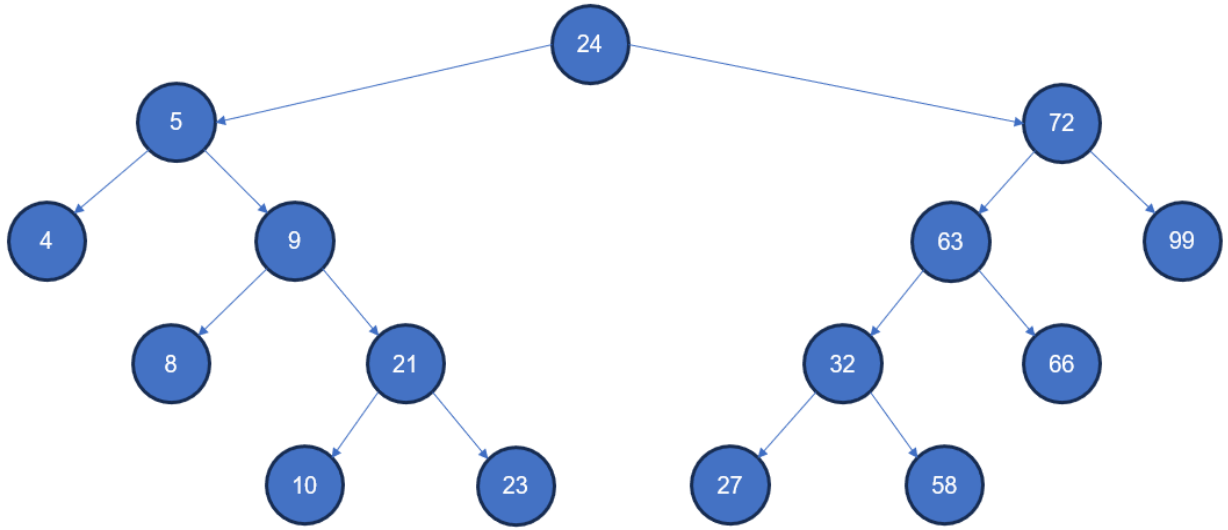
24 72 99 63 66 32 58 27 5 9 21 23 10 8 4

RNL:

99 72 66 63 58 32 27 24 23 21 10 9 8 5 4, **Giảm**

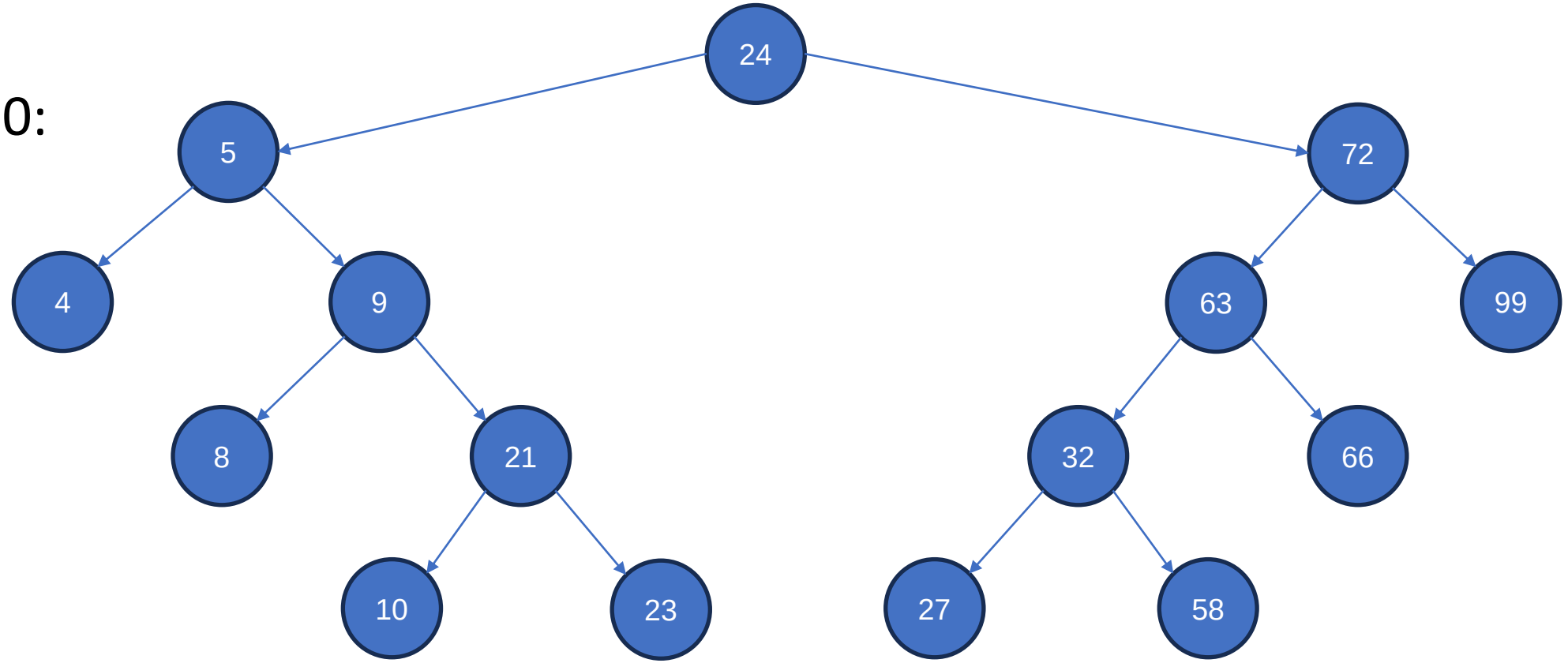
LNR:

4 5 8 9 10 21 23 24 27 32 58 63 66 72 99, **Tăng**



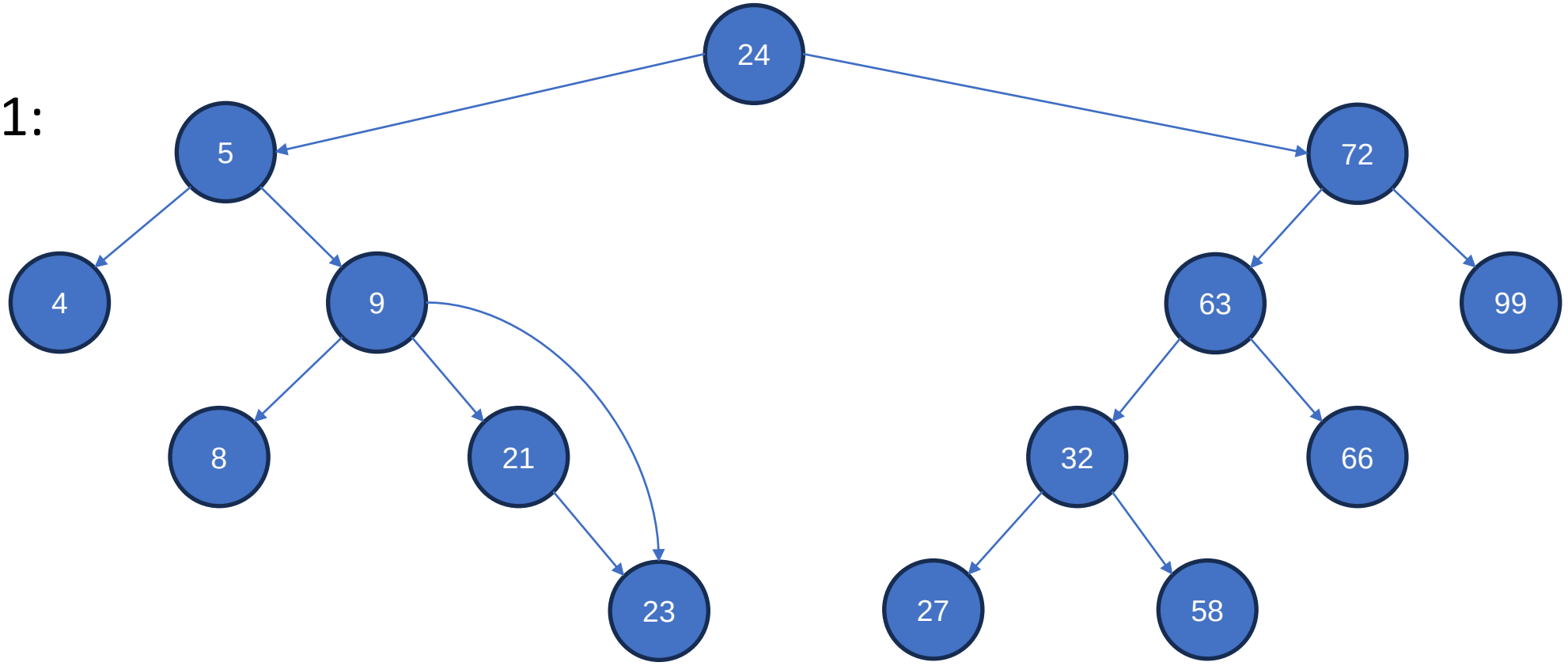
# Câu 2c:

Xoá 10:



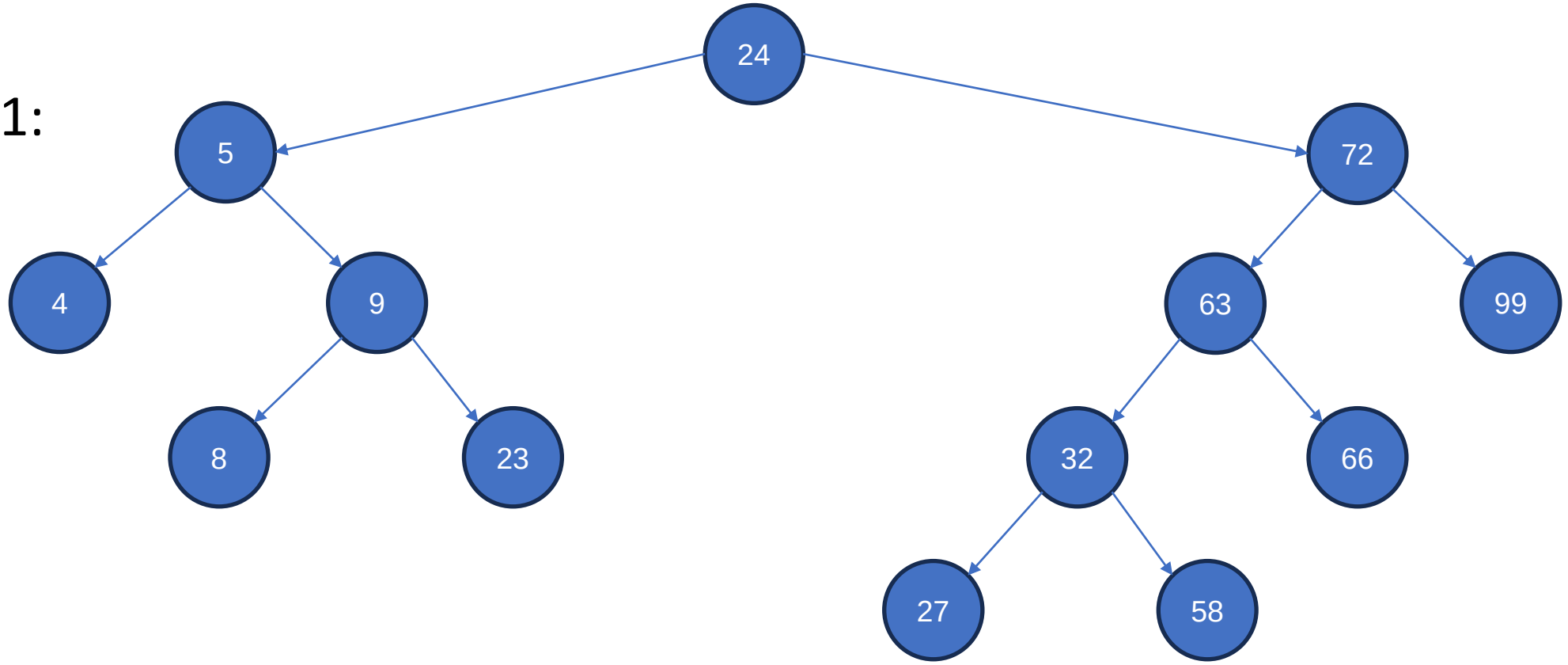
# Câu 2c:

Xoá 21:



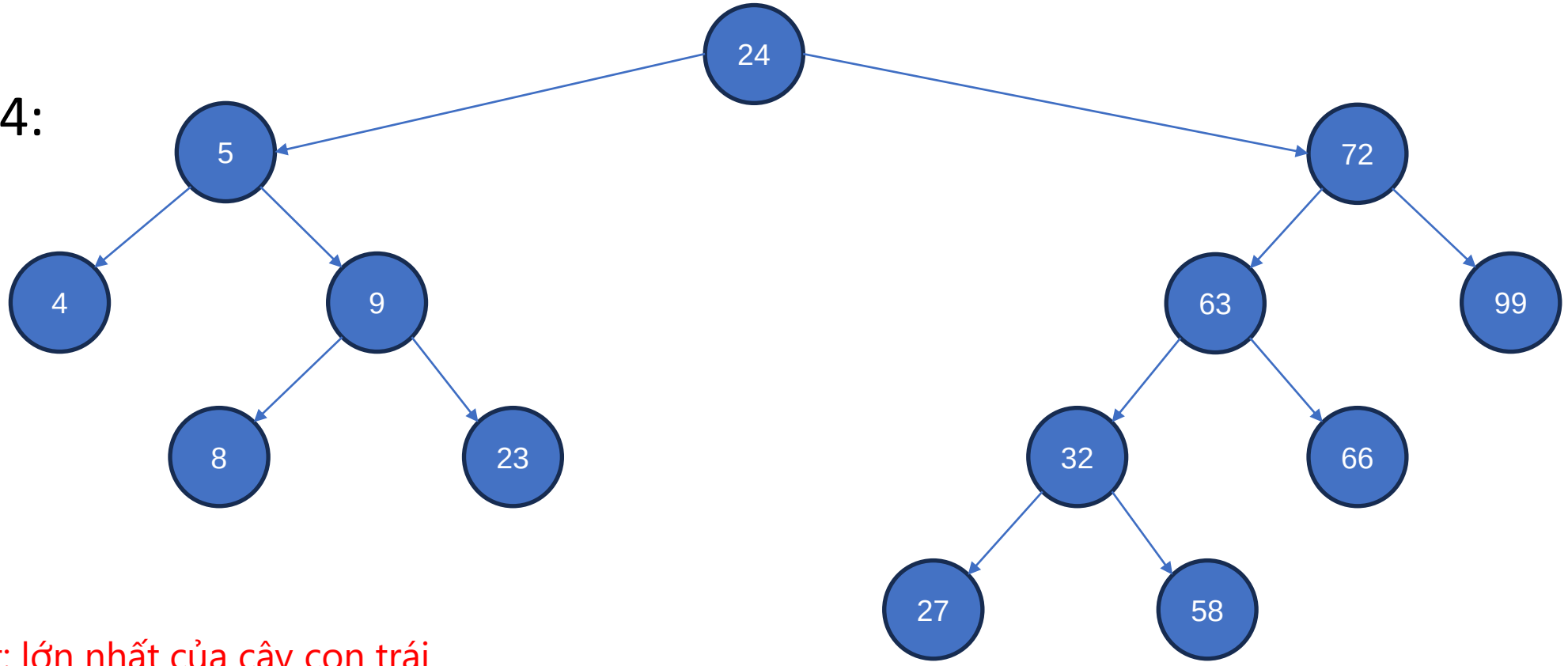
# Câu 2c:

Xoá 21:



# Câu 2c:

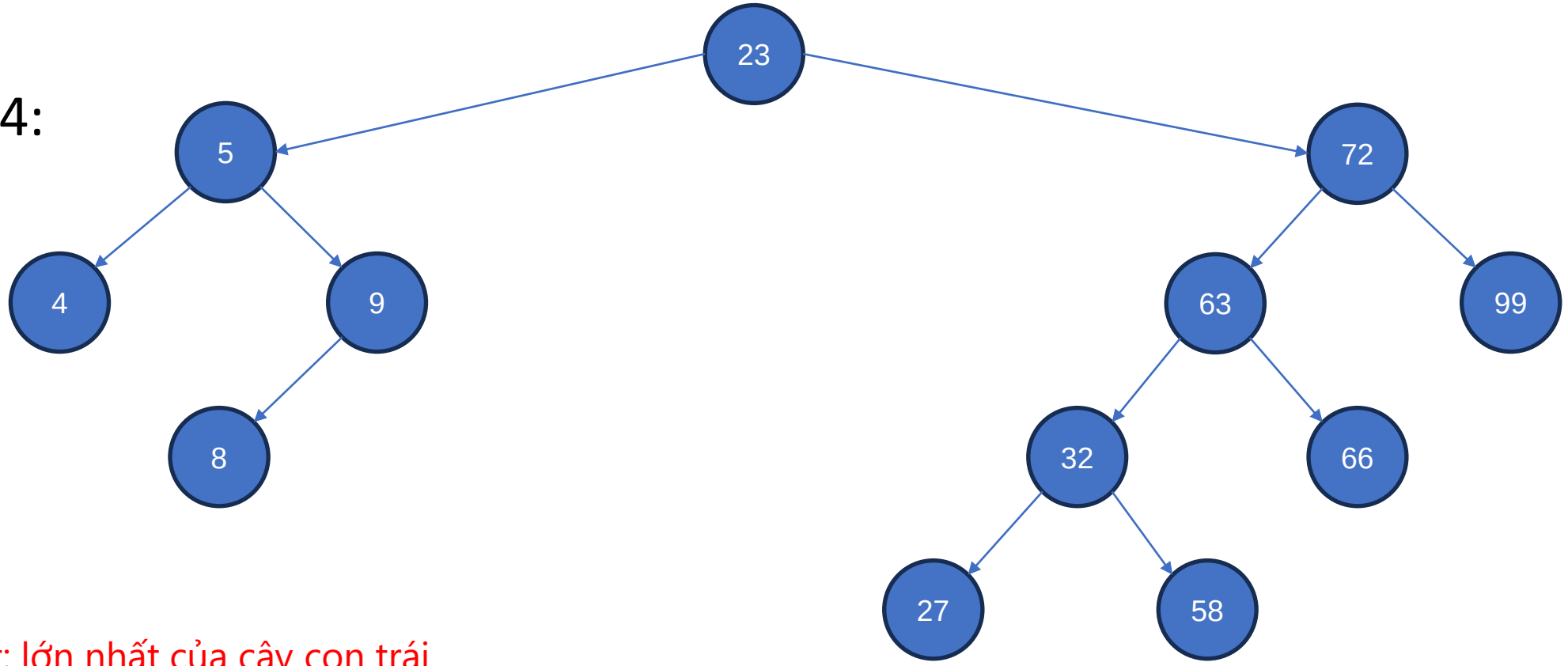
Xoá 24:




-> Phải nhất: lớn nhất của cây con trái

# Câu 2c:

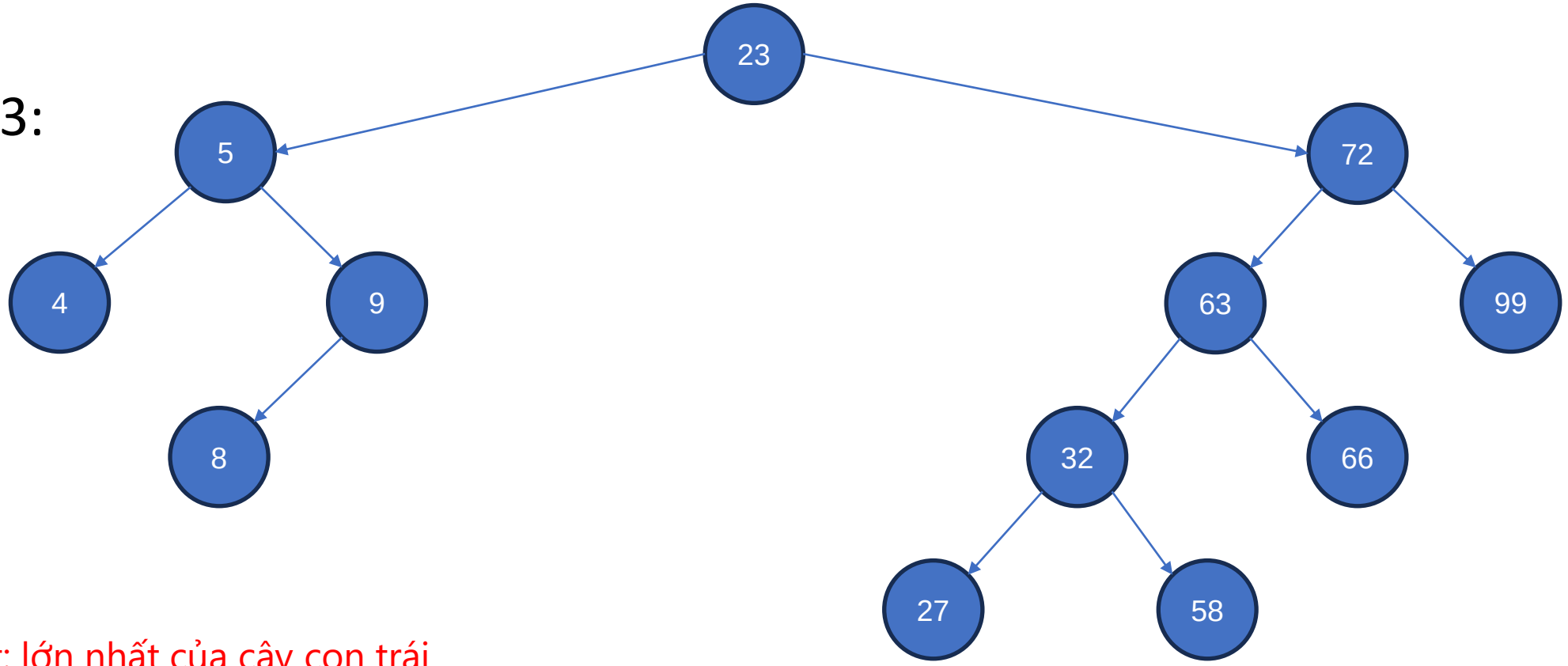
Xoá 24:




-> Phải nhất: lớn nhất của cây con trái

# Câu 2c:

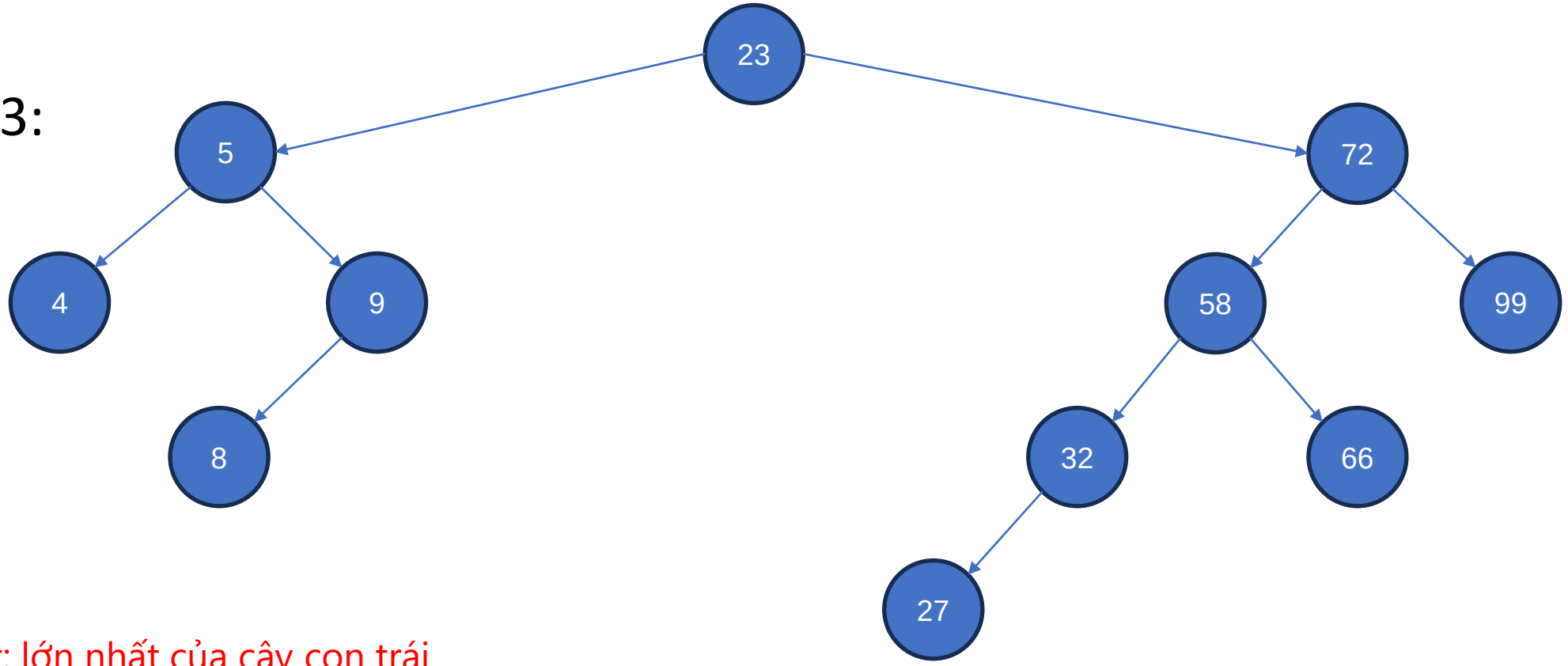
Xoá 63:




-> Phải nhất: lớn nhất của cây con trái

# Câu 2c:

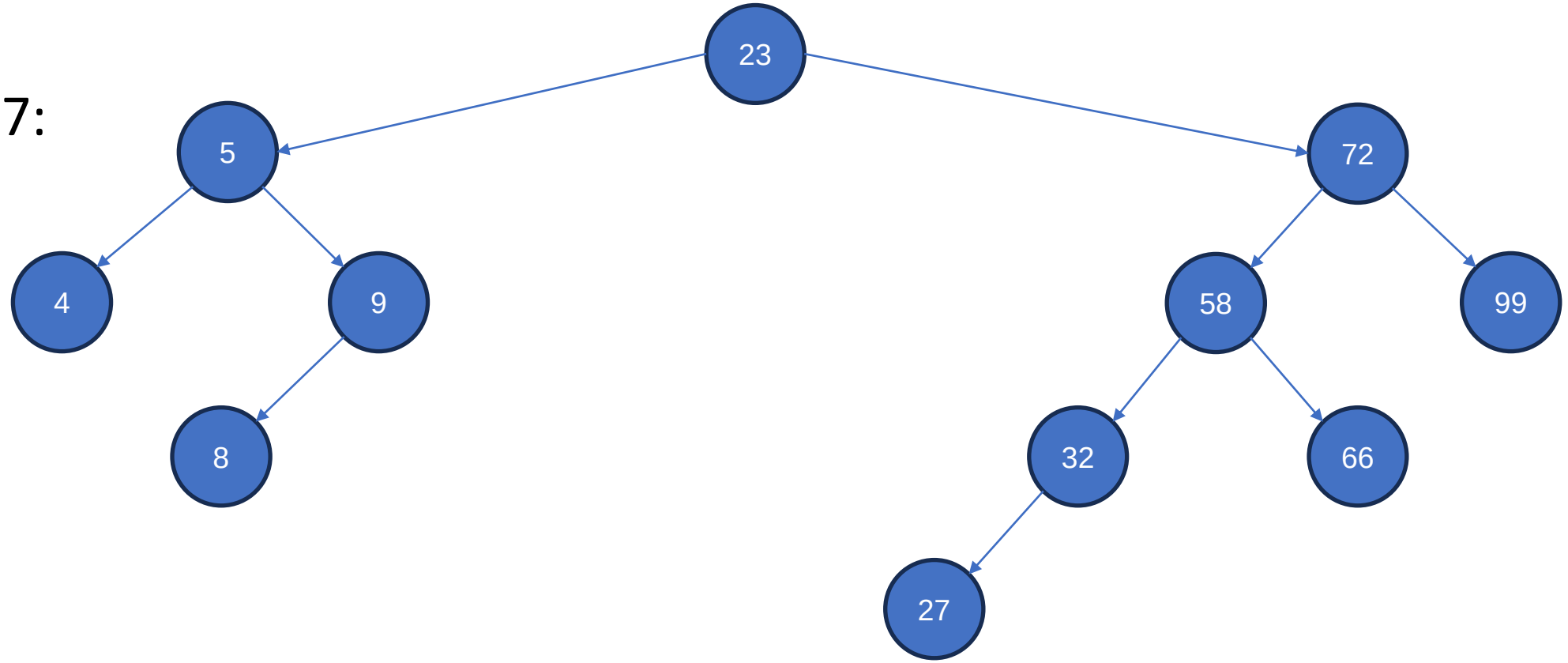
Xoá 63:



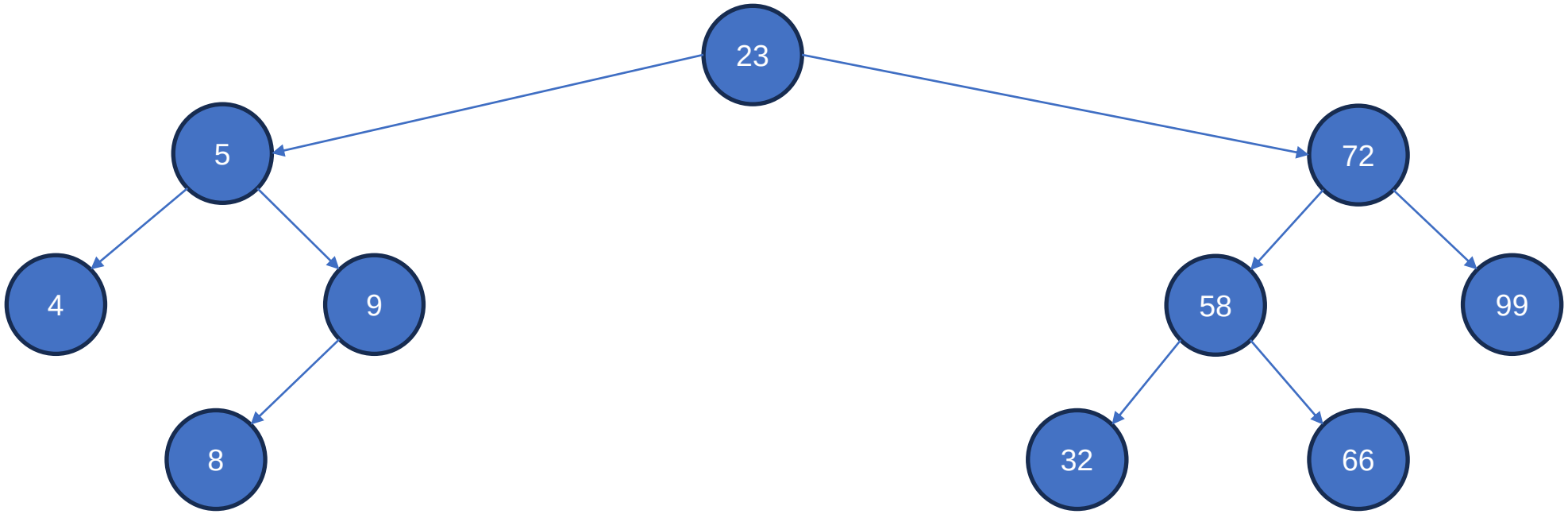

-> Phải nhất: lớn nhất của cây con trái

# Câu 2c:

Xoá 27:



# Câu 2c:



# Câu 2d:

```
void printNode1Con(TREE T)
    {
        if (T != NULL)
        {
                if ((T->pLeft == NULL && T->pRight != NULL) ||
                        (T->pLeft != NULL && T->pRight == NULL))
                {
                        cout << T->Key << " ";
                }
                printNode1Con(T->pLeft);
                printNode1Con(T->pRight);
        }
    }
```

# Câu 2e:

```
void DemNodeLa(TREE T, int& dem)
{
    if (T != NULL)
    {
            if (T->pLeft == NULL && T->pRight == NULL)
            {
                    dem++;
            }
            DemNodeLa(T->pLeft, dem);
            DemNodeLa(T->pRight, dem);
    }
}
```

# Câu 2e:

```
void DemNode2Con(TREE T, int& dem)
{
    if (T != NULL)
    {
            if (T->pLeft != NULL && T->pRight != NULL)
            {
                    dem++;
            }
            DemNode2Con(T->pLeft, dem);
            DemNode2Con(T->pRight, dem);
    }
}
```

# Câu 3:

Hãy tạo cây B-Tree bậc 3:

**a.** Lần lượt thêm các khóa **A, D, Z, B, F, G, H, O, N, P, X, C** vào cây. Và cho biết ở thao tác nào thì có thao tác **split node**.

**b.** Lần lượt xóa các khóa **P, D, F, C** khỏi cây. Xóa khóa nào thì chỉ cần thực hiện thao tác **underflow**, khóa nào thì phải thực hiện **catenate**.

# Câu 3a:

- Các bước tạo cây B-tree với bậc m:

+ Số cây con tối đa: m

+ Số cây con tối thiểu (cho nút trung gian): m/2 nếu m chẵn, (m+1)/2 nếu m lẻ

m/2 <= số cây con min <= (m+1)/2

+ Số khóa tối đa: m-1

+ Số khóa tối thiểu: số cây con tối thiểu – 1.

- Các bước tạo cây B-tree với bậc 3:

+ Số cây con tối đa: 3

+ Số cây con tối thiểu (cho nút trung gian): 2

+ Số khóa tối đa: 2

+ Số khóa tối thiểu: số cây con tối thiểu – 1 = 1

# Câu 3a:

Thêm A, D

# Câu 3a:

Thêm A, D

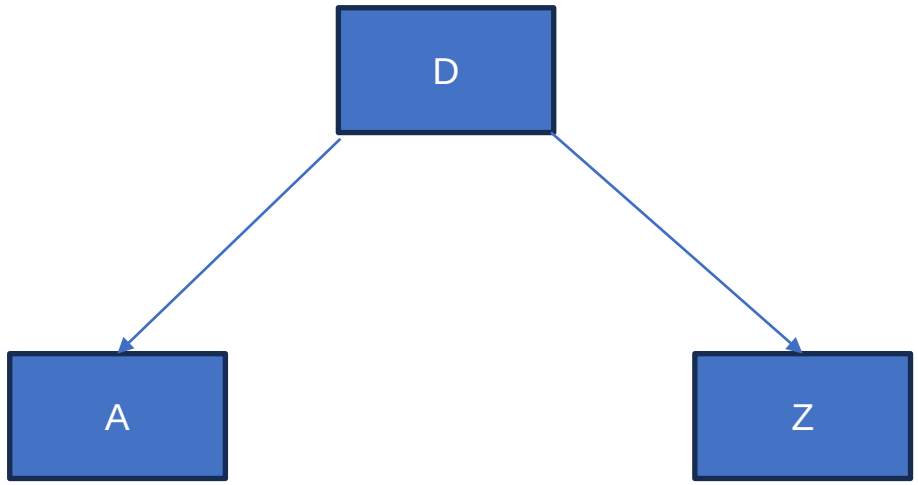
# Câu 3a:

Thêm Z

# Câu 3a:

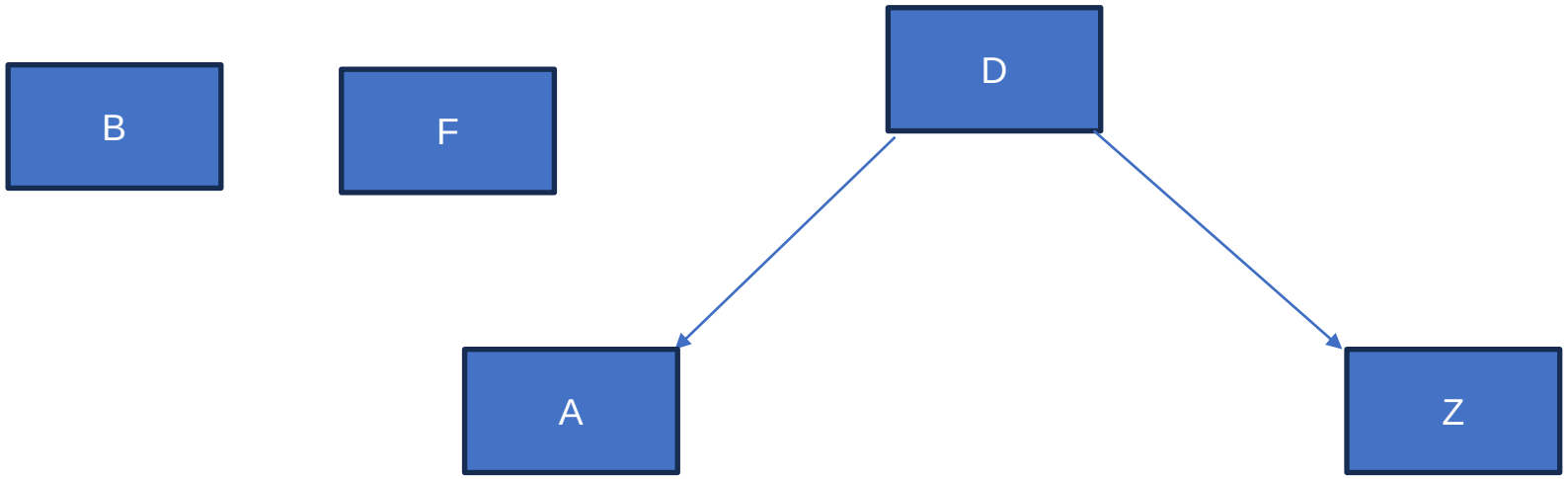
Thêm Z

(split node)



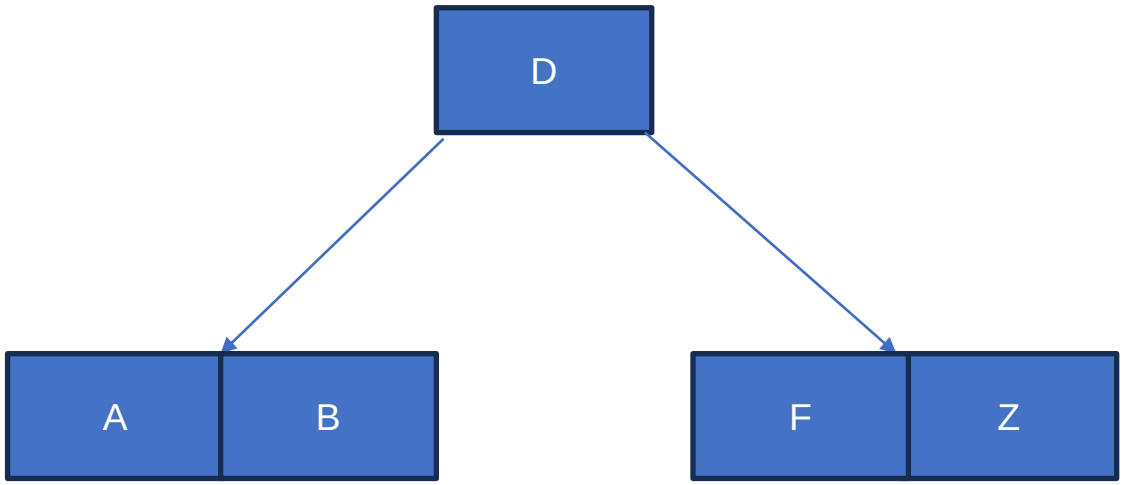
# Câu 3a:

Thêm B, F



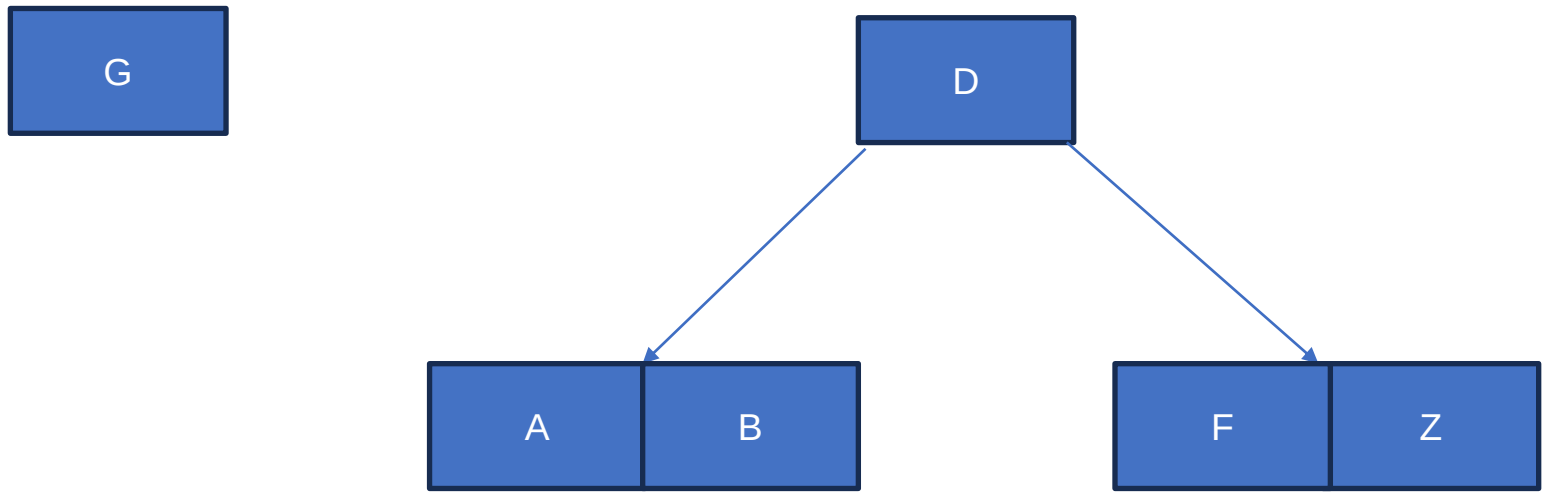
# Câu 3a:

Thêm B, F



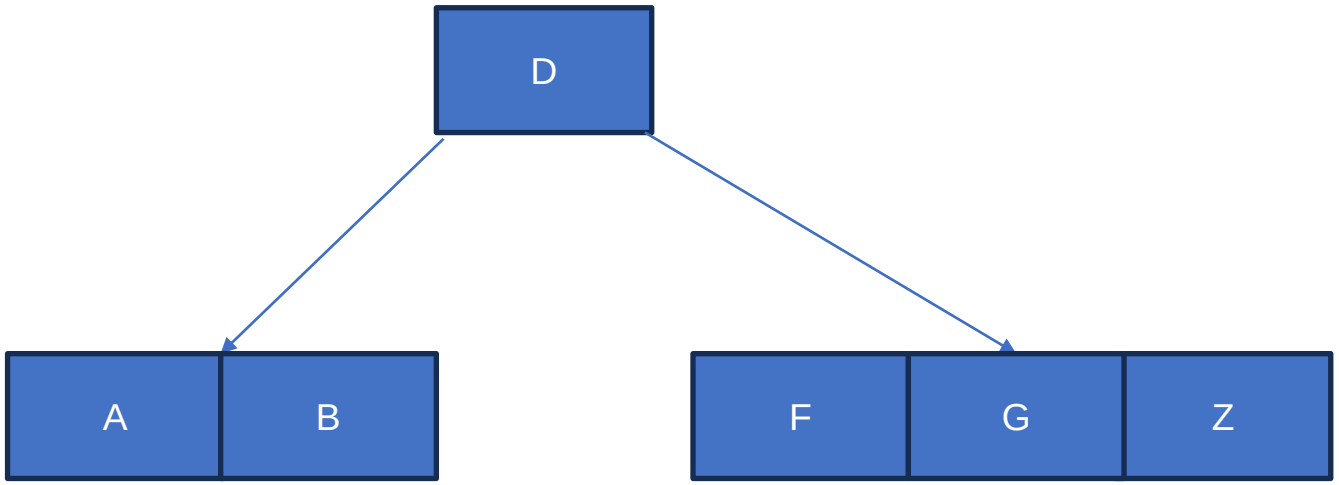
# Câu 3:

Thêm G



# Câu 3a:

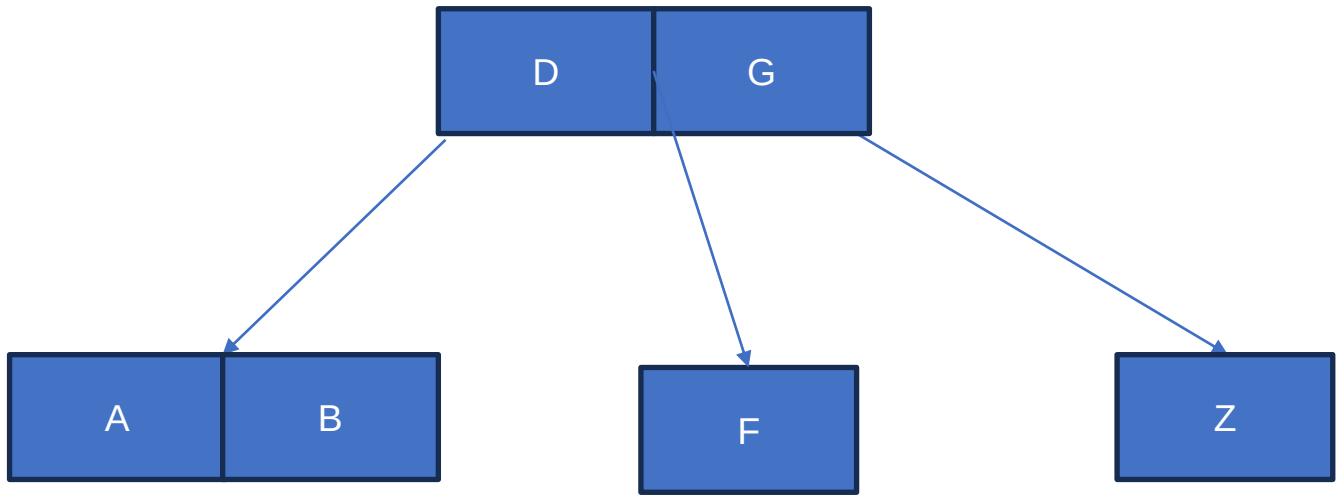
Thêm G



# Câu 3a:

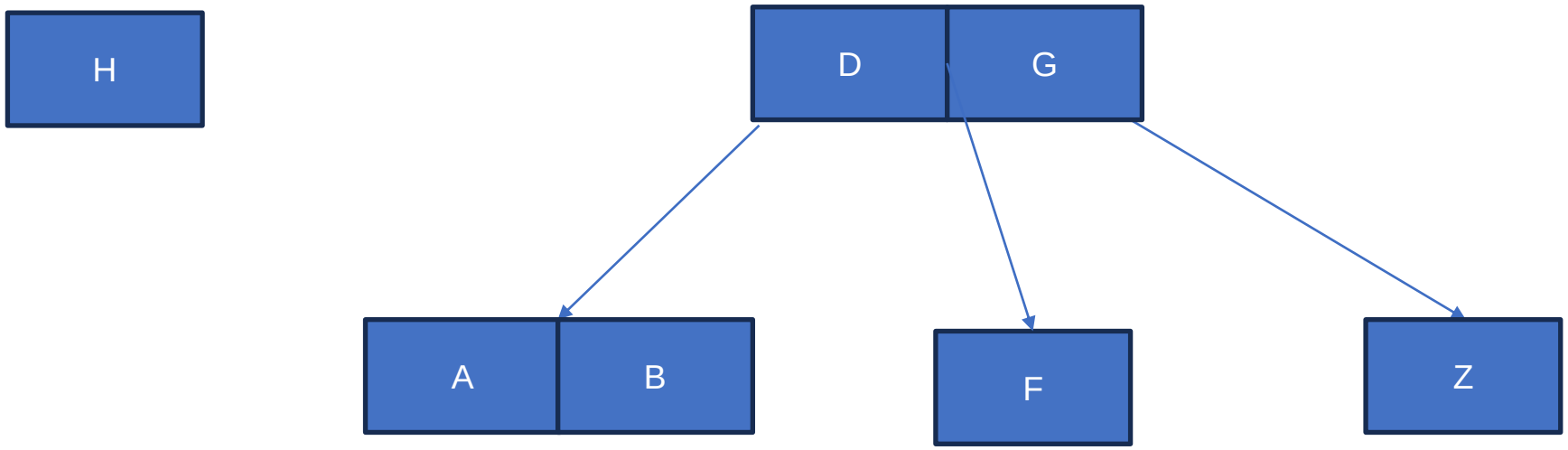
Thêm G

(split node)



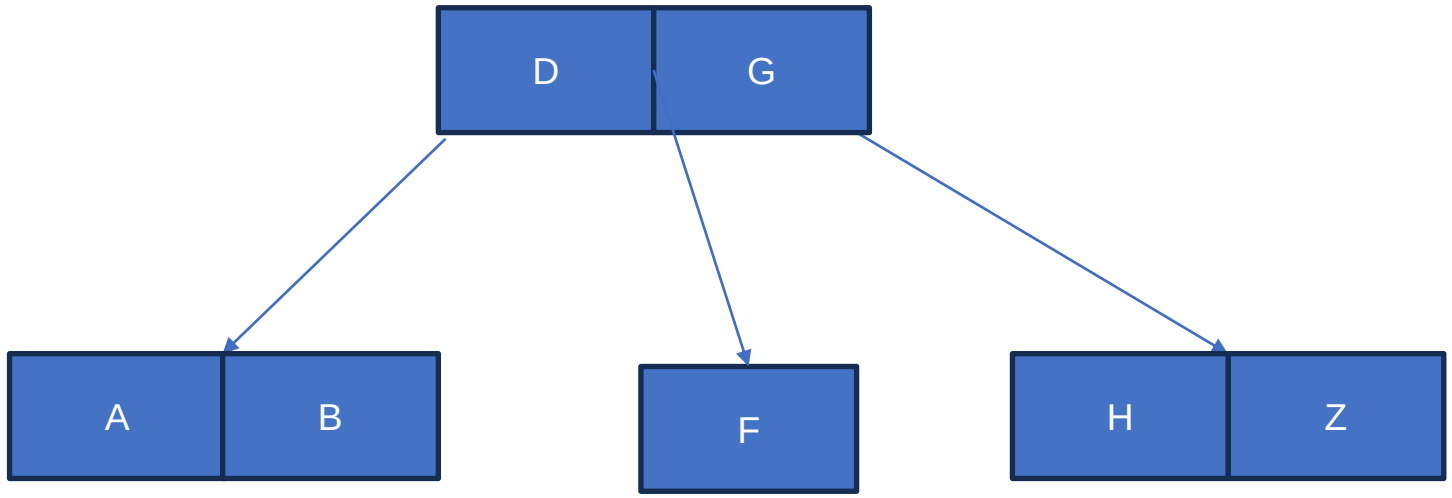
# Câu 3a:

Thêm H



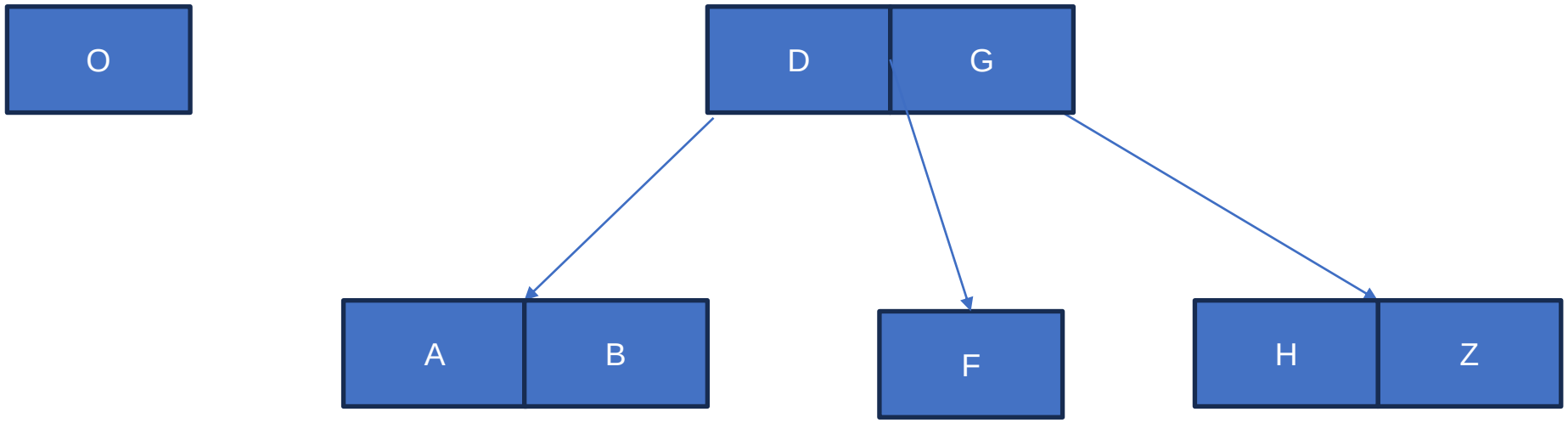
# Câu 3a:

Thêm H



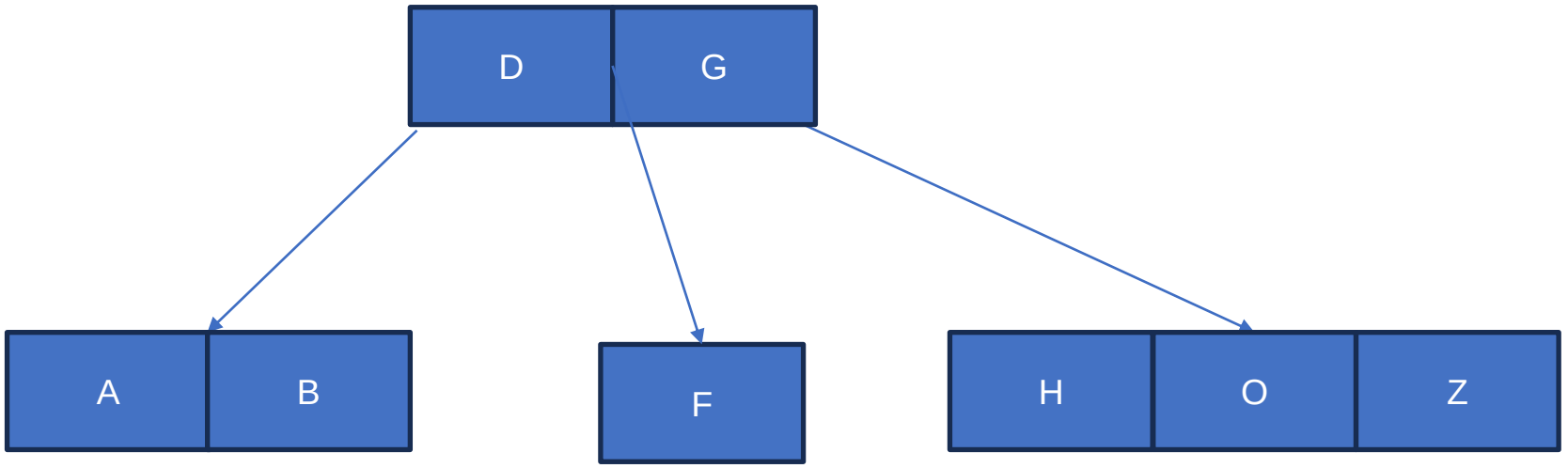
# Câu 3a:

Thêm O



# Câu 3a:

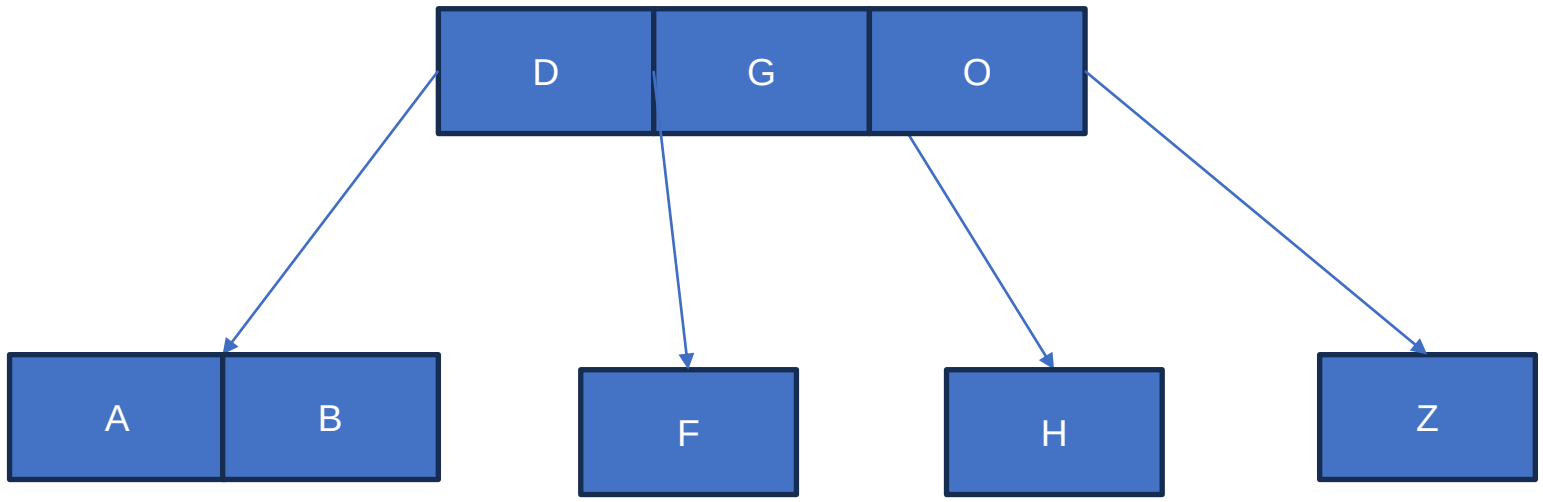
Thêm O



# Câu 3a:

Thêm O

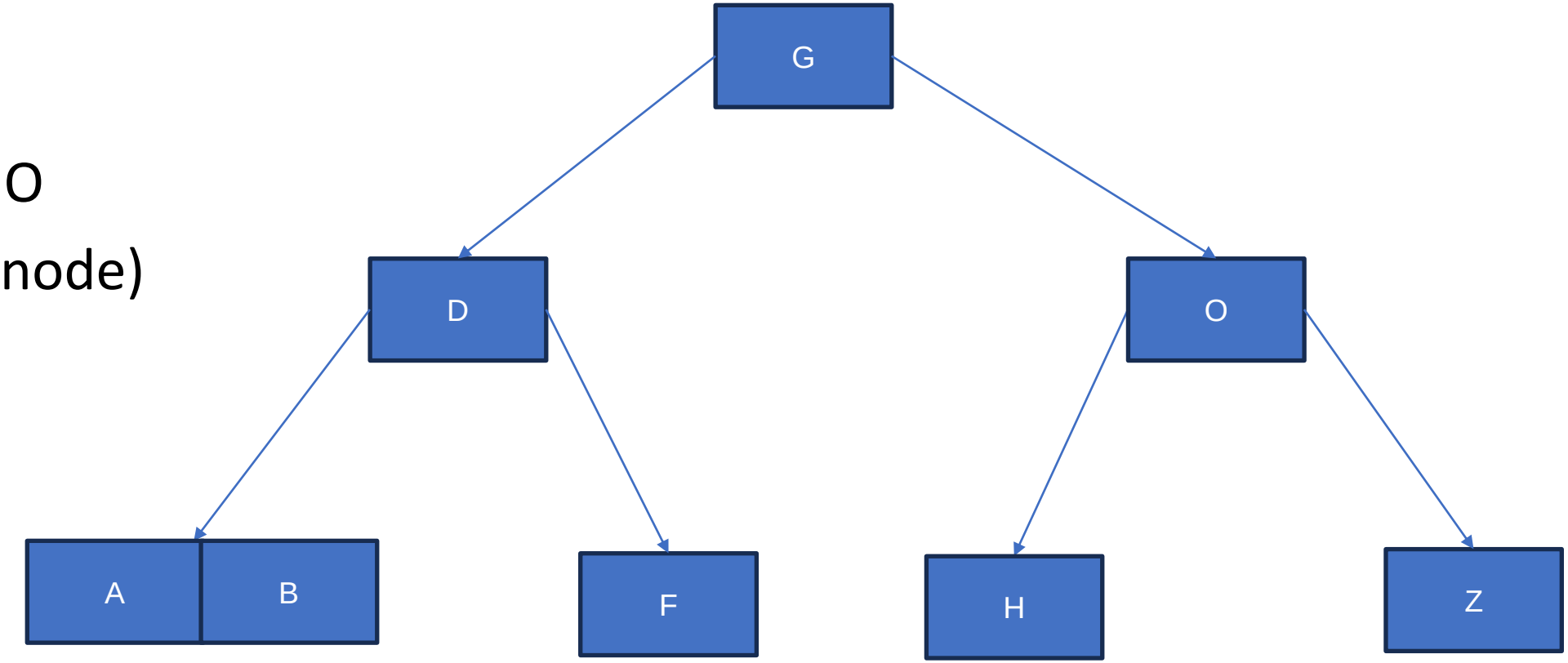
(split node)



# Câu 3:

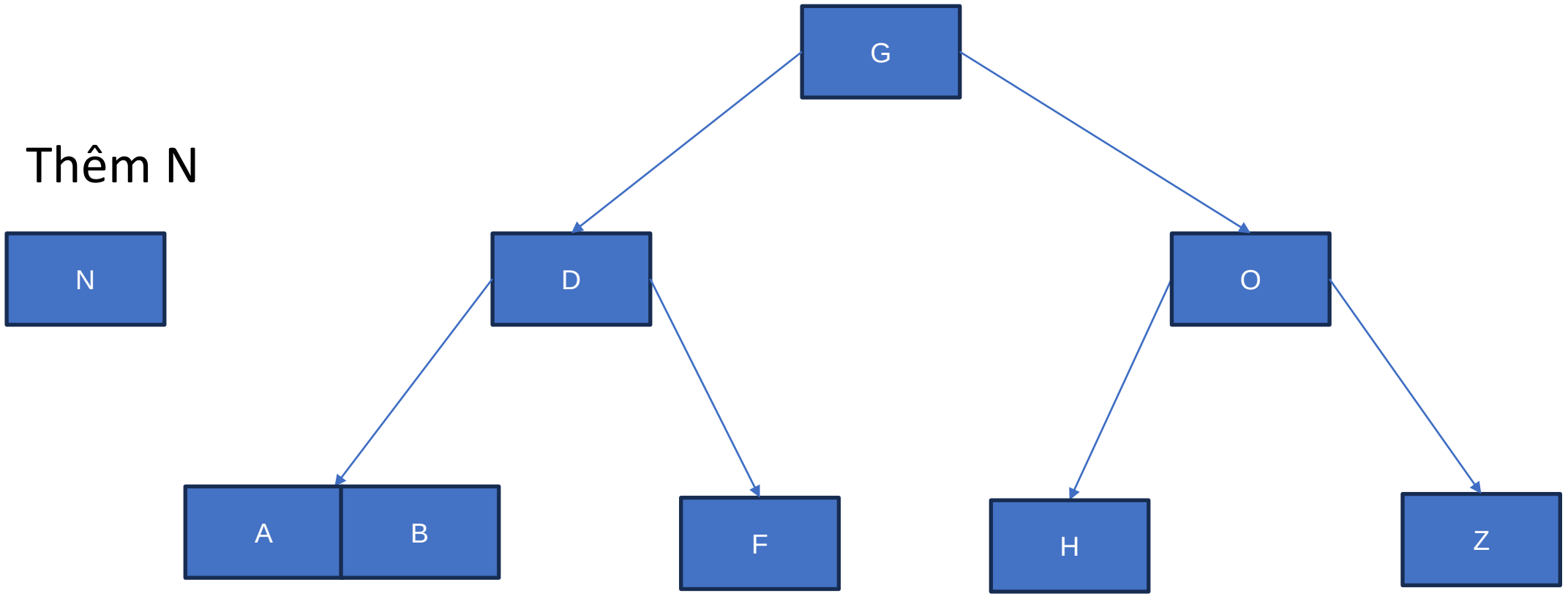
Thêm O

(split node)



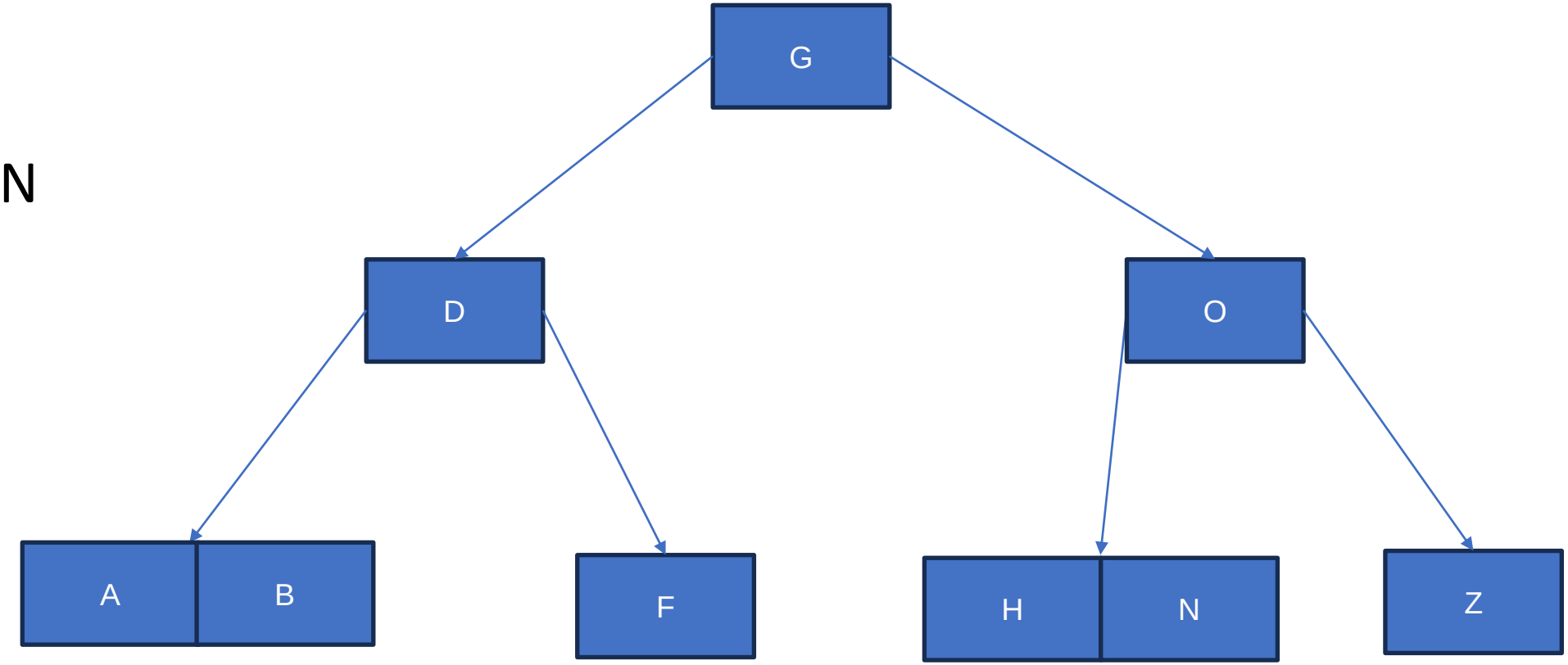
# Câu 3a:

Thêm N



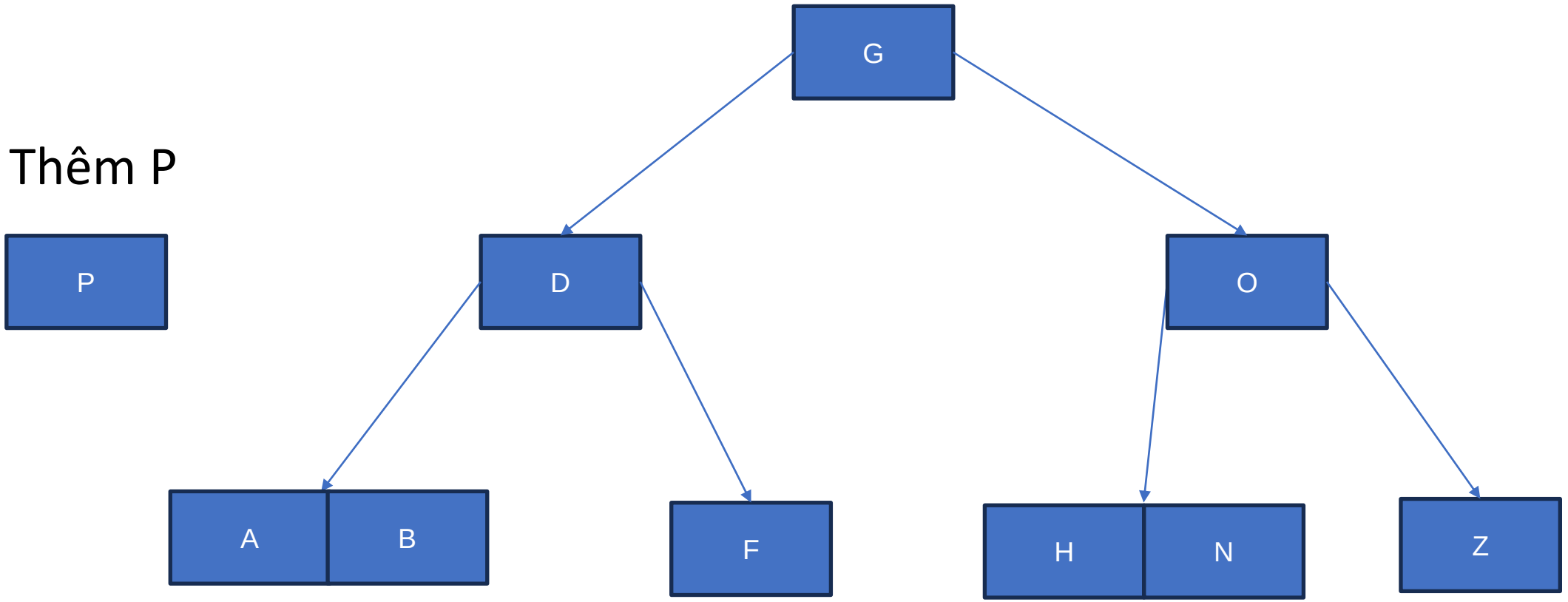
# Câu 3a:

Thêm N



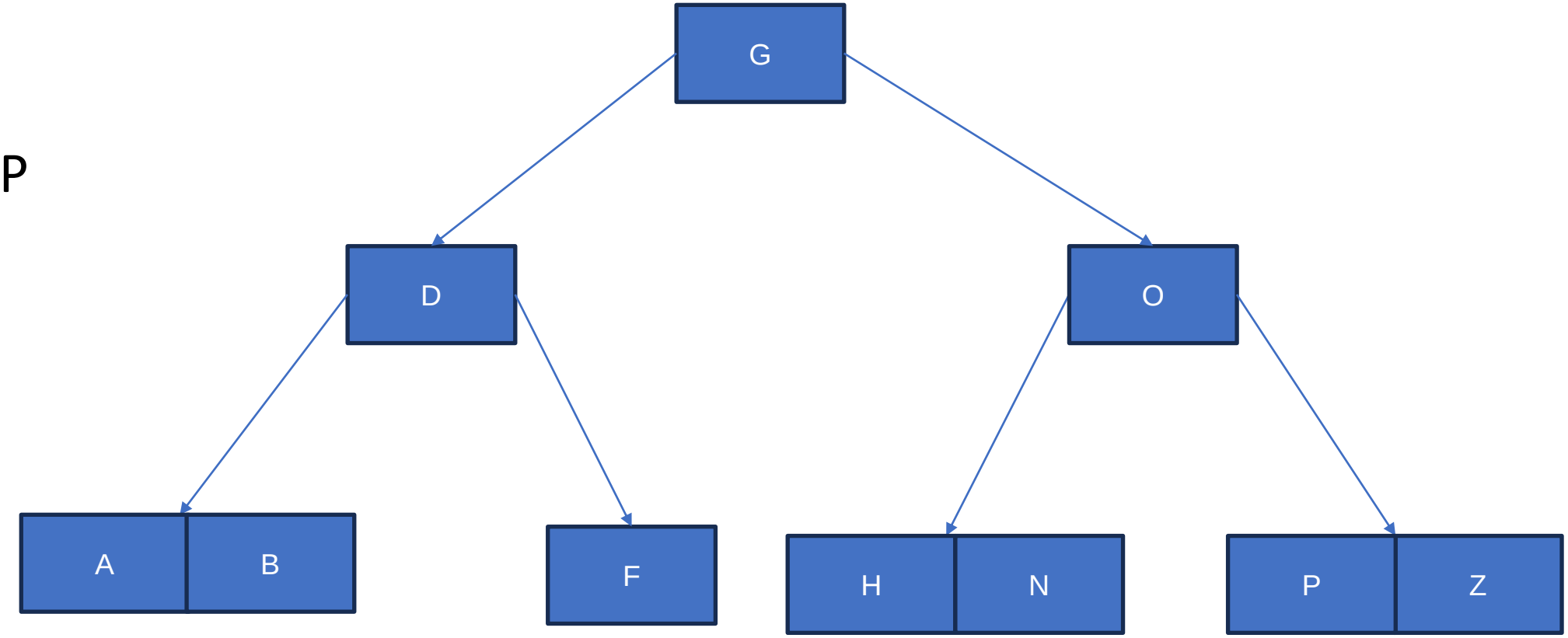
# Câu 3a:

Thêm P



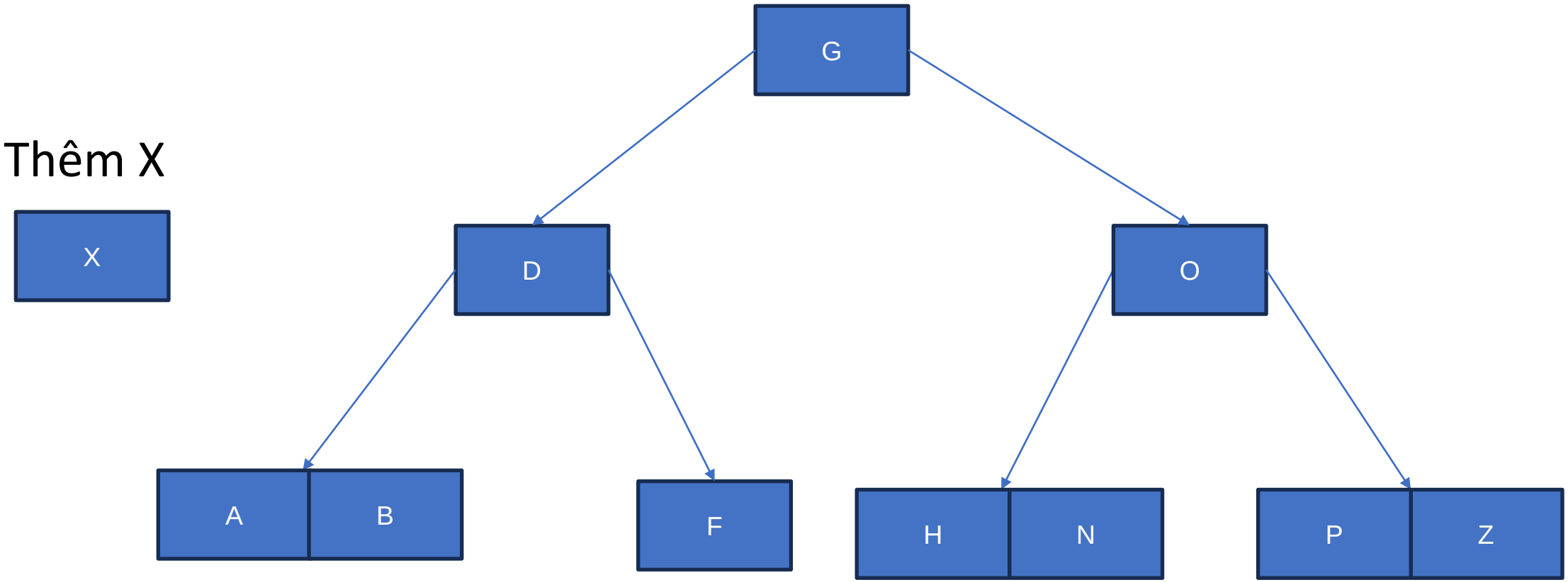
# Câu 3a:

Thêm P



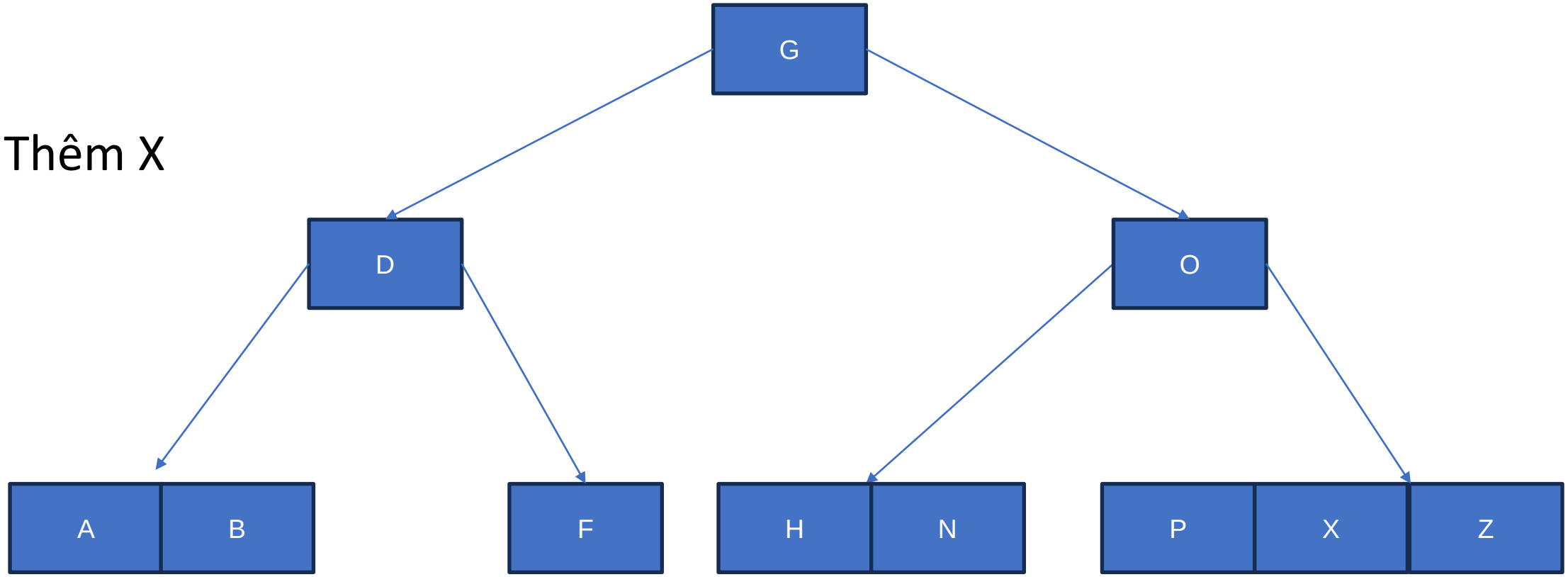
# Câu 3a:

Thêm X



# Câu 3a:

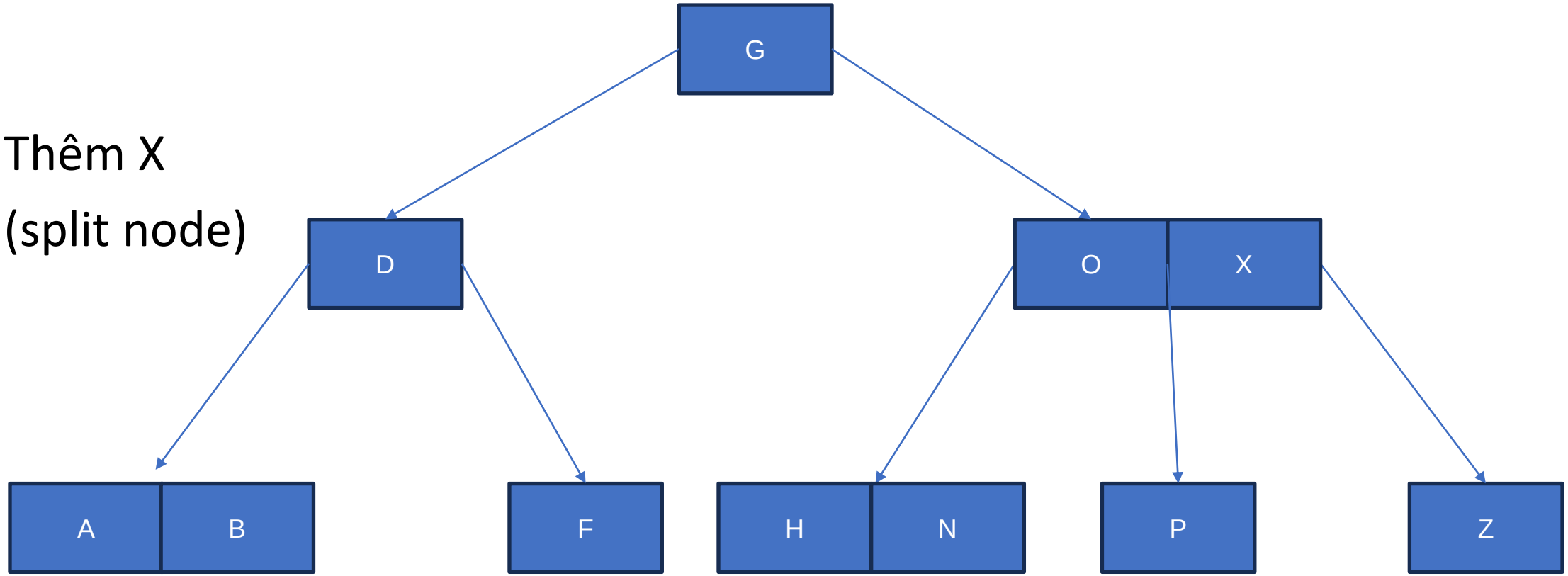
Thêm X



# Câu 3a:

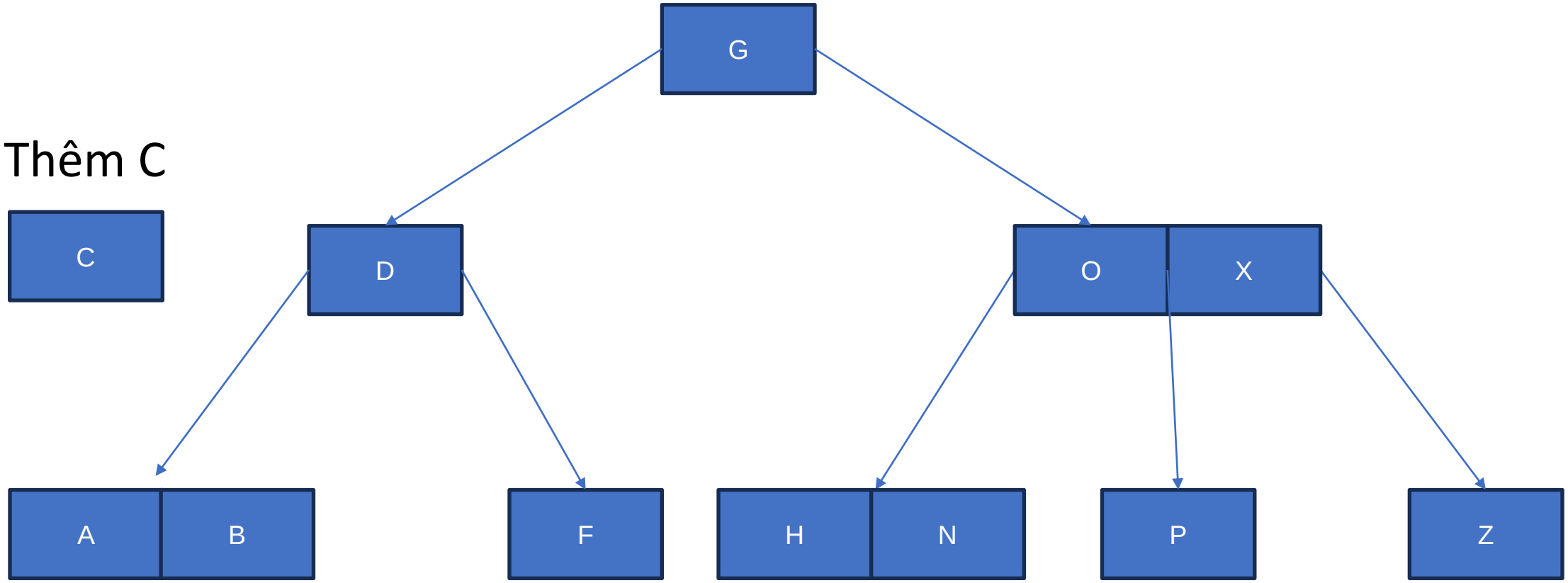
Thêm X

(split node)



# Câu 3a:

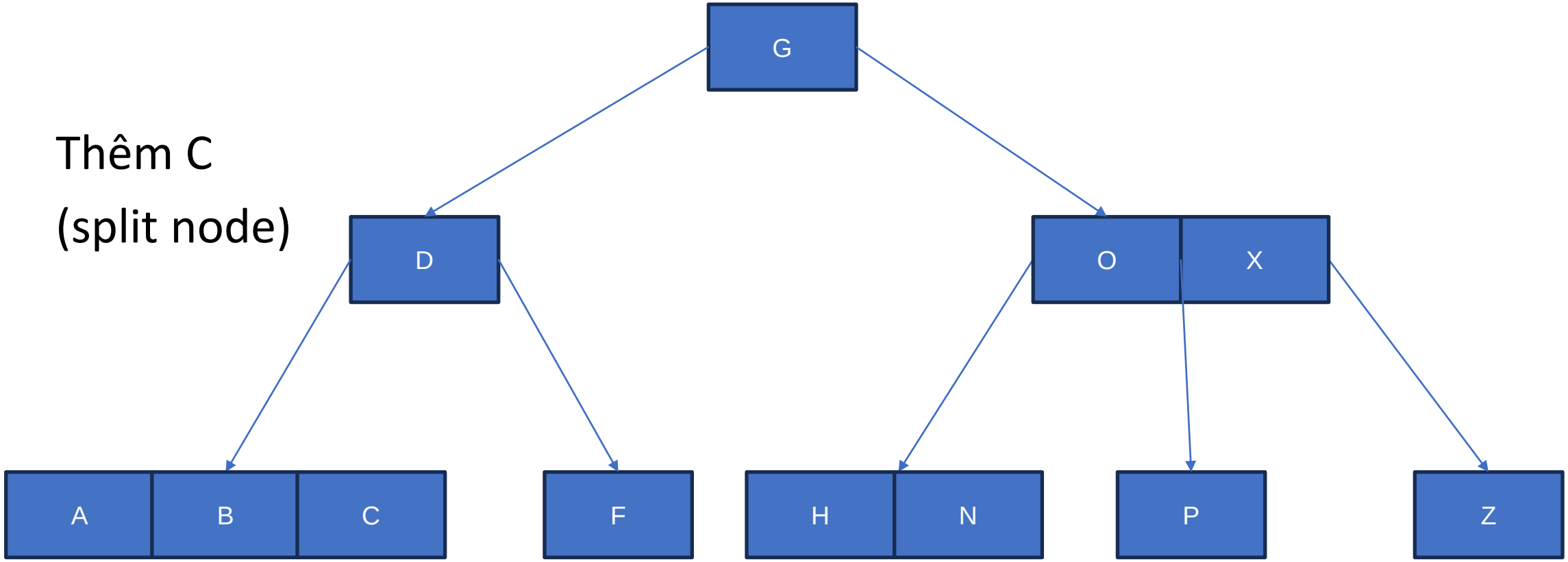
Thêm C



# Câu 3a:

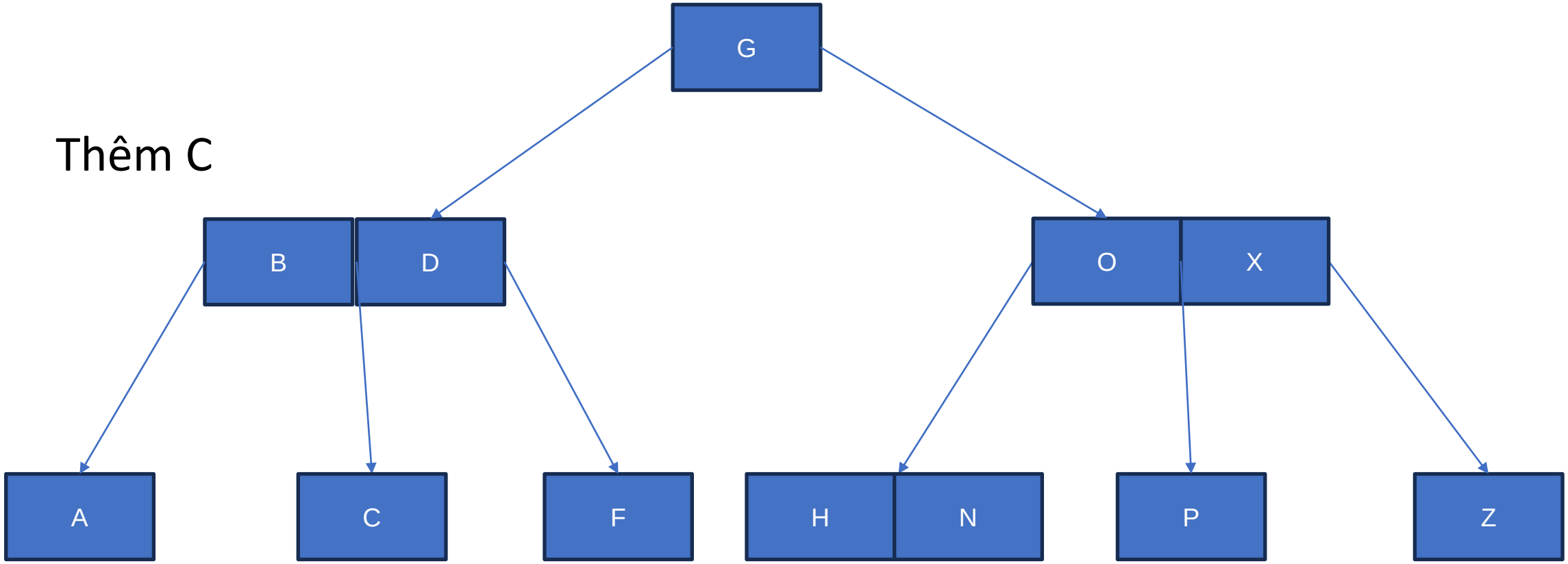
Thêm C

(split node)

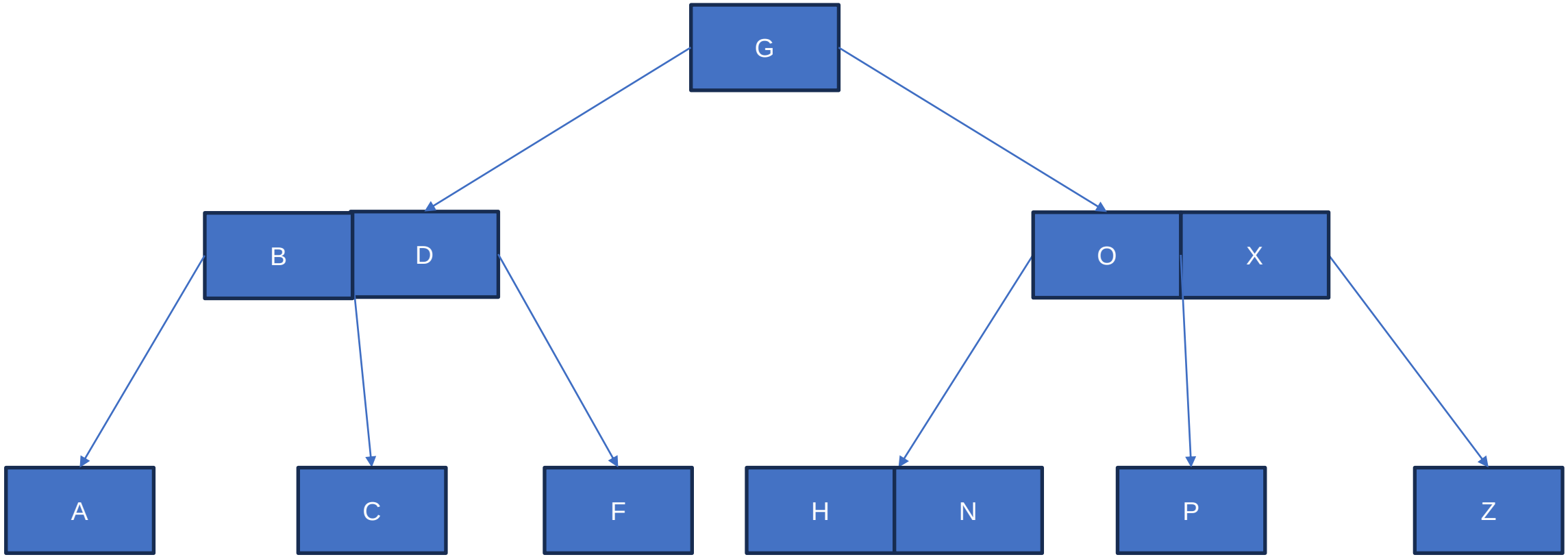


# Câu 3a:

Thêm C

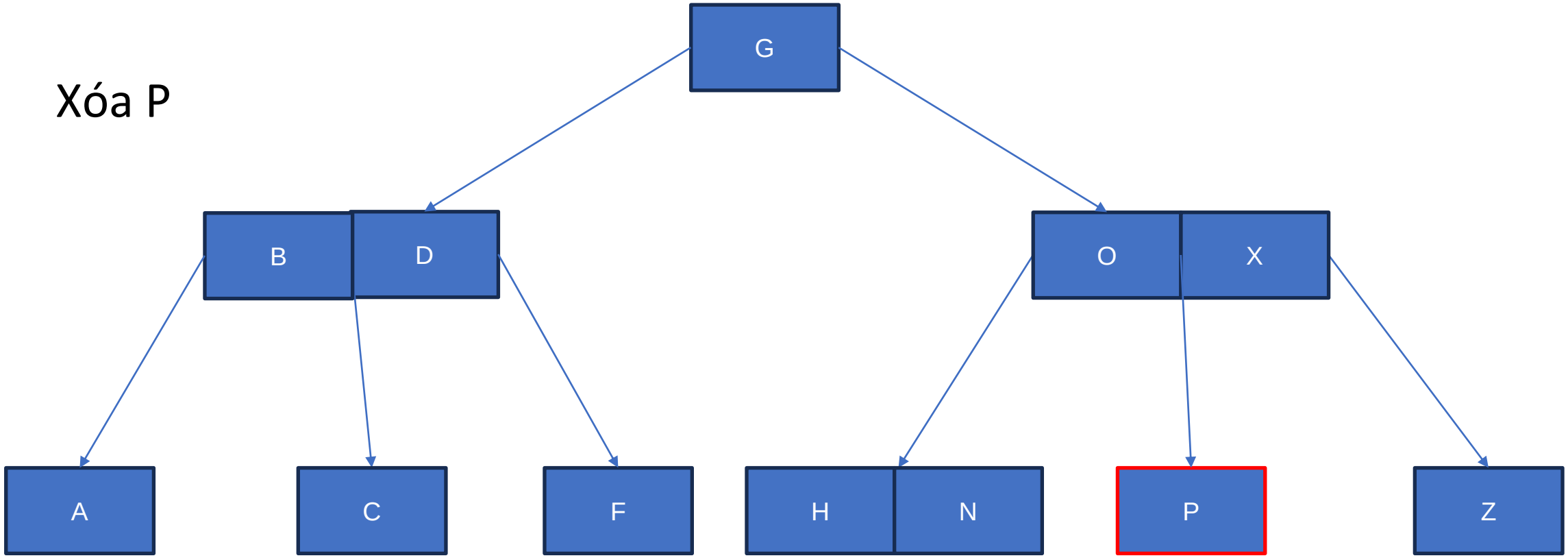


# Câu 3a:



# Câu 3b:

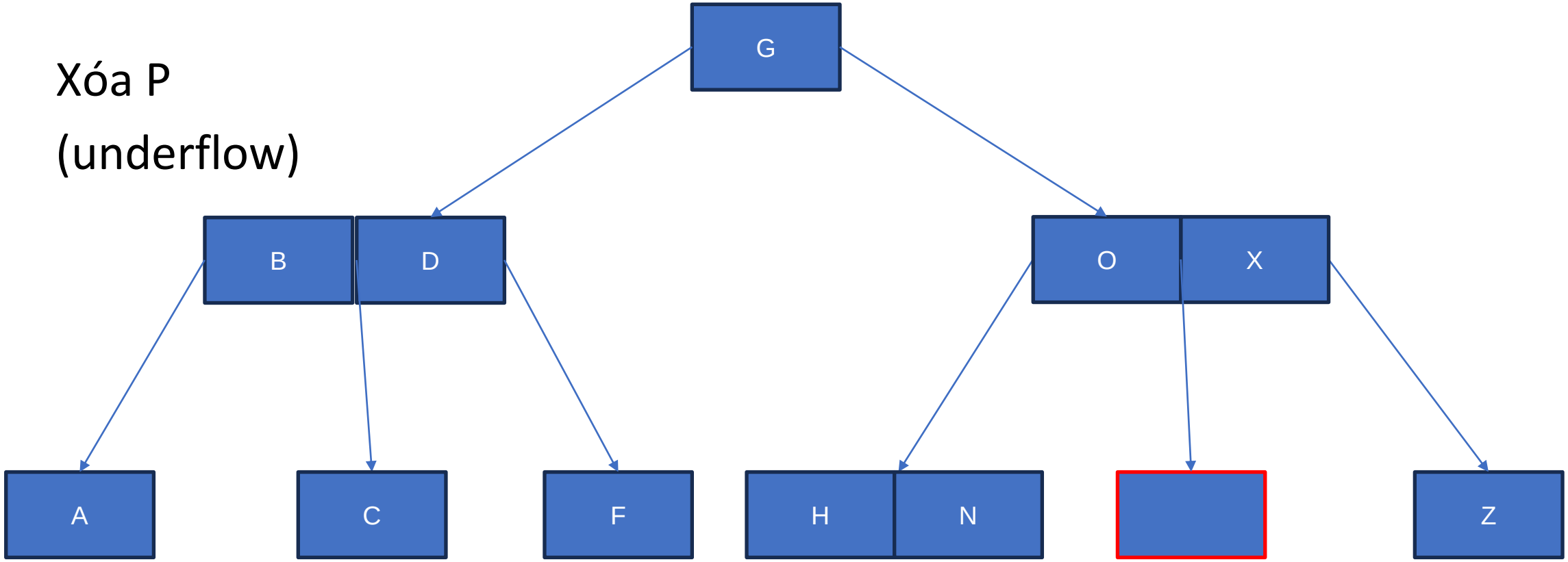
Xóa P



# Câu 3b:

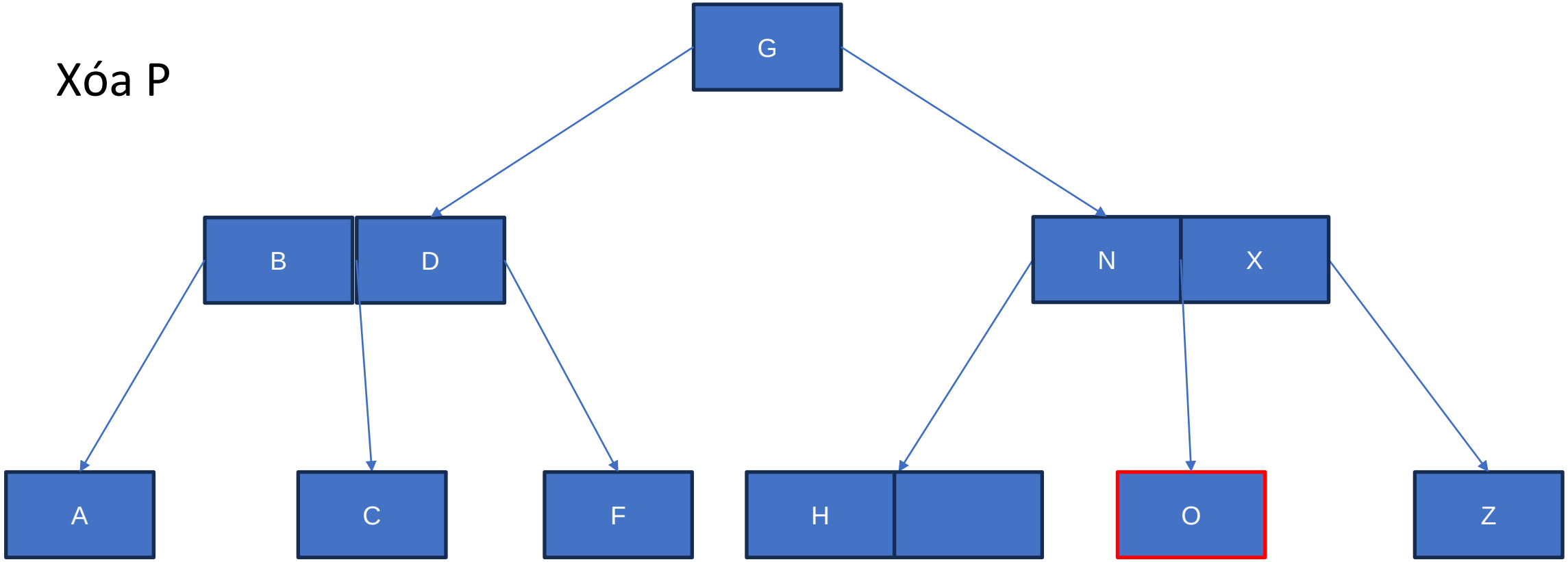
Xóa P

(underflow)



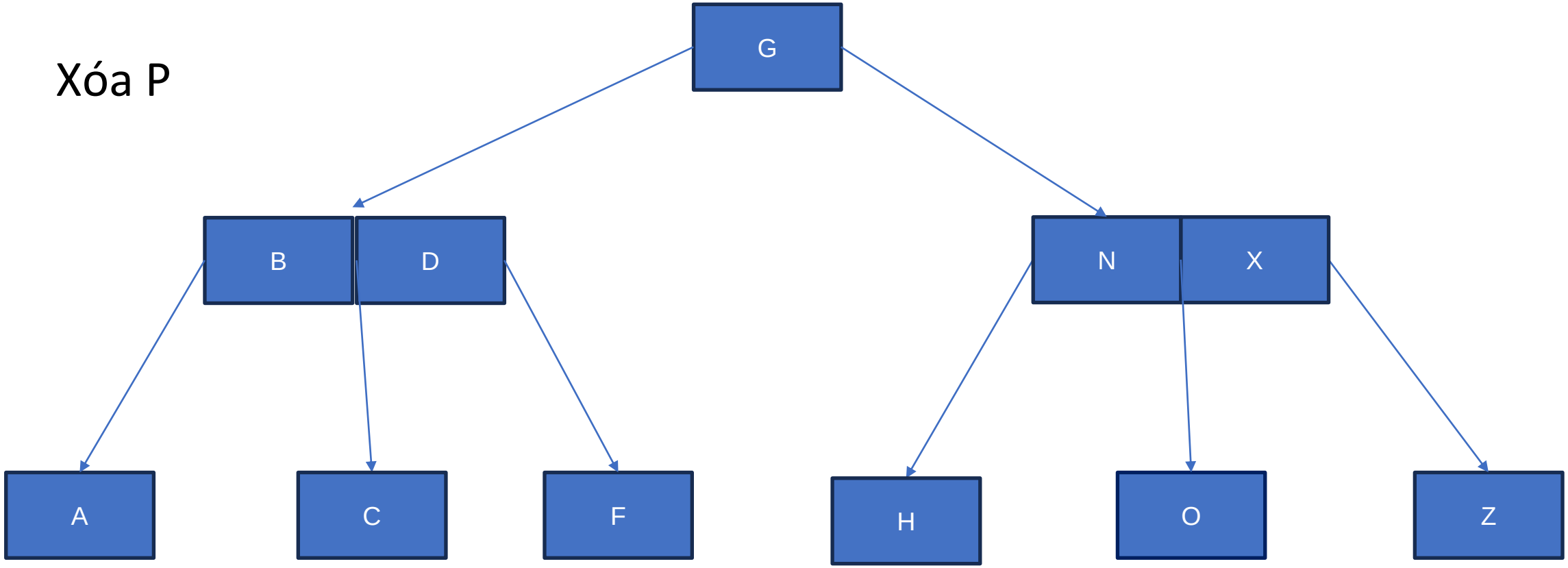
# Câu 3b:

Xóa P



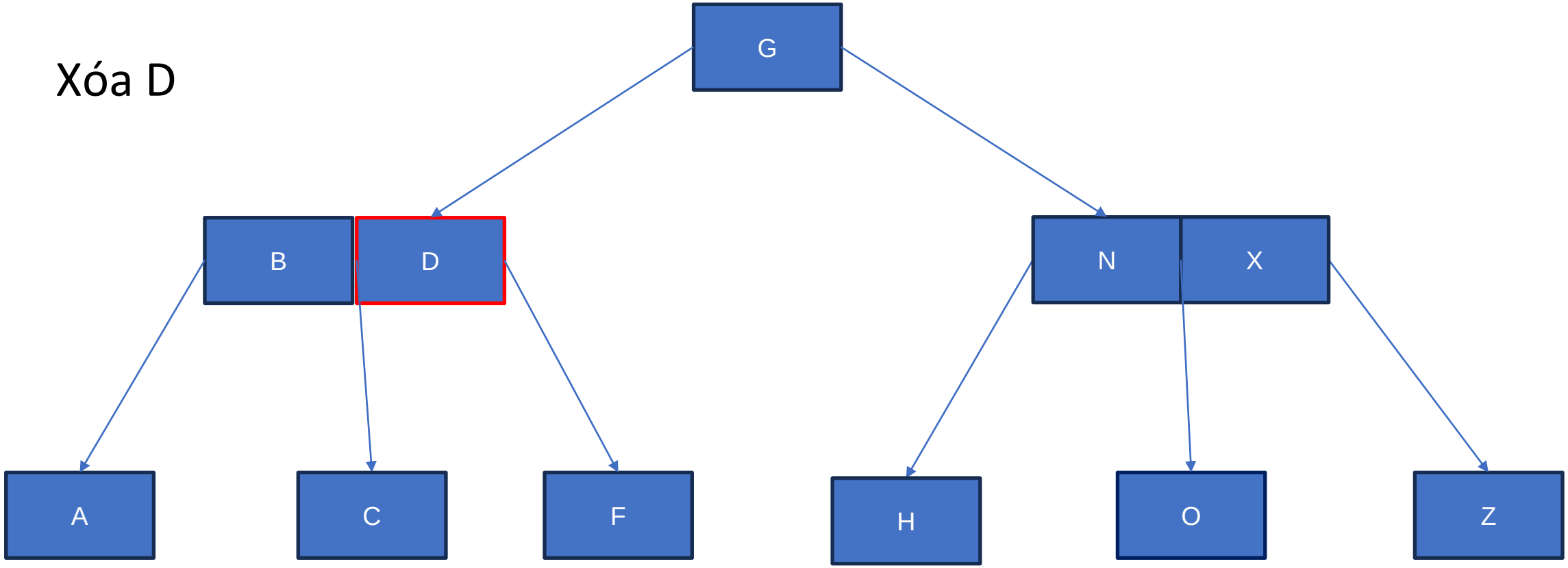
# Câu 3b:

Xóa P



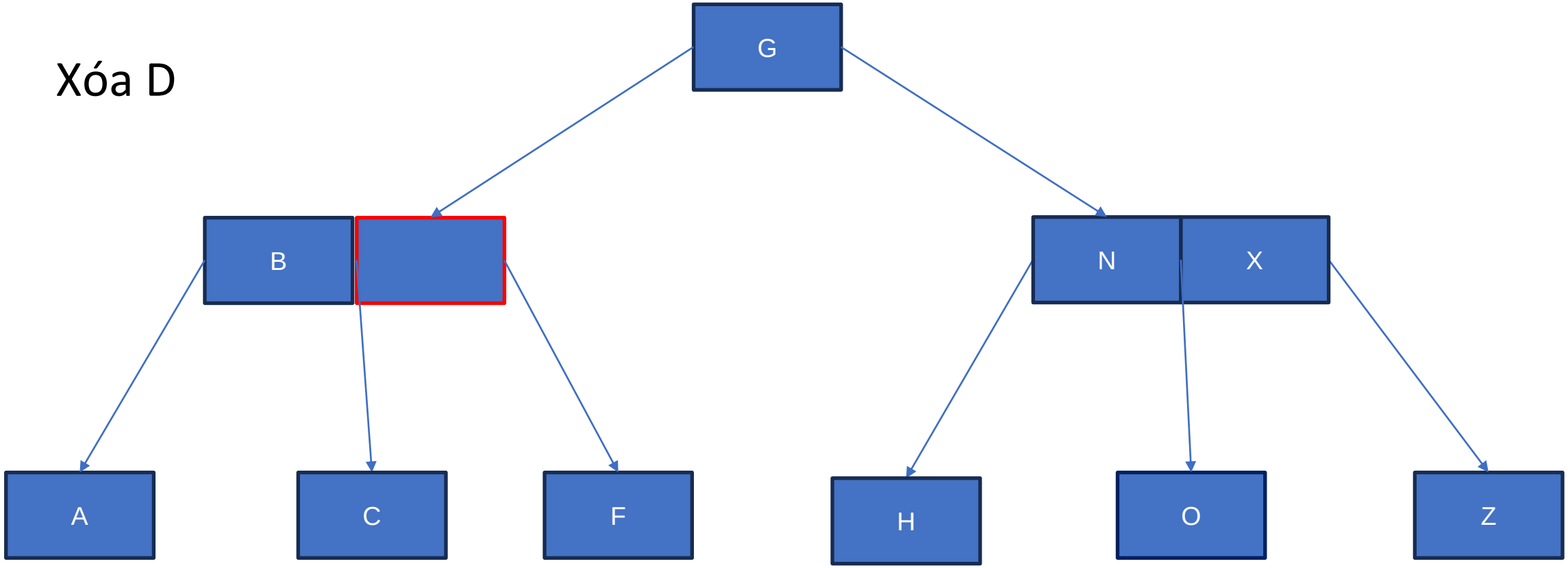
# Câu 3b:

Xóa D



# Câu 3b:

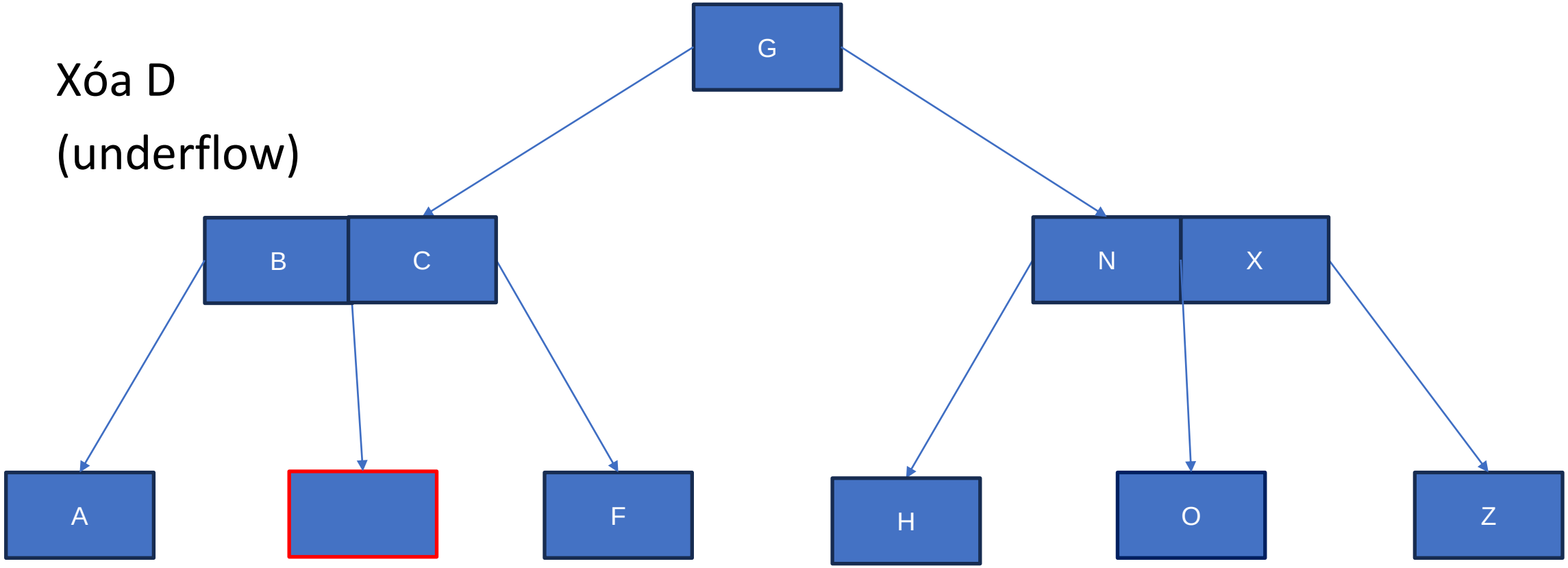
Xóa D



# Câu 3b:

Xóa D

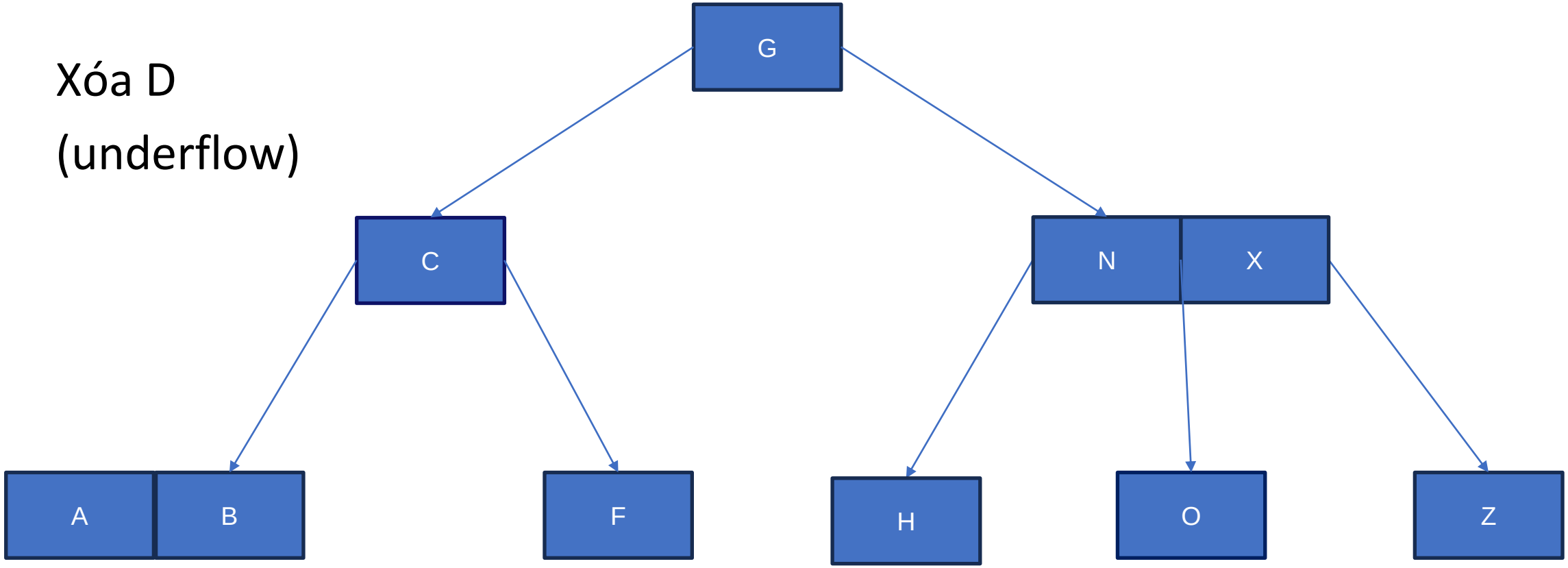
(underflow)



# Câu 3b:

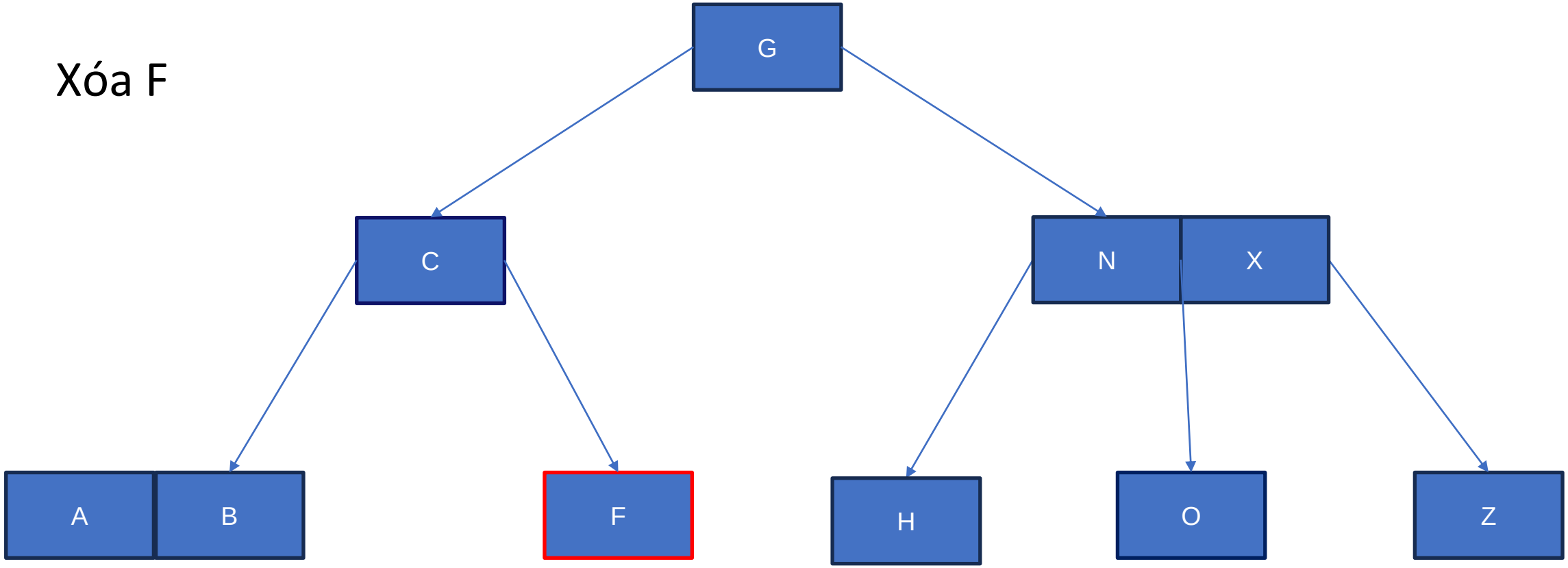
Xóa D

(underflow)



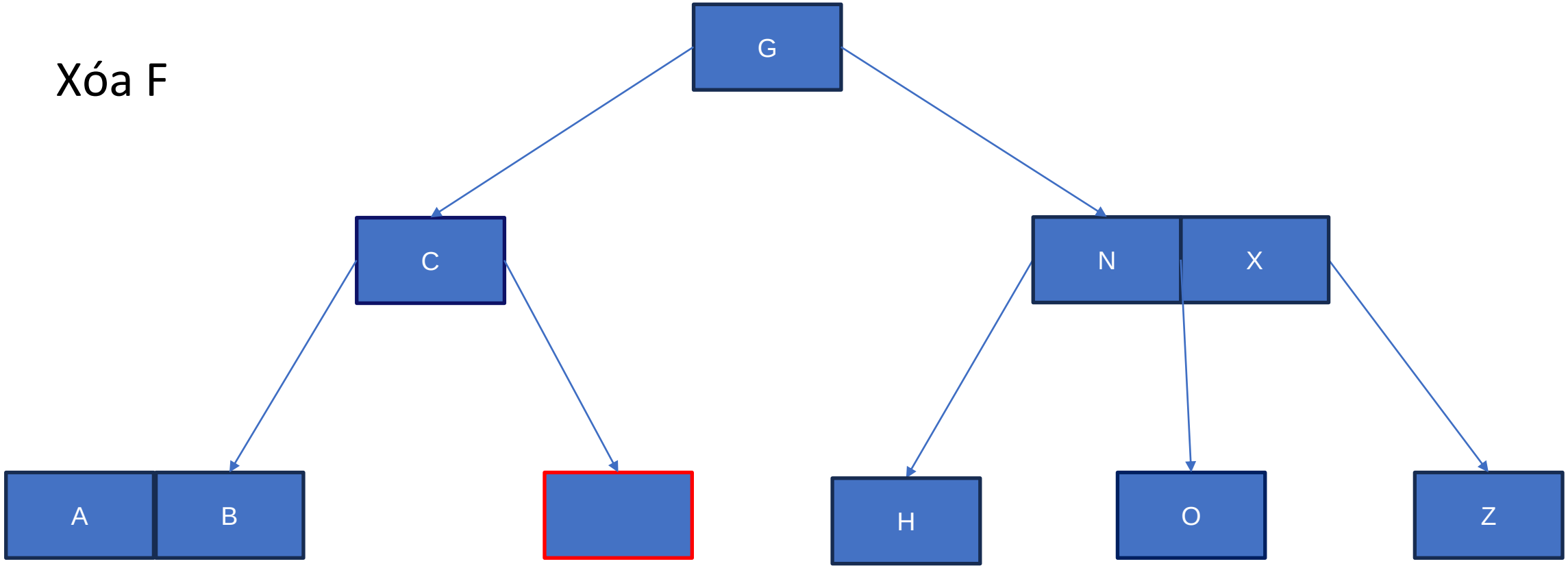
# Câu 3b:

Xóa F



# Câu 3b:

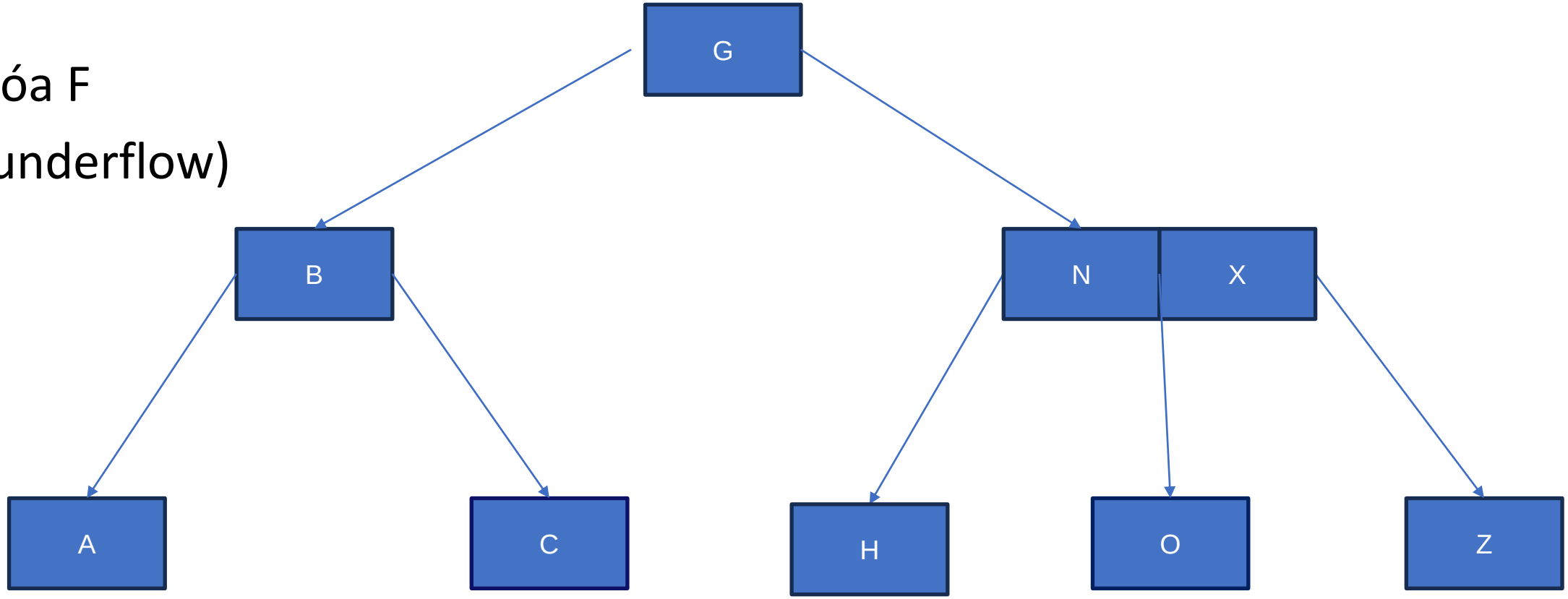
Xóa F



# Câu 3b:

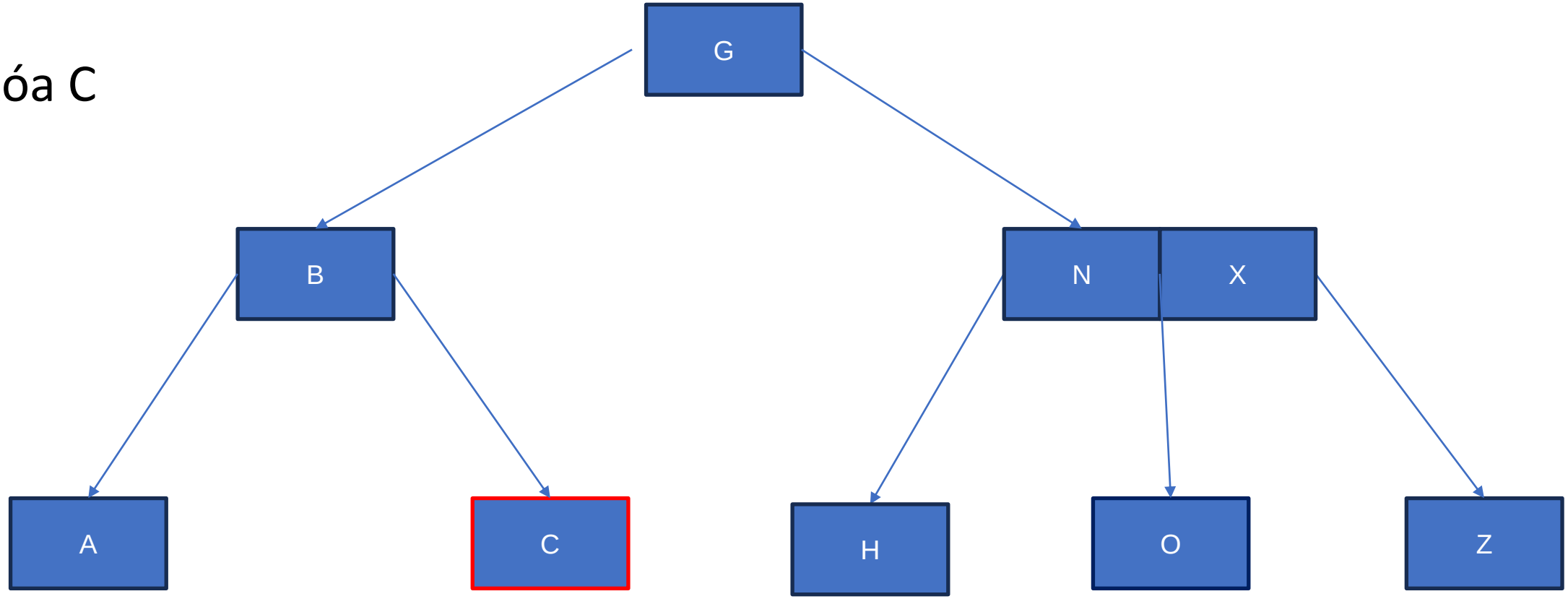
Xóa F

(underflow)



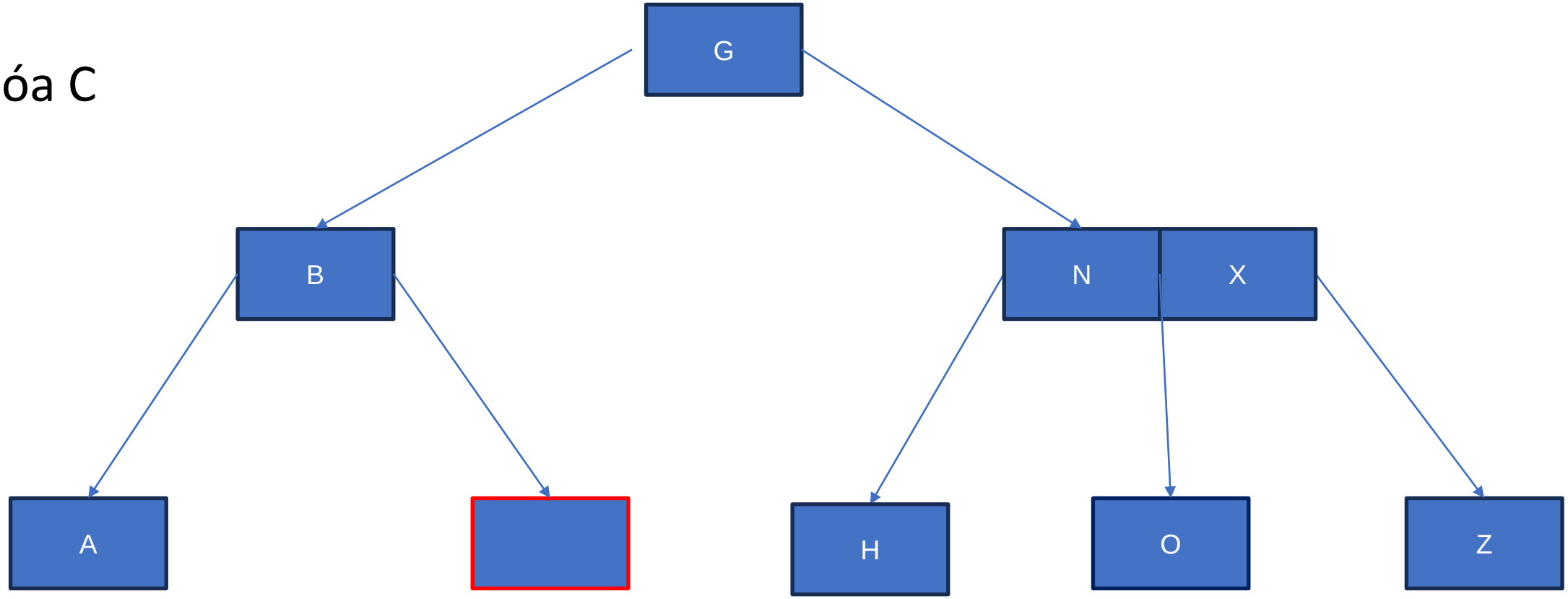
# Câu 3b:

Xóa C



# Câu 3b:

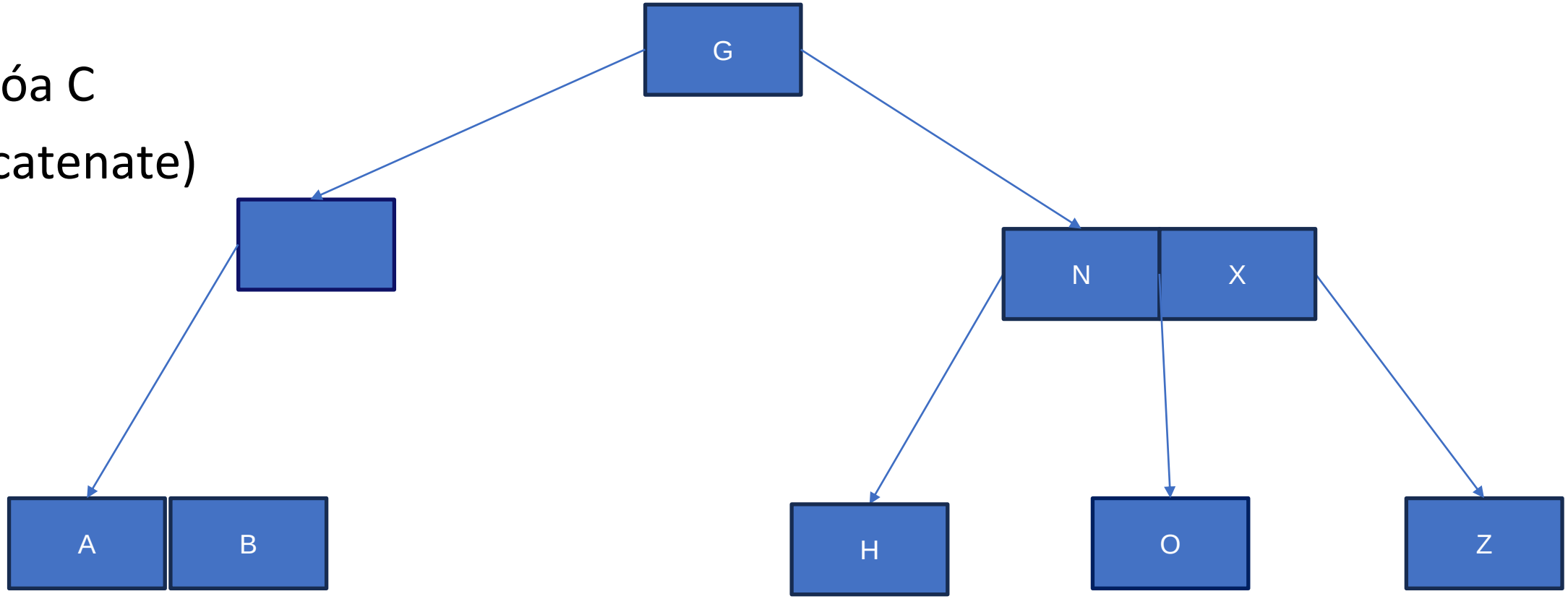
Xóa C



# Câu 3b:

Xóa C

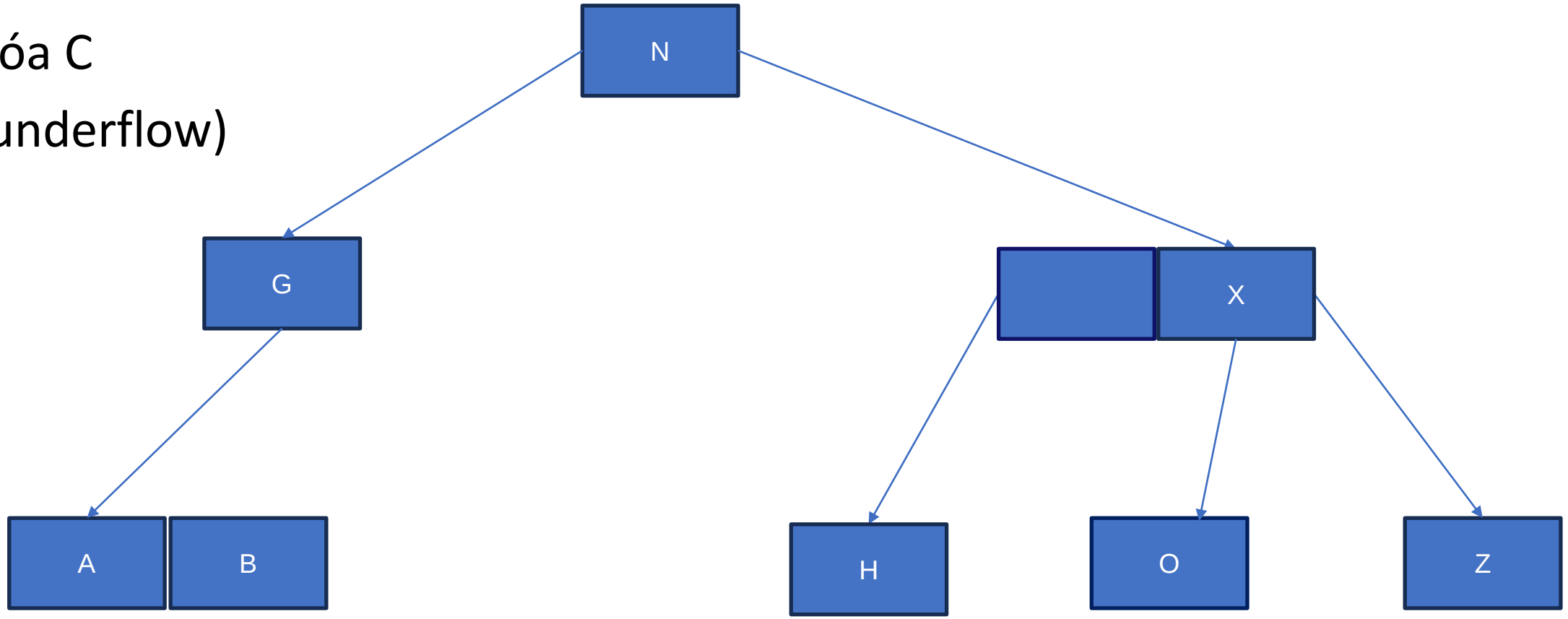
(catenate)



# Câu 3b:

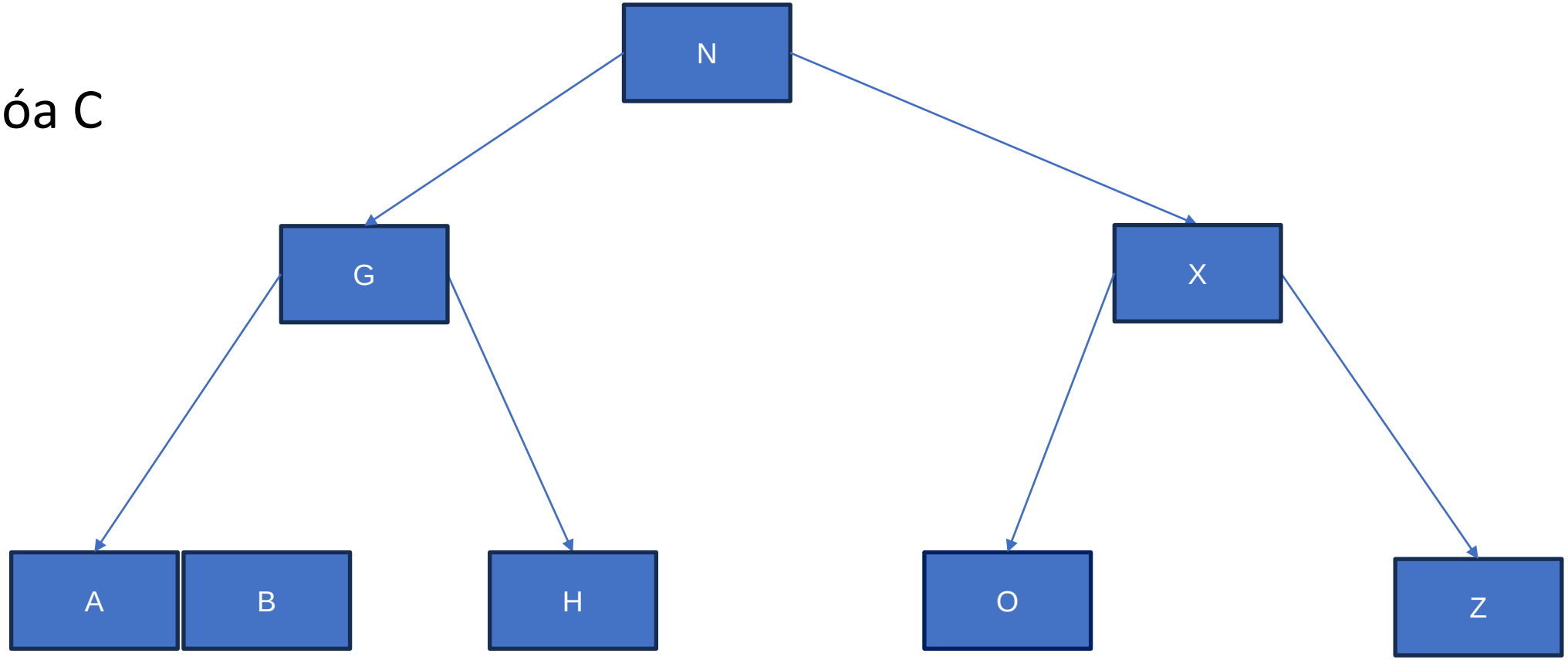
Xóa C

(underflow)

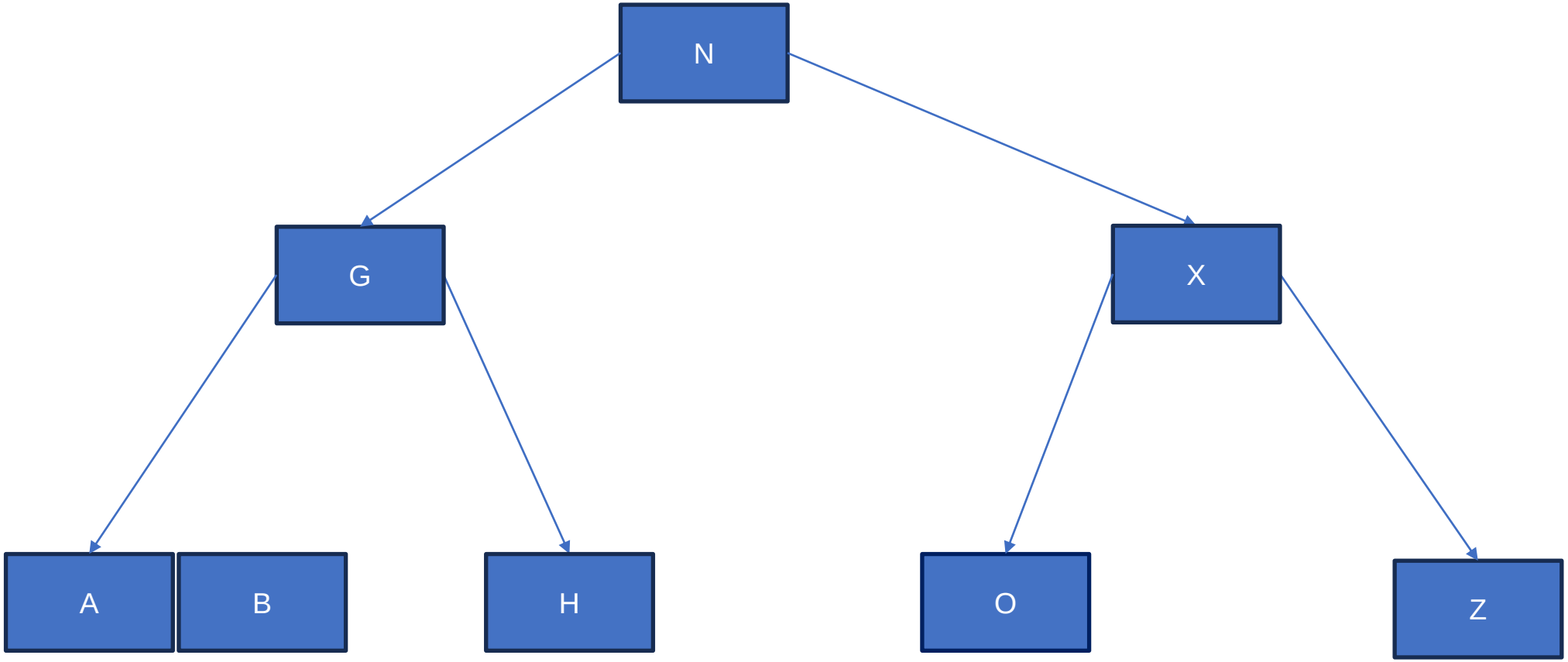


# Câu 3b:

Xóa C



# Câu 3b:



# Câu 4

Cho bảng băm gồm M = 7 ô trống nhớ và đã chứa các ô dữ liệu như hình bên dưới. Biết bảng băm có hàm băm là: h(key) = (2*key + 5) mod 7, trong đó mod là phép toán lấy dư. Bảng băm sử dụng phương pháp băm lại khi xảy ra đụng độ với hàm băm lại (hàm thăm dò): prob(key, i) = (h(key) + i*i + i) mod 7, i là số nguyên cho biết lần băm lại thứ i khi xảy ra đụng độ ở khóa key.

a,Trình bày từng bước và vẽ hình bảng băm khi thêm lần lượt các giá trị khóa (key) sau: 14, 12, 31.

b, Trình bày từng bước khi tìm giá trị khóa 42 trong bảng băm khi đã hoàn thành yêu cầu ở câu a.

| Chỉ số | Khóa |
|---|---|
| 0 | |
| 1 | 26 |
| 2 | |
| 3 | |
| 4 | 17 |
| 5 | |
| 6 | 39 |

# Câu 4a

Thêm **14**, 12, 31

Hàm băm : h(key) = (2*key + 5) mod 7

Hàm băm lại: prob(key, i) = (h(key) + i*i + i) mod 7

**Khi thêm 14**

h(14) = (2.14 + 5) mod 7 = 5

Vị trí 5 đang trống, thêm vào vị trí 5

| **Chỉ số** | **Khóa** |
|---|---|
| **0** | |
| **1** | 26 |
| **2** | |
| **3** | |
| **4** | 17 |
| **5** | |
| **6** | 39 |

# Câu 4a

Thêm **14**, 12, 31

Hàm băm : h(key) = (2*key + 5) mod 7

Hàm băm lại: prob(key, i) = (h(key) + i*i + i) mod 7

**Khi thêm 14**

h(14) = (2.14 + 5) mod 7 = 5

Vị trí 5 đang trống, thêm vào vị trí 5

| Chỉ số | Khóa |
|---|---|
| **0** | |
| **1** | 26 |
| **2** | |
| **3** | |
| **4** | 17 |
| **5** | 14 |
| **6** | 39 |

# Câu 4a

Thêm 14, **12**, 31

Hàm băm : h(key) = (2*key + 5) mod 7

Hàm băm lại: prob(key, i) = (h(key) + i*i + i) mod 7

**Khi thêm 12**

h(12) = (2.12 + 5) mod 7 = 1

Vị trí 1 xảy ra đụng độ → băm lại.

Băm lại lần 1:

prob(12, 1) = (1 + 1.1 + 1) mod 7 = 3

Vị trí 3 trống, thêm vào vị trí 3.

| Chỉ số | Khóa |
|---|---|
| 0 | |
| 1 | 26 |
| 2 | |
| 3 | |
| 4 | 17 |
| 5 | 14 |
| 6 | 39 |

# Câu 4a

Thêm 14, **12**, 31

Hàm băm : h(key) = (2*key + 5) mod 7

Hàm băm lại: prob(key, i) = (h(key) + i*i + i) mod 7

**Khi thêm 12**

h(12) = (2.12 + 5) mod 7 = 1

Vị trí 1 xảy ra đụng độ → băm lại.

Băm lại lần 1:

prob(12, 1) = (1 + 1.1 + 1) mod 7 = 3

Vị trí 3 trống, thêm vào vị trí 3.

| Chỉ số | Khóa |
|---|---|
| 0 | |
| 1 | 26 |
| 2 | |
| 3 | 12 |
| 4 | 17 |
| 5 | 14 |
| 6 | 39 |

# Câu 4a

Thêm 14, 12, **31**

Hàm băm : h(key) = (2*key + 5) mod 7

Hàm băm lại: prob(key, i) = (h(key) + i*i + i) mod 7

**Khi thêm 31**

h(31) = (2.31 + 5) mod 7 = 4.

Vị trí 4 xảy ra đụng độ →băm lại

Băm lại lần 1:

prob(31, 1) = (4 + 1.1 + 1) mod 7 = 6.

Vị trí 6 xảy ra đụng độ → băm lại.

| Chỉ số | Khóa |
|---|---|
| **0** | |
| **1** | 26 |
| **2** | |
| **3** | 12 |
| **4** | 17 |
| **5** | 14 |
| **6** | 39 |

# Câu 4a

Thêm 14, 12, **31**

Hàm băm : h(key) = (2*key + 5) mod 7

Hàm băm lại: prob(key, i) = (h(key) + i*i + i) mod 7

**Khi thêm 31**

Băm lại lần 2:

prob(31, 2) = (4 + 2.2 + 2) mod 7 = 3.

Vị trí 3 xảy ra đụng độ → băm lại.

| **Chỉ số** | **Khóa** |
|---|---|
| **0** | |
| **1** | 26 |
| **2** | |
| **3** | 12 |
| **4** | 17 |
| **5** | 14 |
| **6** | 39 |

# Câu 4a

Thêm 14, 12, **31**

Hàm băm : h(key) = (2*key + 5) mod 7

Hàm băm lại: prob(key, i) = (h(key) + i*i + i) mod 7

**Khi thêm 31**

Băm lại lần 3:

prob(31, 3) = (4 + 3.3 + 3) mod 7 = 2.

Vị trí 2 trống, thêm vào vị trí 2.

| **Chỉ số** | **Khóa** |
|---|---|
| **0** | |
| **1** | 26 |
| **2** | |
| **3** | 12 |
| **4** | 17 |
| **5** | 14 |
| **6** | 39 |

# Câu 4a

Thêm 14, 12, **31**

Hàm băm : h(key) = (2*key + 5) mod 7

Hàm băm lại: prob(key, i) = (h(key) + i*i + i) mod 7

**Khi thêm 31**

| **Chỉ số** | **Khóa** |
|---|---|
| **0** | |
| **1** | 26 |
| **2** | 31 |
| **3** | 12 |
| **4** | 17 |
| **5** | 14 |
| **6** | 39 |

# Câu 4a

Thêm 14, 12, 31

Hàm băm :

h(key) = (2*key + 5) mod 7

Hàm băm lại:

prob(key, i) = (h(key) + i*i + i) mod 7

| Chỉ số | Khóa |
|---|---|
| 0 | |
| 1 | 26 |
| 2 | 31 |
| 3 | 12 |
| 4 | 17 |
| 5 | 14 |
| 6 | 39 |

# Câu 4b

Tìm khóa **42**

h(42) = (2.42 + 5) mod 7 = 5.

Vị trí 5 đã chứa giá trị 14 khác 42 → băm lại

Băm lại lần 1:

prob(42, 1) = (5 + 1.1 + 1) mod 7 = 0.

Vị trí 0 trống

→không tìm thấy khóa 42.

| **Chỉ số** | **Khóa** |
|---|---|
| **0** | |
| **1** | 26 |
| **2** | 31 |
| **3** | 12 |
| **4** | 17 |
| **5** | 14 |
| **6** | 39 |

# Câu 5:

Trong chương trình học của một trường đại học, một số môn học sẽ có các môn học tiên quyết, tức là môn học đó chỉ được học sau khi đã hoàn thành các môn học tiên quyết.

Hãy xác định người học có cần phải hoàn thành môn học A trước khi học môn học B hay không.

# Câu 5:

Ví dụ: bảng môn học và các môn học tiên quyết:

| STT | Môn học | Các môn học tiên quyết |
|---|---|---|
| **1** | Cấu trúc dữ liệu và giải thuật | Lập trình nâng cao |
| **2** | Lý thuyết tính toán | Cấu trúc dữ liệu và giải thuật |
| **3** | Trí tuệ nhân tạo | Cấu trúc dữ liệu và giải thuật, Nhập môn tin học |
| **4** | Nhập môn tin học | |
| **5** | Mật mã học và an toàn dữ liệu | Lý thuyết tính toán, Nhập môn tin học, Lập trình nâng cao |
| **6** | Lập trình nâng cao | Nhập môn tin học |

Với bảng điều kiện ở trên, người học cần hoàn thành môn Lập trình nâng cao trước khi học môn Trí tuệ nhân tạo.

# Câu 5:

a, Hãy mô hình hóa bài toán trên thành bài toán trên đồ thị.

# Câu 5:

b, Giả sử thông tin đầu vào của bài toán được nhập vào chương trình như sau:

| **Ví dụ input** | **Giải thích** |
|---|---|
| **8**<br><br>**Lap_trinh_nang_cao**<br>**Cau_truc_du_lieu_va_giai_thuat**<br>**Cau_truc_du_lieu_va_giai_thuat**<br>**Ly_thuyet_tinh_toan**<br>**…**<br>**Nhap_mon_tin_hoc Lap_trinh_nang_cao** | - Dòng đầu tiên là một số e là số cặp môn học mà môn này là môn học tiên quyết của môn còn lại<br>- e dòng tiếp theo, mỗi dòng chứa 2 chuỗi i và j, chuỗi i là môn học tiên quyết của chuỗi j, nếu một môn không có môn tiên quyết thì nhập chuỗi i là #<br>- dòng tiếp theo nhập vào hai chuỗi là x và y để kiểm tra môn x có phải là môn học trước môn y không |

# Câu 5:

| Ví dụ input | Giải thích |
|---|---|
| **8**<br><br>**Lap_trinh_nang_cao Cau_truc_du_lieu_va_giai_thuat**<br>**Cau_truc_du_lieu_va_giai_thuat Ly_thuyet_tinh_toan**<br>**...**<br>**Nhap_mon_tin_hoc Lap_trinh_nang_cao** | - Dòng đầu tiên là một số e là số cặp môn học mà môn này là môn học tiên quyết của môn còn lại<br>- e dòng tiếp theo, mỗi dòng chứa 2 chuỗi i và j, chuỗi i là môn học tiên quyết của chuỗi j,<br>- dòng tiếp theo nhập vào hai chuỗi là x và y để kiểm tra môn x có phải là môn học trước môn y không |

Hãy xây dựng cấu trúc dữ liệu thích hợp để biểu diễn đồ thị trên máy tính theo input đã cho. Viết hàm nhập theo ví dụ input ở đầu bài và lưu trữ thông tin của đồ thị vào cấu trúc dữ liệu đã đề xây dựng.

# Câu 5:

c, Viết chương trình thực hiện yêu cầu bài toán (có thể làm chung câu c với câu b).

# Câu 5a:

Bài toán có thể được mô hình hóa về bài toán trên đồ thị như sau: xem mỗi môn học là một đỉnh của đồ thị, nếu môn học ứng với đỉnh u là môn học tiên quyết của môn học ứng với đỉnh v thì có một cạnh đi từ u đến v, như vậy đây là một đồ thị có hướng.

Yêu cầu của bài toán trở thành kiểm tra có đường đi đi từ đỉnh một đỉnh i đến một đỉnh j nào đó hay không.

# Câu 5: b&c

```
#include<iostream>
#include<string>
#include <vector>
#include<map>
#include<algorithm>
using namespace std;
// bien toan cuc
map<int, vector<int>> adj_list;// map de luu danh sach cac dinh ke cua moi dinh
map<string, int> subject_to_index;// map de anh xa ten mon hoc thanh mot so
map<int, string> index_to_subject;// map de anh xa so thanh ten mon hoc
vector<bool> visited; // vector luu tru trang thai cua dinh
```

# Câu 5: b&c

```
void input()
{
    int e;
    cin >> e;
    int count = 0;
    for (int i = 0; i < e; i++)
    {
        string x, y;
        cin >> x >> y;
```

```
if (subject_to_index.find(x) ==
                subject_to_index.end())
    {
        subject_to_index[x] = count;
        index_to_subject[count] = x;
        count++;
    }
```

# Câu 5: b&c

```
if (subject_to_index.find(y) ==
        subject_to_index.end())
    {
        subject_to_index[y] = count;
        index_to_subject[count] = y;
        count++;
    }
```

```
int x_idx = subject_to_index[x];
        int y_idx = subject_to_index[y];
        adj_list[x_idx].push_back(y_idx);
    }
    visited.resize(count);
    visited = vector<bool>(count, false);
}
```

# Câu 5: b&c

```
void input()
{
    int e;
    cin >> e;
    int count = 0;
    for (int i = 0; i < e; i++)
    {
        string x, y; cin >> x >> y;
        if (subject_to_index.find(x) ==
                                    subject_to_index.end())
        {
            subject_to_index[x] = count;
            index_to_subject[count] = x;
            count++;
        }
```

```
        if (subject_to_index.find(y) == subject_to_index.end())
            {
                subject_to_index[y] = count;
                index_to_subject[count] = y;
                count++;
            }
            int x_idx = subject_to_index[x];
            int y_idx = subject_to_index[y];
            adj_list[x_idx].push_back(y_idx);
        }
        visited.resize(count);
        visited = vector<bool>(count, false);
    }
```

# Câu 5: b&c

```
bool dfs(int start, int end)
{
    if (start == end) return true; // truong hop dinh ket thuc duoc tim thay
    visited[start] = true; // danh dau la dinh da duoc duyet
    for (int i = 0; i < adj_list[start].size(); i++)
    {
        int next = adj_list[start][i];
        if (!visited[next])
        { // neu dinh tiep theo chua duoc duyet
            if (dfs(next, end)) return true; // neu tim thay tra ve true
        }
    }
    return false; // không tìm thấy đường đi
}
```

# Câu 5: b&c

```
int main()
{
    input();
    string start, end;
    cin >> start >> end;
    int start_index = subject_to_index[start];
    int end_index = subject_to_index[end];
    if (dfs(start_index, end_index))
        cout << "Mon hoc " << start << " can hoan thanh truoc khi hoc mon " << end;
    else
        cout << "Mon hoc " << start << " khong can hoan thanh truoc khi hoc mon " << end;

}
```

# LINK ĐIỂM DANH

# BAN HỌC TẬP CÔNG NGHỆ PHẦN MỀM

## TRAINING GIỮA KỲ HỌC KỲ I NĂM HỌC 2022 – 2023


Sharing is learning

# HẾT

## CẢM ƠN CÁC BẠN ĐÃ THEO DÕI
## CHÚC CÁC BẠN CÓ KẾT QUẢ THI THẬT TỐT!


### BAN HỌC TẬP

*Khoa Công nghệ Phần mềm*

*Trường Đại học Công nghệ Thông tin*

*Đại học Quốc gia thành phố Hồ Chí Minh*

### CONTACT

*bht.cnpm.uit@gmail.com*

*fb.com/bhtcnpm*

*fb.com/groups/bht.cnpm.uit*