



BÀI 10: THỰC NGHIỆM BÀI TOÁN TÌM KIẾM





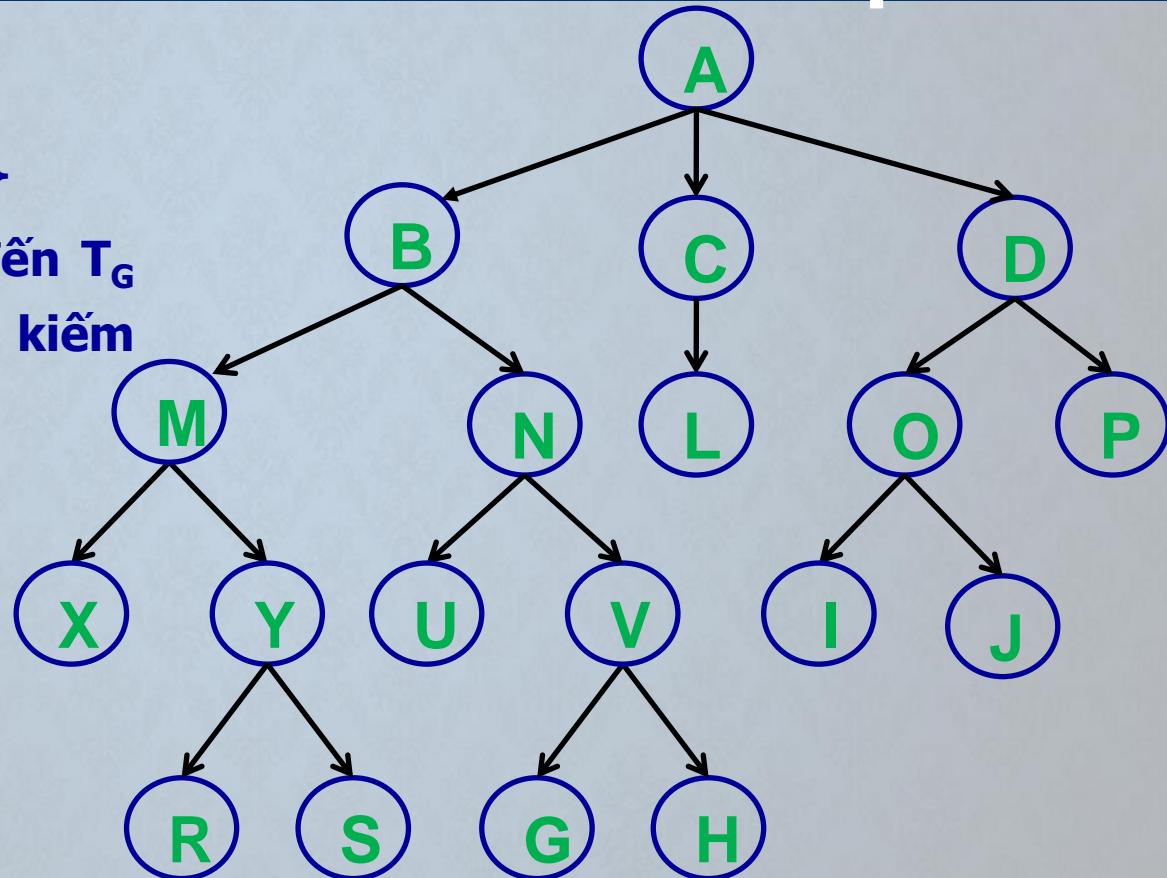
TÌM KIẾM THEO CHIỀU SÂU (DFS)

TÌM KIẾM THEO CHIỀU SÂU – Ví dụ

Cho đồ thị như sau:

Đỉnh đầu $T_0 = A$, $T_G = \{R\}$

Tìm đường đi p từ T_0 đến T_G
bằng phương pháp tìm kiếm
theo chiều sâu?



-
- **Input: Đồ thị, đỉnh đầu $T_0 = A$, $T_G = \{R\}$**
 - **Output: Có đường đi từ T_0 đến T_G không?**

Nếu có thì in ra đường đi, nếu không thì đưa ra thông báo

Tạo lớp Node có tên và cha của node



```
class Node:  
    def __init__(self, ten, Cha = None):  
        self.ten = ten  
        self.Cha = Cha  
    def display(self):  
        print(self.ten)
```

Tạo đồ thị dạng cây theo đề bài

```
[ ] from collections import defaultdict
```

```
[ ] data = defaultdict(list)
data['A'] = ['B', 'C', 'D']
data['B'] = ['M', 'N']
data['C'] = ['L']
data['D'] = ['O', 'P']
data['M'] = ['X', 'Y']
data['N'] = ['U', 'V']
data['O'] = ['I', 'J']
data['Y'] = ['R', 'S']
data['V'] = ['G', 'H']
```

Tạo hàm kiểm tra và hàm lưu vết đường đi

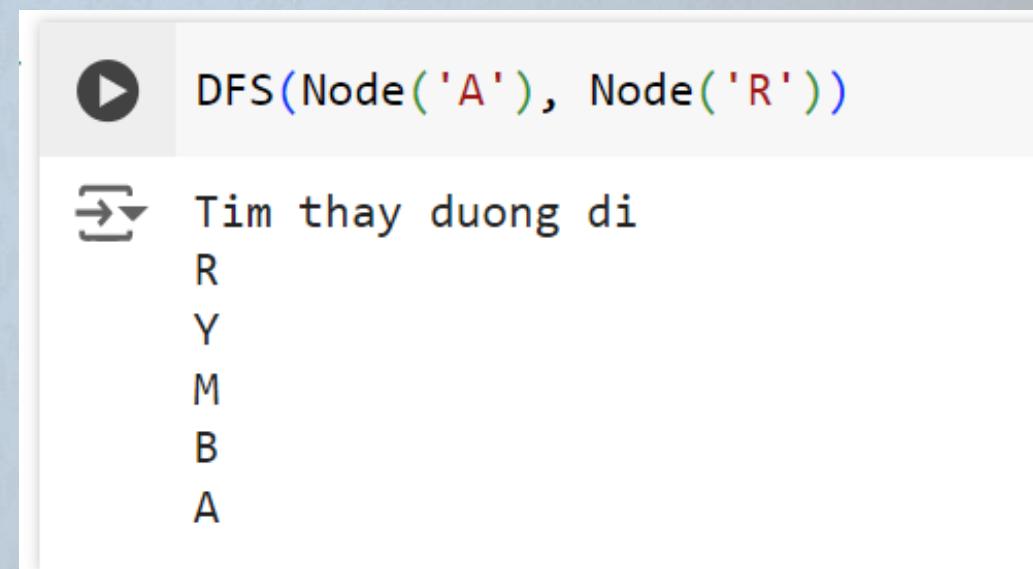
```
[ ] def kiemTra(tam, M0):
    for v in M0:
        if v.ten == tam.ten:
            return True
    return False

def DuongDi(n):
    print(n.ten)
    if n.Cha !=None:
        DuongDi(n.Cha)
    else:
        return
```

Tạo hàm tìm kiếm theo chiều sâu

▶ def DFS (To, Tg):
 MO = []
 DONG = []
 MO.append(To)
 while True:
 if len(MO) == 0:
 print ('Tim kiem khong thanh cong')
 return
 n = MO.pop(0)
 if n.ten == Tg.ten:
 print('Tim thay duong di')
 DuongDi(n)
 return
 DONG.append(n)
 pos = 0
 for v in data[n.ten]:
 tam = Node(v)
 ok1 = kiemTra(tam, MO)
 ok2 = kiemTra(tam, DONG)
 if not ok1 and not ok2:
 MO.insert(pos, tam)
 pos += 1
 tam.Cha = n

Kết quả sau
khi thực
hiện gọi
hàm tìm
kiếm theo
chiều sâu
với đỉnh đầu
là A và đích
là R.





TÌM KIẾM THEO CHIỀU RỘNG (BFS)

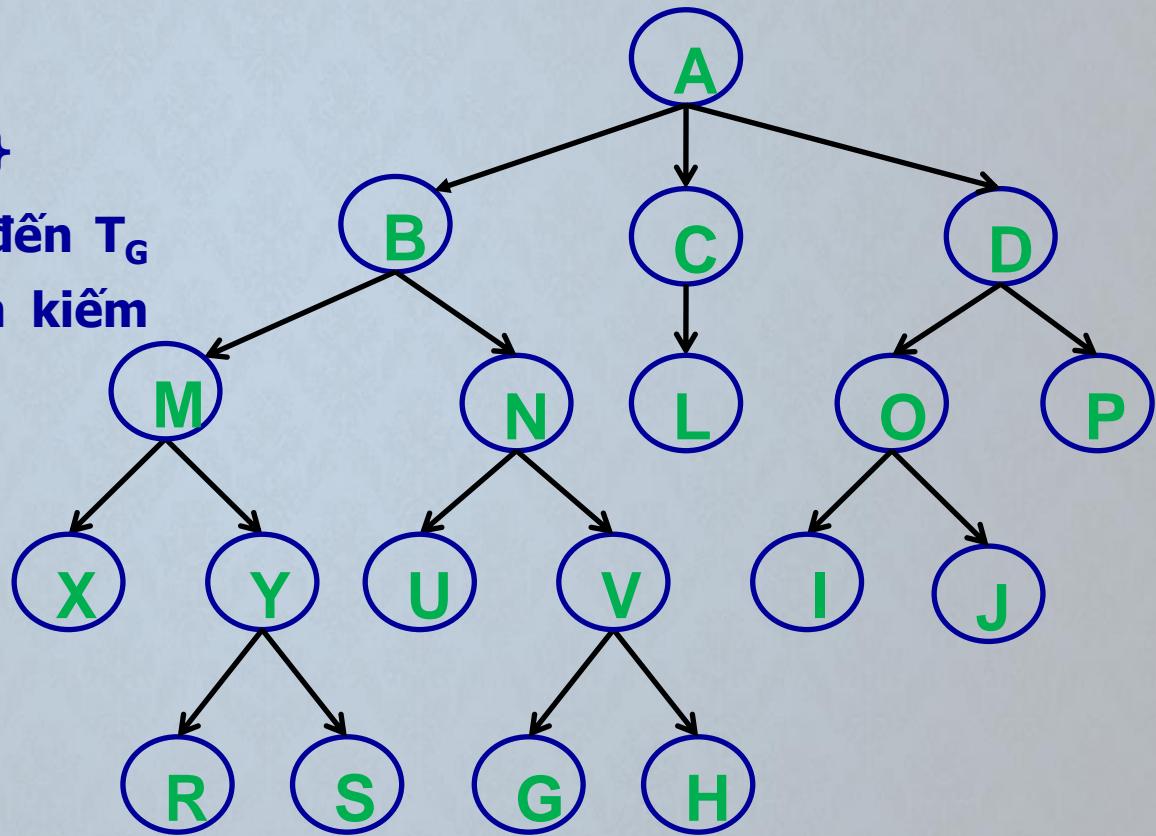


TÌM KIẾM THEO CHIỀU RỘNG – Ví dụ

Cho đồ thị như sau:

Đỉnh đầu $T_0 = A$, $T_G = \{N\}$

Tìm đường đi p từ T_0 đến T_G
bằng phương pháp tìm kiếm
theo chiều rộng?



-
- **Input: Đồ thị, đỉnh đầu $T_0 = A$, $T_G = \{N\}$**
 - **Output: Có đường đi từ T_0 đến T_G không?**

Nếu có thì in ra đường đi, nếu không thì đưa ra thông báo

Tạo lớp Node có tên và cha của node



```
class Node:  
    def __init__(self, ten, Cha = None):  
        self.ten = ten  
        self.Cha = Cha  
    def display(self):  
        print(self.ten)
```

Tạo đồ thị dạng cây theo đề bài

```
[ ] from collections import defaultdict
```

```
[ ] data = defaultdict(list)
data['A'] = ['B', 'C', 'D']
data['B'] = ['M', 'N']
data['C'] = ['L']
data['D'] = ['O', 'P']
data['M'] = ['X', 'Y']
data['N'] = ['U', 'V']
data['O'] = ['I', 'J']
data['Y'] = ['R', 'S']
data['V'] = ['G', 'H']
```

Tạo hàm kiểm tra và hàm lưu vết đường đi

```
[ ] def kiemTra(tam, M0):
    for v in M0:
        if v.ten == tam.ten:
            return True
    return False

def DuongDi(n):
    print(n.ten)
    if n.Cha !=None:
        DuongDi(n.Cha)
    else:
        return
```

Tạo hàm tìm kiếm theo chiều rộng



```
def BFS(To, Tg):
    MO = []
    DONG = []
    MO.append(To)
    while True:
        if len(MO) == 0:
            print('tim kiem khong thanh cong')
            return
        n = MO.pop(0)
        if n.ten == Tg.ten:
            print('Tim kiem thanh cong')
            DuongDi(n)
            return
        DONG.append(n)
        for v in data[n.ten]:
            tam = Node(v)
            ok1 = kiemTra(tam, MO)
            ok2 = kiemTra(tam, DONG)
            if not ok1 and not ok2:
                MO.append(tam)
                tam.Cha = n
```

Kết quả sau
khi thực
hiện gọi
hàm tìm
kiếm theo
chiều rộng
với đỉnh đầu
là A và đích
là N.

```
[ ] BFS(Node('A'), Node('N'))  
→ Tim kiem thanh cong  
N  
B  
A
```