# Bài 17: Thực hành duyệt đồ thị tổng hợp

**Giải Chuyên đề Tin học 12 Bài 17: Thực hành duyệt đồ thị tổng hợp**  
**Khởi động trang 80 Chuyên đề Tin học 12**: Trong bài thực hành trước chúng ta đã được ôn tập và giải một số bài toán có áp dụng thuật toán duyệt đồ thị theo chiều sâu. Còn về thuật toán duyệt theo chiều rộng em có biết gì về các ứng dụng thực tế của bài toán này không?  
**Lời giải:**  
Thuật toán duyệt đồ thị theo chiều rộng (BFS - Breadth-First Search) cũng rất hữu ích và được sử dụng trong nhiều ứng dụng thực tế. Dưới đây là một số ứng dụng phổ biến của BFS:  
- Tìm kiếm ngắn nhất trong đồ thị không có trọng số  
- Tìm kiếm ngắn nhất trong mạng lưới (grid)  
- Tìm kiếm trạng thái  
- Tính toán khoảng cách  
- Kiểm tra tính liên thông của đồ thị  
- Tìm kiếm trong cấu trúc dữ liệu cây  
**Luyện tập 1 trang 82 Chuyên đề Tin học 12**: Sửa lại phần nhập dữ liệu hai học sinh: sẽ nhập trực tiếp tên hai học sinh, kiểm tra các tên này có nhập đúng không và thực hiện yêu cầu như trong chương trình trên.  
**Lời giải:**  
Để sửa lại phần nhập dữ liệu để nhập trực tiếp tên hai học sinh và kiểm tra tính hợp lệ của chúng, chúng ta có thể thực hiện như sau:  
Bài mẫu tổng quát gợi ý:  
def input\_student\_names():  
 while True:  
 name1 = input("Nhập tên học sinh thứ nhất: ")  
 name2 = input("Nhập tên học sinh thứ hai: ")  
 if name1.strip() == "" or name2.strip() == "":  
 print("Tên học sinh không được để trống. Vui lòng nhập lại.")  
 elif name1 == name2:  
 print("Hai học sinh không thể có cùng tên. Vui lòng nhập lại.")  
 else:  
 return name1, name2  
# Hàm tính tổng điểm trung bình  
def calculate\_average\_score(score\_list):  
 return sum(score\_list) / len(score\_list)  
# Hàm in thông tin học sinh  
def print\_student\_info(name, scores):  
 avg\_score = calculate\_average\_score(scores)  
 print(f"Học sinh {name}:")  
 print(f" - Điểm Toán: {scores[0]}")  
 print(f" - Điểm Văn: {scores[1]}")  
 print(f" - Điểm Anh: {scores[2]}")  
 print(f"Điểm trung bình: {avg\_score:.2f}")  
# Hàm nhập điểm cho học sinh  
def input\_student\_scores(name):  
 scores = []  
 for subject in ["Toán", "Văn", "Anh"]:  
 while True:  
 try:  
 score = float(input(f"Nhập điểm {subject} của học sinh {name}: "))  
 if score < 0 or score > 10:  
 print("Điểm phải nằm trong khoảng từ 0 đến 10. Vui lòng nhập lại.")  
 else:  
 scores.append(score)  
 break  
 except ValueError:  
 print("Điểm phải là một số thực. Vui lòng nhập lại.")  
 return scores  
# Hàm chính  
def main():  
 print("Nhập thông tin của hai học sinh:")  
 name1, name2 = input\_student\_names()  
 scores1 = input\_student\_scores(name1)  
 scores2 = input\_student\_scores(name2)  
 print("\nThông tin hai học sinh sau khi nhập:")  
 print\_student\_info(name1, scores1)  
 print()  
 print\_student\_info(name2, scores2)  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
- Trong phiên bản này, chúng ta sử dụng một vòng lặp **while** để yêu cầu người dùng nhập tên hai học sinh. Sau đó, kiểm tra tính hợp lệ của tên đó (không được để trống và hai học sinh không thể có cùng tên). Nếu các tên được nhập đúng cách, chúng ta trả về tên của hai học sinh và tiếp tục với việc nhập điểm và in thông tin học sinh như bình thường.  
**Luyện tập 2 trang 82 Chuyên đề Tin học 12**: Viết lại hàm BFS() trong chương trình trên nhưng sử dụng ma trận kề A thay thế cho danh sách kề Adj.  
**Lời giải:**  
Viết lại hàm BFS sử dụng ma trận kề A thay vì danh sách kề Adj. Dưới đây là cách triển khai:  
from collections import deque  
def BFS(graph, start):  
 n = len(graph)  
 visited = [False] \* n  
 visited[start] = True  
 queue = deque([start])  
 while queue:  
 vertex = queue.popleft()  
 print("Visit:", vertex)  
 for neighbor in range(n):  
 if graph[vertex][neighbor] == 1 and not visited[neighbor]:  
 visited[neighbor] = True  
 queue.append(neighbor)  
# Ma trận kề mẫu  
graph\_matrix = [  
 [0, 1, 1, 0, 0, 0],  
 [1, 0, 0, 1, 1, 0],  
 [1, 0, 0, 0, 0, 1],  
 [0, 1, 0, 0, 0, 0],  
 [0, 1, 0, 0, 0, 1],  
 [0, 0, 1, 0, 1, 0]  
]  
# Thực hiện BFS từ đỉnh 0  
BFS(graph\_matrix, 0)  
- Trong hàm BFS này, chúng ta duyệt qua các hàng của ma trận kề để xác định các đỉnh kề của mỗi đỉnh. Nếu có một đỉnh kề chưa được duyệt, chúng ta đánh dấu đỉnh đó đã được thăm và thêm nó vào hàng đợi để duyệt tiếp.  
**Vận dụng 1 trang 82 Chuyên đề Tin học 12**: Bổ sung thêm yêu cầu của nhiệm vụ trên như sau: Có hay không hai bạn học sinh trong lớp mà không thể đi xe đạp từ nhà bạn này đến nhà bạn kia. Nếu có thì thông báo tên hai bạn học sinh đó.  
**Lời giải:**  
Để giải quyết vấn đề này, trước hết chúng ta cần xác định liệu có đường đi từ một nhà đến nhà khác không, bằng cách sử dụng hàm find\_path mà chúng ta đã viết trước đó. Sau đó, chúng ta sẽ kiểm tra xem có đường đi giữa hai người bạn không.  
Dưới đây là cách triển khai trong Python:  
def find\_unreachable\_students(graph):  
 unreachable\_students = [] # Danh sách các bạn học sinh không thể đi xe đạp đến nhau  
 # Duyệt qua tất cả các cặp học sinh trong lớp  
 for student1 in graph:  
 for student2 in graph:  
 if student1 != student2:  
 # Kiểm tra xem có đường đi từ nhà của student1 đến nhà của student2 không  
 path = find\_path(graph, student1, student2)  
 if not path:  
 unreachable\_students.append((student1, student2))  
 return unreachable\_students  
# Ví dụ về đồ thị được biểu diễn bằng danh sách kề  
graph = {  
 'Alice': ['Bob', 'Charlie'],  
 'Bob': ['Alice', 'David', 'Eva'],  
 'Charlie': ['Alice', 'Frank'],  
 'David': ['Bob'],  
 'Eva': ['Bob', 'Frank'],  
 'Frank': ['Charlie', 'Eva']  
}  
# Tìm các bạn học sinh không thể đi xe đạp đến nhau  
unreachable\_students = find\_unreachable\_students(graph)  
if unreachable\_students:  
 print("Có những bạn học sinh không thể đi xe đạp đến nhau:")  
 for student1, student2 in unreachable\_students:  
 print(student1, "và", student2)  
else:  
 print("Không có bạn học sinh nào không thể đi xe đạp đến nhau.")  
- Lưu ý:Trong đoạn mã trên, hàmfind\_unreachable\_students sẽ duyệt qua tất cả các cặp học sinh trong danh sách và kiểm tra xem có đường đi nào giữa họ không. Nếu không có đường đi, họ sẽ được thêm vào danh sách unreachable\_students. Sau đó, danh sách này sẽ được in ra để thông báo cho chúng ta biết những học sinh nào không thể đi xe đạp đến nhau.  
**Vận dụng 2 trang 82 Chuyên đề Tin học 12**: Thiết lập hàm printpath(s,t) không đệ quy có tính năng tương tự hàm cùng tên trong chương trình trên.  
**Lời giải:**  
Phiên bản không đệ quy của hàmprintpath(s, t) bằng cách sử dụng một vòng lặp thay vì đệ quy:  
def printpath(s, t):  
 if t == s:  
 print(names[s], end="")  
 return  
 path = []  
 current = t  
 while current != s:  
 if prev[current] == None:  
 print("Không tồn tại đường đi")  
 return  
 path.append(names[current])  
 current = prev[current]  
 path.append(names[s])  
 path.reverse()  
 print(" -> ".join(path), end=" ")  
# Đoạn mã dưới đây mô phỏng dữ liệu của biến names và prev  
names = ["A", "B", "C", "D", "E"]  
prev = [None, 0, 1, 0, 2]  
# Sử dụng hàm printpath  
printpath(0, 4) # Mô phỏng đường đi từ đỉnh 0 đến 4  
**-** Chú ý: Trong hàm này, chúng ta sử dụng một vòng lặp để lặp lại từ đỉnh đích **t** đến đỉnh nguồn **s**, và trong quá trình này, chúng ta xây dựng một danh sách đường đi. Sau đó, chúng ta đảo ngược danh sách này và in ra. Nếu không có đường đi từ **s** đến **t**, chúng ta in ra thông báo "Không tồn tại đường đi".