

# ĐỀ THI THỬ THỰC HÀNH TOÁN TỐ HỢP - ĐỀ SỐ 2 (BỔ SUNG)

*Thời gian: 60 phút / Mục tiêu: Lắp đầy kiến thức nâng cao & Kỹ năng Python \_\_\_\_\_*

## I. LÝ THUYẾT & KHÁI NIỆM MỞ RỘNG (20%)

**Câu 1.** Một đồ thị được gọi là **Đồ thị hai phía** (Bipartite Graph) khi và chỉ khi:

- A. Nó không chứa chu trình nào.
- B. Nó không chứa chu trình lẻ.
- C. Nó là một cây khung.
- D. Độ bắc của mọi đỉnh đều là số chẵn.

**Câu 2.** Trong bài toán **Tô màu đồ thị** (Graph Coloring), "Sắc số" (Chromatic Number)  $\chi(G)$  là gì?

- A. Số lượng màu ít nhất để tô các đỉnh sao cho 2 đỉnh kề nhau khác màu.
- B. Số lượng màu ít nhất để tô các cạnh sao cho 2 cạnh kề nhau khác màu.
- C. Số lượng đỉnh tối đa có thể tô cùng một màu.
- D. Tổng số màu cần thiết để tô hết tất cả các đỉnh (mỗi đỉnh 1 màu).

**Câu 3.** Thuật toán **Prim** phát triển cây khung nhỏ nhất (MST) dựa trên nguyên lý nào khác với **Kruskal**?

- A. Prim sắp xếp tất cả các cạnh ngay từ đầu.
- B. Prim phát triển cây từ một đỉnh xuất phát, luôn chọn cạnh nhẹ nhất nối từ cây ra ngoài.
- C. Prim chỉ hoạt động trên đồ thị có hướng.
- D. Prim sử dụng cấu trúc Union-Find (Disjoint Set).

**Câu 4.** Điều kiện cần và đủ để một đồ thị có hướng có **chu trình Euler** là:

- A. Đồ thị liên thông yếu và mọi đỉnh có bậc vào bằng bậc ra.
- B. Đồ thị liên thông mạnh và mọi đỉnh có bậc vào bằng bậc ra.
- C. Đồ thị liên thông và tất cả các đỉnh có bậc chẵn.
- D. Có đúng 2 đỉnh có bậc lẻ.

**Câu 5.** Trong Python, độ phức tạp của thao tác kiểm tra `x in s` với `s` là một **Set** và một **List** có  $n$  phần tử lần lượt là:

- A.  $O(1)$  và  $O(n)$
- B.  $O(n)$  và  $O(1)$

- C.  $O(\log n)$  và  $O(n)$   
D.  $O(1)$  và  $O(1)$

**Câu 6. Đường đi Euler khác Chu trình Euler ở điểm nào?**

- A. Đường đi Euler đi qua mỗi cạnh đúng 1 lần nhưng không nhất thiết quay về điểm đầu.  
B. Đường đi Euler đi qua mỗi đỉnh đúng 1 lần.  
C. Đường đi Euler không được đi qua cạnh trọng số âm.  
D. Không có sự khác biệt.

**Câu 7.** Ma trận  $A$  biểu diễn đồ thị có hướng, nếu  $A^k$  (lũy thừa bậc  $k$  của ma trận) có phần tử  $a_{ij} > 0$ , điều này có ý nghĩa gì?

- A. Khoảng cách ngắn nhất từ  $i$  đến  $j$  là  $k$ .  
B. Có ít nhất một đường đi độ dài đúng bằng  $k$  từ  $i$  đến  $j$ .  
C. Có  $k$  đường đi từ  $i$  đến  $j$ .  
D. Điểm  $i$  và  $j$  thuộc thành phần liên thông thứ  $k$ .

**Câu 8.** Thuật toán nào sau đây giải quyết bài toán **Luồng cực đại** (Max Flow)?

- A. Floyd-Warshall  
B. Ford-Fulkerson  
C. Kruskal  
D. Bellman-Ford

## II. KỸ NĂNG PYTHON: FILE, CLASS & DATA (30%)

**Câu 9.** Cho file `data.txt` có nội dung:

```
3
0 1 5
1 2 3
```

Đoạn code sau thực hiện việc gì?

```
1 with open('data.txt', 'r') as f:
2     n = int(f.readline())
3     edges = []
4     for line in f:
5         edges.append(list(map(int, line.split())))
```

- A. Đọc ma trận kè kề kích thước  $3 \times 3$ .  
B. Đọc số lượng đỉnh và danh sách cạnh  $(u, v, w)$  vào list `edges`.  
C. Báo lỗi vì dòng đầu tiên chỉ có 1 số.  
D. Tạo danh sách kè trực tiếp.

**Câu 10.** Để tạo một bản sao sâu (Deep Copy) của một ma trận `matrix` trong Python để khi sửa bản sao không ảnh hưởng bản gốc, ta dùng:

- A. `new_mat = matrix.copy()`
- B. `new_mat = list(matrix)`
- C. `new_mat = matrix[:]`
- D. `import copy; new_mat = copy.deepcopy(matrix)`

**Câu 11.** Cho class `Graph` như sau. Phương thức `add_edge` đang cài đặt đồ thị loại nào?

```

1 class Graph:
2     def __init__(self):
3         self.adj = {}
4     def add_edge(self, u, v):
5         if u not in self.adj: self.adj[u] = []
6         self.adj[u].append(v)

```

- A. Đồ thị vô hướng.
- B. Đồ thị có hướng.
- C. Đồ thị có trọng số.
- D. Cây nhị phân.

**Câu 12.** Kết quả của `print(set([1, 2, 2, 3]) & set([2, 3, 4]))` là:

- A. {1, 2, 3, 4}
- B. {2, 3}
- C. [2, 3]
- D. {1, 4}

**Câu 13.** Đoạn code sau dùng để làm gì trong thuật toán Dijkstra?

```

1 import heapq
2 # ... pq là priority queue
3 d, u = heapq.heappop(pq)
4 if d > dist[u]: continue

```

- A. Cập nhật khoảng cách.
- B. Lazy deletion (Bỏ qua các cặp đỉnh-khoảng cách cũ đã lỗi thời trong Heap).
- C. Dừng thuật toán.
- D. Kiểm tra đồ thị có chu trình âm hay không.

**Câu 14.** Hàm `sys.stdin.read().split()` thường được dùng trong các bài thi lập trình để:

- A. Đọc từng dòng một.
- B. Đọc toàn bộ dữ liệu đầu vào và tách thành list các chuỗi (tokens) nhanh chóng.

- C. Chỉ đọc các số nguyên.  
D. Ghi dữ liệu ra file.

**Câu 15.** Lỗi thường gặp nhất khi cài đặt DFS bằng đệ quy trên đồ thị lớn ( $> 10^5$  đỉnh) trong Python là gì?

- A. `TimeLimitExceeded`  
B. `MemoryError`  
C. `RecursionError` (Maximum recursion depth exceeded)  
D. `KeyError`

**Câu 16.** Biểu thức `[x**2 for x in range(5) if x % 2 == 0]` trả về:

- A. [0, 1, 4, 9, 16]  
B. [0, 2, 4]  
C. [0, 4, 16]  
D. [1, 9]

### III. TƯ DUY & TÍNH TOÁN BẰNG TAY (30%)

**Câu 17.** Cho đồ thị có các cạnh và trọng số:  $(A, B) : 2, (B, C) : -5, (C, A) : 1$ . Áp dụng **Relaxation** (làm mềm) của Bellman-Ford. Giả sử  $d[A] = 0, d[B] = \infty, d[C] = \infty$ . Sau vòng lặp đầu tiên duyệt qua các cạnh theo thứ tự  $(A, B), (B, C), (C, A)$ , giá trị  $d[A]$  sẽ là:

- A. 0  
B. -2  
C. 1  
D.  $\infty$

**Câu 18.** Sắc số  $\chi(G)$  của đồ thị đầy đủ  $K_6$  là:

- A. 2  
B. 4  
C. 5  
D. 6

**Câu 19.** Đồ thị lưỡng phân (Bipartite)  $K_{3,4}$  có tổng cộng bao nhiêu cạnh?

- A. 7  
B. 12  
C. 6  
D. 34

**Câu 20.** Chạy thuật toán Prim bắt đầu từ đỉnh A cho đồ thị:  $A - B(1), A - C(4), B - C(2), B - D(6)$ . Thứ tự các cạnh được thêm vào MST là:

- A.  $(A, B), (B, C), (B, D)$
- B.  $(A, B), (A, C), (B, D)$
- C.  $(A, B), (B, C), (C, D)$  (Nếu có cạnh C-D)
- D.  $(A, C), (B, C), (B, D)$

**Câu 21.** Ma trận kề của đồ thị  $G$  là  $M$ . Phần tử  $M[i][j] = 1$  nghĩa là có cạnh nối  $i \rightarrow j$ . Nếu  $M[i][i] = 1$  với mọi  $i$ , đồ thị này có đặc điểm gì?

- A. Đồ thị vô hướng.
- B. Mọi đỉnh đều có một khuyên (loop) nối với chính nó.
- C. Đồ thị liên thông mạnh.
- D. Ma trận đơn vị.

**Câu 22.** Cho đồ thị 5 đỉnh  $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ . Ma trận khoảng cách cuối cùng của thuật toán Floyd-Warshall tại phần tử  $\text{dist}[0][4]$  sẽ có giá trị bao nhiêu? (Giả sử trọng số mỗi cạnh là 1).

- A. 1
- B. 3
- C. 4
- D.  $\infty$

**Câu 23.** Một đồ thị vô hướng có  $V = 10$  đỉnh và  $E = 15$  cạnh. Số thành phần liên thông tối thiểu của đồ thị này là:

- A. 1
- B. 10
- C. 5
- D. 2

**Câu 24.** Trong đồ thị luồng (Flow Network), nếu đường tăng luồng (augmenting path) tìm được là  $S \rightarrow A \rightarrow B \rightarrow T$  với khả năng thông qua (capacity) còn dư lần lượt là  $(3, 5, 2)$ . Luồng sẽ được tăng bao nhiêu?

- A. 2 (Giá trị nhỏ nhất - Bottleneck)
- B. 3
- C. 5
- D. 10 (Tổng)

**Câu 25.** Phát hiện chu trình âm bằng Bellman-Ford được thực hiện ở vòng lặp thứ mấy (với  $V$  đỉnh)?

- A.  $V - 1$
- B.  $V$
- C.  $V + 1$
- D.  $V/2$

**Câu 26.** Đồ thị chu trình  $C_5$  (ngũ giác) cần tối thiểu bao nhiêu màu để tô đỉnh?

- A. 2
- B. 3
- C. 4
- D. 5

#### IV. ỨNG DỤNG THỰC TẾ & MÔ HÌNH HÓA (20%)

**Câu 27.** Bài toán "Xếp lịch thi sao cho số ca thi là ít nhất mà không sinh viên nào bị trùng lịch" được mô hình hóa bằng bài toán:

- A. Tìm luồng cực đại.
- B. Tìm chu trình Euler.
- C. Tô màu đồ thị (Graph Coloring).
- D. Tìm cây khung nhỏ nhất.

**Câu 28.** Để phân công  $N$  nhân viên vào  $N$  công việc sao cho tổng chi phí thấp nhất (hoặc năng suất cao nhất), ta dùng thuật toán trên mô hình nào?

- A. Shortest Path trên DAG.
- B. Minimum Spanning Tree.
- C. Matching trên đồ thị hai phía (Bipartite Matching) có trọng số.
- D. Topological Sort.

**Câu 29.** Trong quản lý dự án, phương pháp **PERT/CPM** tìm "Đường găng" (Critical Path) để xác định thời gian hoàn thành dự án tối thiểu tương đương với bài toán nào trên đồ thị?

- A. Đường đi ngắn nhất.
- B. Đường đi dài nhất trên đồ thị không chu trình (Longest path on DAG).
- C. Chu trình Hamiltonian.
- D. Luồng cực đại.

**Câu 30.** Hệ thống khuyến nghị (Recommender System) muốn gợi ý: "Người dùng A và B cùng thích phim X, Y; B thích Z nên gợi ý Z cho A". Đây là ứng dụng của:

- A. Lọc cộng tác dựa trên Đồ thị hai phía (User-Item Bipartite Graph).

- B. Tìm đường đi ngắn nhất Dijkstra.
- C. Tô màu đồ thị.
- D. Duyệt DFS.

**Câu 31.** Bài toán "Người đưa thư Trung Hoa" (Chinese Postman Problem) - đi qua tất cả các con đường trong khu phố rồi quay về điểm xuất phát với chi phí thấp nhất - liên quan mật thiết đến khái niệm:

- A. Chu trình Euler.
- B. Chu trình Hamiltonian.
- C. Cây khung nhỏ nhất.
- D. Luồng cực đại.

**Câu 32.** Để phát hiện tình trạng **Deadlock** (khóa chết) trong hệ điều hành (Resource Allocation Graph), ta tìm kiếm cái gì trên đồ thị?

- A. Cây khung.
- B. Chu trình trong đồ thị có hướng.
- C. Đường đi ngắn nhất.
- D. Dỉnh có bậc cao nhất.

**Câu 33.** Khi router trên Internet gửi gói tin, giao thức OSPF (Open Shortest Path First) thường sử dụng thuật toán nào để dựng bảng định tuyến?

- A. Dijkstra.
- B. Ford-Fulkerson.
- C. Prim.
- D. DFS.

**Câu 34.** Để xác định xem một phân tử hóa học có cấu trúc vòng hay không, ta dùng:

- A. BFS/DFS để phát hiện chu trình.
- B. Dijkstra.
- C. Kruskal.
- D. Tô màu đồ thị.

**Câu 35.** Bài toán xếp hậu (N-Queens) có thể giải quyết bằng kỹ thuật quay lui (Backtracking), thực chất là duyệt đồ thị trạng thái theo phương pháp nào?

- A. BFS.
- B. DFS.
- C. Prim.
- D. Floyd-Warshall.

**Câu 36.** Một mạng lưới điện muốn đảm bảo nếu một đường dây bất kỳ bị đứt thì hệ thống vẫn không bị chia cắt. Đây là bài toán kiểm tra tính chất gì?

- A. Cầu (Bridge) và Khớp (Articulation Point).
- B. Chu trình âm.
- C. Sắc số của đồ thị.
- D. Đường đi Euler.

**Câu 37.** Tính chất "6 bậc phân cách" (Six degrees of separation) trong mạng xã hội ám chỉ điều gì?

- A. Đường kính của đồ thị xã hội rất nhỏ (khoảng 6).
- B. Mọi đỉnh có bậc là 6.
- C. Đồ thị có 6 thành phần liên thông.
- D. Cần 6 màu để tô đồ thị.

**Câu 38.** Nếu muốn tìm đường đi ngắn nhất trong mê cung mà tại mỗi bước đi có thêm chi phí khác nhau (trọng số dương), thuật toán tốt nhất là:

- A. BFS.
- B. Dijkstra / A\*.
- C. DFS.
- D. Bellman-Ford.

**Câu 39.** Cấu trúc Disjoint Set (Union-Find) được sử dụng hiệu quả nhất trong thuật toán nào?

- A. Prim.
- B. Kruskal.
- C. Dijkstra.
- D. Floyd-Warshall.

**Câu 40.** Để kiểm tra xem một đồ thị có phải là cây hay không, ta cần kiểm tra 2 điều kiện nào?

- A. Có  $N$  đỉnh và  $N$  cạnh.
- B. Liên thông và không có chu trình (số cạnh =  $N - 1$ ).
- C. Có hướng và không có chu trình.
- D. Mọi đỉnh đều có bậc chẵn.

# ĐÁP ÁN THAM KHẢO

Câu	ĐA	Câu	ĐA	Câu	ĐA	Câu	ĐA
1	B	11	B	21	B	31	A
2	A	12	B	22	C	32	B
3	B	13	B	23	A	33	A
4	B	14	B	24	A	34	A
5	A	15	C	25	B	35	B
6	A	16	C	26	B	36	A
7	B	17	B	27	C	37	A
8	B	18	D	28	C	38	B
9	B	19	B	29	B	39	B
10	D	20	A	30	A	40	B

## GIẢI THÍCH MỘT SỐ CÂU KHÓ:

- **Câu 5:** Tìm kiếm trong Set là  $O(1)$  (trung bình), List là  $O(n)$ .
- **Câu 17:** Trace Bellman-Ford:  $d[A] = 0$ .  $(A, B) : d[B] = 2$ .  $(B, C) : d[C] = 2 - 5 = -3$ .  $(C, A) : d[A] = -3 + 1 = -2$ .
- **Câu 19:**  $K_{m,n}$  có  $m \times n$  cạnh.  $K_{3,4}$  có  $3 \times 4 = 12$  cạnh.
- **Câu 29:** Critical Path Method (CPM) tìm đường đi dài nhất trên đồ thị có hướng không chu trình (DAG).

# GIẢI THÍCH CHI TIẾT ĐỀ SỐ 2

## PHẦN I: LÝ THUYẾT & KHÁI NIỆM (Câu 1 - 8)

**Câu 1 (B):** **Đồ thị hai phía (Bipartite Graph)** có tính chất đặc trưng là không chứa chu trình lẻ. Nếu có chu trình lẻ, ta không thể chia các đỉnh thành 2 tập  $U, V$  rời nhau sao cho các cạnh chỉ nối giữa  $U$  và  $V$ .

**Câu 2 (A):** **Sắc số**  $\chi(G)$  là số màu tối thiểu cần dùng để tô các đỉnh sao cho không có 2 đỉnh kề nhau cùng màu.

**Câu 3 (B):** **Prim** phát triển cây khung (MST) bằng cách bắt đầu từ 1 đỉnh và luôn chọn cạnh có trọng số nhỏ nhất nối từ tập đỉnh đã chọn ( $V_{MST}$ ) ra tập đỉnh chưa chọn ( $V \setminus V_{MST}$ ). Ngược lại, **Kruskal** chọn cạnh nhỏ nhất trên toàn đồ thị (miễn không tạo chu trình).

**Câu 4 (B):** Với đồ thị có hướng, điều kiện để có **Chu trình Euler** là đồ thị phải liên thông mạnh (các cạnh thuộc một thành phần liên thông) và tại mọi đỉnh:  $\text{Bậc vào} = \text{Bậc ra}$  ( $\deg^-(v) = \deg^+(v)$ ).

**Câu 5 (A):** **Set** trong Python dùng Hash Table, nên tra cứu trung bình là  $O(1)$ . **List** phải duyệt tuần tự nên là  $O(n)$ .

**Câu 6 (A):** **Đường đi Euler** đi qua mỗi cạnh đúng 1 lần nhưng điểm đầu và điểm cuối có thể khác nhau. Chu trình Euler bắt buộc điểm đầu trùng điểm cuối.

**Câu 7 (B):** Trong lý thuyết đồ thị đại số, phần tử  $(i, j)$  của ma trận  $A^k$  cho biết số lượng đường đi (walks) có độ dài đúng bằng  $k$  từ  $i$  đến  $j$ . Nếu  $a_{ij} > 0$  nghĩa là tồn tại ít nhất một đường đi như vậy.

**Câu 8 (B):** **Ford-Fulkerson** (và biến thể Edmonds-Karp) là thuật toán kinh điển giải bài toán Luồng cực đại trên mạng.

## PHẦN II: KỸ NĂNG PYTHON (Câu 9 - 16)

**Câu 9 (B):** `f.readline()` đọc dòng đầu (số 3 - số đỉnh). Vòng lặp `for line in f` đọc các dòng còn lại (cạnh), `split()` tách chuỗi và `map(int, ...)` chuyển về số nguyên. Kết quả là list các cạnh.

**Câu 10 (D):** Ma trận là list chứa các list con. `copy()` chỉ sao chép lớp ngoài (Shallow copy). Để sao chép đệ quy cả các list con bên trong, bắt buộc dùng `copy.deepcopy()`.

**Câu 11 (B):** Code chỉ có `adj[u].append(v)` mà không có chiều ngược lại `adj[v].append(u)`, suy ra đây là đồ thị **có hướng** (Directed Graph).

**Câu 12 (B):** `set([1, 2, 2, 3]) → {1, 2, 3}`. `set([2, 3, 4]) → {2, 3, 4}`. Phép `&` là giao hai tập hợp  $\rightarrow \{2, 3\}$ .

**Câu 13 (B):** Trong Priority Queue, ta không xóa được các phần tử cũ khi cập nhật khoảng cách ngắn hơn. Do đó, khi pop ra một đỉnh  $u$  với khoảng cách  $d$ , nếu  $d > dist[u]$  (nghĩa là đã tìm được đường khác ngắn hơn trước đó), ta phải bỏ qua (`continue`). Đây là kỹ thuật **Lazy Deletion**.

**Câu 14 (B):** `sys.stdin.read()` đọc toàn bộ input vào 1 chuỗi lớn, `.split()` tự động tách theo khoảng trắng/xuống dòng. Cách này cực nhanh cho bài toán có input lớn ( $> 10^5$  dòng).

**Câu 15 (C):** Giới hạn đệ quy mặc định của Python là 1000. Đồ thị 1 đường thẳng  $10^5$  đỉnh sẽ gây lỗi `RecursionError` nếu dùng DFS đệ quy mà không set lại `sys.setrecursionlimit()`.

**Câu 16 (C):** `range(5)` là  $0, 1, 2, 3, 4$ . Điều kiện `if x % 2 == 0` chọn  $0, 2, 4$ . Bình phương lên  $\rightarrow [0, 4, 16]$ .

### PHẦN III: TƯ DUY & TÍNH TOÁN (Câu 17 - 26)

**Câu 17 (B):** Chạy mô phỏng (Trace) Bellman-Ford:

- Khởi tạo:  $d[A] = 0, d[B] = \infty, d[C] = \infty$ .
- Xét  $(A, B, 2) : d[B] = \min(\infty, 0 + 2) = 2$ .
- Xét  $(B, C, -5) : d[C] = \min(\infty, 2 - 5) = -3$ .
- Xét  $(C, A, 1) : d[A] = \min(0, -3 + 1) = -2$ .

Vậy sau vòng 1,  $d[A] = -2$ .

**Câu 18 (D):** Đồ thị đầy đủ  $K_n$  luôn cần  $n$  màu vì tất cả các đỉnh đều nối với nhau, không cặp nào chung màu được.

**Câu 19 (B):** Đồ thị hai phía đầy đủ  $K_{m,n}$  có số cạnh là  $m \times n$ .  $3 \times 4 = 12$ .

**Câu 20 (A):** Thuật toán Prim từ A:

- Kè A có  $(A, B, 1), (A, C, 4)$ . Chọn  $(A, B)$  (nhỏ nhất). MST hiện có  $\{A, B\}$ .
- Kè MST hiện tại có  $(A, C, 4), (B, C, 2), (B, D, 6)$ . Chọn  $(B, C)$  (nhỏ nhất là 2). MST có  $\{A, B, C\}$ .
- Kè MST hiện tại có  $(A, C, 4)$  (bỏ vì tạo chu trình),  $(B, D, 6)$ . Chọn  $(B, D)$ .

Thứ tự:  $(A, B) \rightarrow (B, C) \rightarrow (B, D)$ .

**Câu 21 (B):**  $M[i][i] = 1$  nghĩa là có cạnh đi từ đỉnh  $i$  về chính nó, gọi là khuyên (self-loop).

**Câu 22 (C):** Đồ thị đường thẳng  $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ . Khoảng cách là tổng trọng số các cạnh liên tiếp:  $1 + 1 + 1 + 1 = 4$ .

**Câu 23 (A):** Với  $V = 10$ , chỉ cần  $V - 1 = 9$  cạnh là đủ để tạo thành 1 cây liên thông. Đề cho 15 cạnh ( $> 9$ ), nên hoàn toàn có thể nối tất cả thành **1 thành phần liên thông**. (Câu hỏi hỏi số lượng tối thiểu, đáp án là 1).

**Câu 24 (A):** Trên đường tăng luồng, lượng luồng có thể đẩy thêm bị giới hạn bởi cạnh có sức chứa dư nhỏ nhất (Bottleneck).  $\min(3, 5, 2) = 2$ .

**Câu 25 (B):** Bellman-Ford chạy  $V - 1$  vòng để tìm đường ngắn nhất. Nếu chạy thêm vòng thứ  $V$  mà khoảng cách vẫn giảm, chứng tỏ có chu trình âm.

**Câu 26 (B):**  $C_5$  là chu trình lẻ (5 đỉnh). Tô màu xen kẽ: 1(Đỏ)-2(Xanh)-3(Đỏ)-4(Xanh). Đỉnh 5 nối với 1(Đỏ) và 4(Xanh) nên cần màu thứ 3 (Vàng).

## PHẦN IV: ỨNG DỤNG THỰC TẾ (Câu 27 - 40)

**Câu 27 (C):** Các môn thi là đỉnh, nếu có sinh viên thi cả 2 môn thì nối cạnh (xung đột). Bài toán xếp lịch để không trùng nhau chính là tô màu đồ thị sao cho 2 đỉnh kề nhau khác màu (màu = ca thi). Mục tiêu dùng ít màu nhất.

**Câu 28 (C):** Bài toán phân công (Assignment Problem) là dạng điển hình của Cặp ghép cực đại trọng số nhỏ nhất trên đồ thị hai phía (Min-Weight Perfect Matching on Bipartite Graph).

**Câu 29 (B):** Trong sơ đồ PERT, để dự án hoàn thành, tất cả các công việc tiền trạm phải xong. Thời gian sớm nhất để xong dự án bị quyết định bởi chuỗi công việc dài nhất (Đường găng). Vì đồ thị không chu trình (DAG), ta tìm đường đi dài nhất (Critical Path).

**Câu 30 (A):** Hệ thống gợi ý thường mô hình hóa User và Item là 2 tập đỉnh của đồ thị hai phía. Quan hệ "thích" là cạnh nối.

**Câu 31 (A):** Bài toán đi qua mọi con đường (cạnh) rồi quay về: Nếu đồ thị có chu trình Euler, đáp án chính là nó. Nếu không, phải lặp lại một số cạnh (Chinese Postman Problem). Gốc rẽ là Euler.

**Câu 32 (B):** Trong đồ thị cấp phát tài nguyên, nếu xuất hiện một chu trình (tiến trình A giữ tài nguyên 1 chờ 2, B giữ 2 chờ 1), hệ thống sẽ bị khóa chết (Deadlock).

**Câu 33 (A):** OSPF (Open Shortest Path First) là giao thức định tuyến trạng thái liên kết, sử dụng thuật toán **Dijkstra** để tính bảng định tuyến trên mỗi router.

**Câu 34 (A):** Cấu trúc vòng trong hóa học tương đương với chu trình trong đồ thị. BFS/DFS đều phát hiện được chu trình.

**Câu 35 (B):** Quay lui (Backtracking) bản chất là tìm kiếm theo chiều sâu (**DFS**) trên cây trạng thái. Thử một bước, đi sâu xuống, nếu sai thì quay lại (backtrack).

**Câu 36 (A):** Cạnh mà khi xóa đi làm tăng số thành phần liên thông (ngắt đôi đồ thị) gọi là **Cầu** (Bridge). Đỉnh tương tự gọi là **Khớp** (Articulation Point).

**Câu 37 (A):** Khái niệm này nói rằng trung bình bất kỳ 2 người nào trên trái đất cũng kết nối được với nhau qua tối đa khoảng 6 bước nhảy. Điều này ám chỉ **Đường kính** (Diameter) của đồ thị xã hội rất nhỏ ("Thế giới phẳng").

**Câu 38 (B):** Mê cung có trọng số (chi phí đi mỗi bước khác nhau) thì BFS thường không đúng (BFS chỉ đúng với trọng số bằng nhau). Phải dùng **Dijkstra** hoặc **A\*** (nếu có heuristic tọa độ).

**Câu 39 (B):** Union-Find cực kỳ hiệu quả để kiểm tra tính liên thông và phát hiện chu trình khi thêm cạnh, là nòng cốt của thuật toán **Kruskal**.

**Câu 40 (B):** Định nghĩa Cây: Là đồ thị vô hướng **Liên thông** và **Không có chu trình**. (Hết quả là  $E = V - 1$ ).