ĐỀ THI THỬ KẾT THÚC HỌC PHẦN

Môn: Toán học tổ hợp Thực hành

Thời gian: 60 phút | Soạn bởi Cường Võ

I. LÝ THUYẾT & KHÁI NIỆM (20%)

1. Đồ thị đầy đủ (Complete Graph) K_n là đồ thị vô hướng mà mỗi cặp đỉnh đều được nối với nhau bởi một cạnh. Số lượng cạnh của đồ thị K_5 là bao nhiêu?

   A. 5

   B. 10

   C. 20

   D. 25

2. Thuật toán nào sau đây **KHÔNG** đảm bảo tìm được đường đi ngắn nhất trong trường hợp đồ thị có cạnh trọng số âm?

   A. Bellman-Ford

   B. Floyd-Warshall

   C. Dijkstra

   D. SPFA

3. Trong biểu diễn đồ thị bằng **Danh sách kề (Adjacency List)**, tổng độ dài của tất cả các danh sách đối với đồ thị vô hướng có V đỉnh và E cạnh là:

   A. V

   B. E

   C. $2E$

   D. $V + E$

4. Để kiểm tra tính liên thông của một đồ thị vô hướng, thuật toán nào sau đây là phù hợp nhất?

   A. Prim

   B. Dijkstra

   C. BFS hoặc DFS

   D. Floyd-Warshall

5. Đỉnh "nguồn"(Source) và đỉnh "đích"(Sink) là khái niệm quan trọng trong bài toán nào?

   A. Cây khung nhỏ nhất (MST)

B. Luồng cực đại trên mạng (Max Flow)

C. Tìm chu trình Euler

D. Sắp xếp Topo

6. Một cây (Tree) có N đỉnh sẽ luôn có chính xác bao nhiêu cạnh?

A. N

B. $N + 1$

C. $N - 1$

D. $N/2$

7. Độ phức tạp thời gian của thuật toán Floyd-Warshall để tìm đường đi ngắn nhất giữa mọi cặp đỉnh là:

A. $O(V^2)$

B. $O(E \log V)$

C. $O(V^3)$

D. $O(V + E)$

8. Cấu trúc dữ liệu nào thường được sử dụng để cài đặt thuật toán BFS?

A. Stack (Ngăn xếp)

B. Queue (Hàng đợi)

C. Priority Queue (Hàng đợi ưu tiên)

D. Hash Map

II. ĐỌC HIỂU CODE PYTHON (30%)

9. Đoạn code sau in ra kết quả gì?

```
a = [1, 2, 3]
b = a
b.append(4)
print(a)
```

A. `[1, 2, 3]`

B. `[1, 2, 3, 4]`

C. `[4, 1, 2, 3]`

D. Báo lỗi.

10. Trong module `heapq` của Python, hàm `heappop(heap)` sẽ lấy ra phần tử nào?

A. Phần tử lớn nhất.

B. Phần tử nhỏ nhất.

C. Phần tử bất kỳ.

D. Phần tử vừa được thêm vào cuối cùng.

11. Đoạn code khởi tạo ma trận nào sau đây là **ĐÚNG** (không bị lỗi tham chiếu) cho kích thước 3×3?

    A. `[[0] * 3] * 3`

    B. `[[0] * 3 for _ in range(3)]`

    C. `[[0, 0, 0]] * 3`

    D. `[0 * 3] * 3`

12. Kết quả của đoạn code `print(list(range(1, 5)))` là:

    A. `[1, 2, 3, 4, 5]`

    B. `[0, 1, 2, 3, 4]`

    C. `[1, 2, 3, 4]`

    D. `[2, 3, 4, 5]`

13. Mục đích của dòng lệnh `if neighbor not in visited:` trong thuật toán BFS/DFS là:

    A. Để tìm đường ngắn nhất.

    B. Để tránh duyệt lại các đỉnh đã đi qua, gây lặp vô tận.

    C. Để tính trọng số của cạnh.

    D. Để sắp xếp danh sách kề.

14. Cho đoạn code đệ quy. Giá trị trả về của `func(3)` là?

```
def func(n):
    if n == 0: return 0
    return n + func(n-1)
```

    A. 3

    B. 6

    C. 5

    D. Lặp vô tận

15. Để sắp xếp danh sách các cạnh `edges = [(u, v, w), ...]` theo trọng số `w` tăng dần, ta dùng lệnh:

    A. `edges.sort()`

    B. `edges.sort(key=lambda x: x[1])`

    C. `edges.sort(key=lambda x: x[2])`

    D. `edges.reverse()`

16. Cấu trúc `defaultdict(list)` trong Python hữu ích nhất để biểu diễn đồ thị dạng nào?

    A. Ma trận kề.

    B. Danh sách kề.

    C. Danh sách cạnh.

    D. Ma trận trọng số.

17. Biến toàn cục (Global variable) trong Python muốn thay đổi giá trị bên trong hàm thì cần từ khóa gì?

    A. `static`

    B. `extern`

    C. `global`

    D. `public`

18. Khi sử dụng `sys.setrecursionlimit(2000)`, lập trình viên đang muốn giải quyết vấn đề gì?

    A. Tăng tốc độ chạy code.

    B. Giảm bộ nhớ sử dụng.

    C. Tránh lỗi `RecursionError` khi duyệt DFS trên đồ thị sâu.

    D. Tăng độ chính xác của tính toán số thực.

19. Hàm `zip([1, 2], [3, 4])` trong Python sẽ tạo ra iterator tương ứng với list nào?

    A. `[(1, 3), (2, 4)]`

    B. `[(1, 2), (3, 4)]`

    C. `[1, 2, 3, 4]`

    D. `[(1, 4), (2, 3)]`

20. Lệnh `queue.pop(0)` có độ phức tạp thời gian là bao nhiêu (với List thường)?

    A. $O(1)$

    B. $O(n)$

    C. $O(\log n)$

    D. $O(n^2)$

III. TƯ DUY & TÍNH TOÁN (30%)

21. Cho đồ thị vô hướng: `A-B(2), B-C(3), A-C(1)`. MST của đồ thị này có tổng trọng số là:

    A. 3

    B. 4

    C. 5

    D. 6

22. Thứ tự duyệt **BFS** bắt đầu từ đỉnh 0 của đồ thị:
    `0:[1, 2], 1:[0, 3, 4], 2:[0, 5], 3:[1], 4:[1], 5:[2]`. (Ưu tiên số nhỏ trước).

    A. 0, 1, 2, 3, 4, 5

    B. 0, 1, 3, 4, 2, 5

    C. 0, 2, 1, 5, 4, 3

    D. 0, 1, 2, 5, 4, 3

23. Thứ tự duyệt **DFS** (Stack) bắt đầu từ đỉnh 0 của đồ thị trên (theo cách cài đặt phổ biến):

    A. 0, 1, 3, 4, 2, 5

    B. 0, 1, 2, 3, 4, 5

    C. 0, 2, 5, 1, 4, 3

    D. 0, 1, 2, 5, 4, 3

24. Trong thuật toán Dijkstra, tại một bước ta có khoảng cách đến các đỉnh chưa chốt là: `B: 5, C: 2, D: 8`. Đỉnh nào sẽ được chọn để xét tiếp theo?

    A. B

    B. C

    C. D

    D. Bất kỳ

25. Một đồ thị có hướng không có chu trình (DAG) có các cạnh: `A->B, A->C, B->D, C->D`. Thứ tự sắp xếp Topo hợp lệ là:

    A. A, D, B, C

    B. A, B, C, D

    C. D, C, B, A

    D. B, A, C, D

26. Cho đồ thị trọng số: `S->A (4), S->B (2), B->A (1), A->D (2), B->D (5)`. Đường đi ngắn nhất từ S đến D là:

    A. 6

    B. 5

    C. 7

    D. 4

27. Áp dụng thuật toán Kruskal, cạnh có trọng số nhỏ nhất luôn luôn:

    A. Bị loại bỏ.

    B. Được chọn vào cây khung (trừ khi tạo chu trình).

    C. Được chọn sau cùng.

    D. Không quan trọng.

28. Ma trận kề của đồ thị vô hướng có tính chất gì đặc biệt?

    A. Là ma trận đơn vị.

    B. Đối xứng qua đường chéo chính.

    C. Luôn chứa toàn số 1.

    D. Hàng tổng bằng cột tổng.

29. Số lượng thành phần liên thông của đồ thị có các cạnh: `(1-2), (2-3), (4-5)` là bao nhiêu? (Có 5 đỉnh: 1,2,3,4,5)

A. 1

B. 2

C. 3

D. 5

30. Nếu thuật toán Bellman-Ford phát hiện ra một chu trình âm, điều đó có nghĩa là:

A. Không tồn tại đường đi ngắn nhất.

B. Đường đi ngắn nhất bằng 0.

C. Thuật toán đã tính sai.

D. Đồ thị bị ngắt quãng.

31. Đỉnh có bậc (degree) cao nhất trong đồ thị sao (Star Graph) 5 đỉnh là bao nhiêu?

A. 1

B. 2

C. 3

D. 4

32. Trong bài toán Luồng cực đại, nếu Min-Cut có dung lượng là 30, thì Max Flow bằng:

A. 15

B. 30

C. 60

D. 0

IV. ỨNG DỤNG THỰC TẾ (20%)

33. Ứng dụng bản đồ Google Maps tìm đường đi nhanh nhất sử dụng thuật toán nào?

A. DFS

B. Dijkstra / A*

C. Kruskal

D. Prim

34. Để lập lịch thi học kỳ sao cho không bị trùng môn (Bài toán tô màu đồ thị), ta dùng:

A. Dijkstra

B. Topological Sort

C. Prim

D. Floyd-Warshall

35. Virus máy tính lây lan theo tầng từ máy chủ A, mô phỏng bằng:

A. DFS

B. BFS

C. Binary Search

D. Quick Sort

36. Để xây dựng hệ thống đường dây điện nối tất cả các xã với chi phí thấp nhất, ta dùng:

A. Dijkstra

B. Max Flow

C. MST (Prim/Kruskal)

D. Topological Sort

37. Trong game, con ma tự động tìm đường đuổi theo người chơi trong mê cung dùng:

A. BFS

B. Prim

C. Max Flow

D. Sorting

38. Tính toán lưu lượng xe tối đa qua mạng lưới giao thông giờ cao điểm dùng:

A. Shortest Path

B. MST

C. Max Flow Network

D. Graph Coloring

39. Phân tích mạng xã hội, tìm người lạ có mối quan hệ bạn bè chung, ta kiểm tra:

A. Shortest Path

B. Connectivity (Liên thông)

C. MST

D. Topological Sort

40. Chiến thuật thoát mê cung "bám tay phải vào tường"mô phỏng thuật toán:

A. BFS

B. DFS

C. Dijkstra

D. Kruskal

BẢNG ĐÁP ÁN THAM KHẢO

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

GIẢI THÍCH CHI TIẾT

PHẦN I: LÝ THUYẾT

- **Câu 1 (B):** Số cạnh của đồ thị đầy đủ K_n được tính bằng công thức: $\frac{n(n-1)}{2}$. Với $n = 5 \Rightarrow \frac{5\times4}{2} = 10$.
- **Câu 2 (C): Dijkstra** là thuật toán tham lam (greedy), nó giả định rằng khi đã tìm được đường ngắn nhất đến một đỉnh thì đường đó là tối ưu, không quay lại sửa đổi. Trọng số âm làm sai lệch giả định này.
- **Câu 3 (C):** Trong đồ thị vô hướng, cạnh (u, v) được lưu 2 lần: một lần trong danh sách của u (chứa v) và một lần trong danh sách của v (chứa u). Do đó tổng độ dài là $2E$.
- **Câu 4 (C):** BFS hoặc DFS duyệt qua toàn bộ các đỉnh có thể đến được từ đỉnh xuất phát. Nếu sau khi duyệt mà vẫn còn đỉnh chưa thăm → Đồ thị không liên thông.
- **Câu 5 (B):** Bài toán Luồng cực đại (Max Flow) tìm lượng luồng lớn nhất chuyển từ Nguồn (Source - nơi phát sinh) đến Đích (Sink - nơi tiêu thụ).
- **Câu 6 (C):** Đây là định lý cơ bản của cây. Cây là đồ thị liên thông nhỏ nhất. V đỉnh luôn nối với nhau bằng $V - 1$ cạnh.
- **Câu 7 (C):** Floyd-Warshall dùng 3 vòng lặp lồng nhau (cho k, cho i, cho j) để cập nhật ma trận khoảng cách. Độ phức tạp là $O(V^3)$.
- **Câu 8 (B): BFS** duyệt theo chiều rộng (lớp này đến lớp khác), tuân theo nguyên tắc "Vào trước ra trước"(FIFO) nên dùng **Queue**.

PHẦN II: CODE PYTHON

- **Câu 9 (B):** Lệnh ‘b = a‘ không tạo list mới mà chỉ tạo thêm một tham chiếu trỏ vào cùng một list trong bộ nhớ. Thay đổi ‘b‘ thì ‘a‘ cũng thay đổi.
- **Câu 10 (B):** ‘heapq‘ trong Python cài đặt Min-Heap (Vun đống nhỏ nhất), nên phần tử gốc (lấy ra bởi pop) luôn là phần tử nhỏ nhất.
- **Câu 11 (B):** List comprehension tạo ra các đối tượng list con độc lập. Các cách khác (‘*‘) chỉ sao chép tham chiếu, dẫn đến lỗi sửa 1 hàng đổi tất cả các hàng.
- **Câu 12 (C):** Hàm ‘range(start, end)‘ chạy từ ‘start‘ đến ‘end - 1‘.
- **Câu 13 (B):** Trong đồ thị có chu trình, nếu không đánh dấu ‘visited‘, thuật toán sẽ đi vòng tròn mãi mãi (A->B->A->...).
- **Câu 14 (B):** Trace: ‘func(3) = 3 + func(2) = 3 + 2 + func(1) = 3 + 2 + 1 + 0 = 6‘.
- **Câu 15 (C):** ‘x‘ đại diện cho một phần tử ‘(u, v, w)‘. ‘x[2]‘ chính là trọng số ‘w‘.
- **Câu 16 (B):** ‘defaultdict(list)‘ giúp ta thêm cạnh ‘graph[u].append(v)‘ mà không cần kiểm tra xem ‘u‘ đã có trong dict chưa. Đây là cách chuẩn để cài Danh sách kề.

- **Câu 17 (C):** Để sửa biến toàn cục bên trong hàm, bắt buộc phải khai báo 'global variable$_name$'.
- **Câu 18 (C):** Mặc định Python giới hạn đệ quy khoảng 1000 lớp. DFS trên đồ thị sâu (VD: 2000 đỉnh thẳng hàng) sẽ bị lỗi nếu không tăng giới hạn này.
- **Câu 19 (A):** 'zip' ghép cặp phần tử thứ nhất với thứ nhất, thứ hai với thứ hai... $\rightarrow (1, 3)$ và $(2, 4)$.
- **Câu 20 (B):** Khi xóa phần tử đầu list ('index 0'), Python phải dời tất cả các phần tử phía sau lên một hạng. Tốn $O(n)$.

PHẦN III: TÍNH TOÁN

- **Câu 21 (A):** Sắp xếp cạnh: (A-C: 1), (A-B: 2), (B-C: 3). Chọn (A-C), chọn (A-B). Lúc này 3 đỉnh đã nối liền. Tổng $= 1 + 2 = 3$. Không chọn (B-C) vì tạo chu trình.
- **Câu 22 (A):** Queue: [0] $\rightarrow$ ra 0, nạp 1, 2 (Q: [1, 2]) $\rightarrow$ ra 1, nạp 3, 4 (Q: [2, 3, 4]) $\rightarrow$ ra 2, nạp 5 (Q: [3, 4, 5]) $\rightarrow$ lần lượt ra 3, 4, 5.
- **Câu 23 (A):** (Theo quy ước duyệt số nhỏ trước): Từ 0 qua 1. Từ 1 qua 0 (bỏ), 3. Từ 3 qua 1(bỏ) $\rightarrow$ Hết đường, lùi về 1. Từ 1 qua 4. Hết, lùi về 1, lùi về 0. Từ 0 qua 2. Từ 2 qua 5.
- **Câu 24 (B):** Dijkstra luôn chọn đỉnh có nhãn khoảng cách nhỏ nhất trong số các đỉnh chưa chốt. $\min(5, 2, 8) = 2$ (Đỉnh C).
- **Câu 25 (B):** A không phụ thuộc ai $\rightarrow$ A đầu. B và C phụ thuộc A. D phụ thuộc B, C $\rightarrow$ D cuối. Thứ tự A, B, C, D hợp lý.
- **Câu 26 (B):**
  - Đường $S \rightarrow A \rightarrow D$: $4 + 2 = 6$.
  - Đường $S \rightarrow B \rightarrow D$: $2 + 5 = 7$.
  - Đường $S \rightarrow B \rightarrow A \rightarrow D$: $2 + 1 + 2 = 5$. (Ngắn nhất).
- **Câu 27 (B):** Cạnh nhỏ nhất toàn cục không bao giờ tạo chu trình với chính nó (vì nó là cạnh đầu tiên hoặc cạnh nối 2 cây rời rạc), nên luôn được chọn.
- **Câu 28 (B):** Nếu có cạnh nối A với B, thì ma trận tại dòng A cột B là 1, và dòng B cột A cũng là 1. $\rightarrow$ Đối xứng qua đường chéo chính.
- **Câu 29 (B):** Nhóm 1: 1, 2, 3. Nhóm 2: 4, 5. Tổng cộng 2 thành phần liên thông.
- **Câu 30 (A):** Nếu có chu trình âm, ta có thể đi vòng quanh chu trình đó vô hạn lần để giảm tổng trọng số xuống âm vô cùng $\rightarrow$ Không tồn tại đường ngắn nhất thực tế.
- **Câu 31 (D):** Đồ thị sao S_5 có 1 đỉnh trung tâm nối với 4 đỉnh vệ tinh. Bậc đỉnh trung tâm là $5 - 1 = 4$.
- **Câu 32 (B):** Theo định lý Max-Flow Min-Cut: Giá trị luồng cực đại đúng bằng dung lượng lát cắt nhỏ nhất.

PHẦN IV: ỨNG DỤNG

- **Câu 33 (B):** Tìm đường đi trên bản đồ số thực tế luôn dùng Dijkstra hoặc A* (cải tiến của Dijkstra có heuristic hướng đích).
- **Câu 34 (B):** Các môn học có điều kiện tiên quyết (Học A xong mới được học B) tạo thành đồ thị có hướng. Sắp xếp Topo đưa ra thứ tự học hợp lý.
- **Câu 35 (B):** Lây lan theo "tầng"(F0, F1, F2...) chính là tư tưởng duyệt theo lớp của BFS.
- **Câu 36 (C):** Bài toán "Nối tất cả các điểm"với "Chi phí thấp nhất"chính là định nghĩa của Cây khung nhỏ nhất (MST).
- **Câu 37 (A):** Trong lưới ô vuông không trọng số (mê cung đơn giản), BFS tìm đường ngắn nhất (ít bước đi nhất).
- **Câu 38 (C):** Xe cộ = Luồng, Đường = Ống nước, Sức chứa đường = Dung lượng $\rightarrow$ Bài toán Luồng cực đại.
- **Câu 39 (B):** Nếu A và B thuộc cùng một "Thành phần liên thông", tức là có đường dây liên hệ (bạn của bạn...) dẫn từ A đến B.
- **Câu 40 (B):** "Bám tay phải"là đi sâu mãi theo một phía bức tường, nếu tắc thì quay lại. Đây là hành vi của DFS.