Câu 1:

Đồ thị 1: 0-1 0-2 0-3 1-3 1-4 2-5 2-9 3-6 4-7 4-8 5-8 5-9 6-7 6-9 7-8

Câu 2:

số lượng đồ thị vô hướng khác nhau với *V* đỉnh và *E* cạnh là tổ hợp chập E của (V2)

Câu 4

Để chứng minh rằng một đồ thị là đồ thị hai màu (bipartite) khi và chỉ khi nó không chứa chu trình độ dài lẻ, ta sẽ sử dụng phương pháp chứng minh bằng phản chứng.

**Phần 1: Nếu đồ thị là hai màu thì nó không chứa chu trình độ dài lẻ**

Giả sử bạn có một đồ thị hai màu. Nếu có một chu trình trong đồ thị này, chu trình đó phải xen kẽ giữa các đỉnh thuộc hai nhóm màu khác nhau. Bởi vì không có hai đỉnh cùng màu nào được nối với nhau, chu trình phải "nhảy" từ một nhóm màu sang nhóm màu kia sau mỗi cạnh. Điều này có nghĩa là chu trình phải có độ dài chẵn, vì nó sẽ kết thúc ở nhóm màu bắt đầu. Do đó, không thể có chu trình độ dài lẻ trong một đồ thị hai màu.

**Phần 2: Nếu đồ thị không chứa chu trình độ dài lẻ thì nó là hai màu**

Giả sử bạn có một đồ thị không chứa chu trình độ dài lẻ nhưng không phải là đồ thị hai màu. Điều này có nghĩa là có ít nhất một cạnh trong đồ thị nối hai đỉnh cùng màu.

Tuy nhiên, nếu bạn theo dõi đường đi từ một đỉnh qua cạnh này đến đỉnh kia và quay trở lại đỉnh ban đầu qua các cạnh khác (điều này luôn có thể xảy ra vì đồ thị là liên thông), bạn sẽ tạo thành một chu trình. Vì hai đỉnh cuối cùng này cùng màu, chu trình này sẽ có độ dài lẻ (vì nó phải "nhảy" số lẻ lần giữa các màu để quay lại một đỉnh cùng màu). Điều này mâu thuẫn với giả thiết ban đầu rằng đồ thị không chứa chu trình độ dài lẻ. Do đó, mọi đồ thị không chứa chu trình độ dài lẻ đều phải là đồ thị hai màu.

Như vậy, ta đã chứng minh rằng một đồ thị là đồ thị hai màu khi và chỉ khi nó không chứa chu trình độ dài lẻ.

Câu 5

Chứng minh rằng một đồ thị không có điểm articulation là một đồ thị biconnected

Bây giờ, chúng ta sẽ chứng minh rằng một đồ thị không có điểm articulation là một đồ thị biconnected:

* **Bước 1**: Xét một đồ thị *G* không có điểm articulation. Điều này có nghĩa là việc loại bỏ bất kỳ đỉnh nào cũng không làm mất tính liên thông của đồ thị.
* **Bước 2**: Lấy hai đỉnh bất kỳ *s* và *t* trong đồ thị. Vì *G* là liên thông, tồn tại ít nhất một đường đi từ *s* đến *t*.
* **Bước 3**: Giả sử có một đường đi *P* từ *s* đến *t*. Vì không có điểm articulation, việc loại bỏ bất kỳ đỉnh nào trên đường đi *P* không làm mất tính liên thông của đồ thị.
* **Bước 4**: Nếu ta loại bỏ một đỉnh *v* trên đường đi *P*, vẫn phải tồn tại một đường đi khác từ *s* đến *t* không qua *v* (do tính liên thông không bị mất).
* **Bước 5**: Áp dụng luận điểm trên cho mỗi đỉnh trên đường đi *P*, ta có thể tìm ra một đường đi khác từ *s* đến *t* không giao nhau với *P*.

Kết luận: Từ các bước trên, ta thấy rằng cho mọi cặp đỉnh *s* và *t* trong *G*, luôn tồn tại ít nhất hai đường đi không giao nhau nối *s* và *t*. Do đó, *G* là một đồ thị biconnected. Điều này chứng minh rằng mọi đồ thị không có điểm articulation đều là đồ thị biconnected.

Câu 8

BFS đánh dấu các đỉnh theo khoảng cách tăng dần từ đỉnh nguồn của nó, nhưng điều này không đảm bảo rằng đỉnh nào đó sẽ nằm trước đỉnh khác trong thứ tự tô pô. Điều này là do BFS không nhất thiết phải thăm tất cả các đỉnh của một cấp trước khi chuyển sang cấp tiếp theo.

Nếu đồ thị chứa chu trình, thì thuật toán BFS không thể tạo ra một thứ tự tô pô đúng, vì thứ tự này yêu cầu không có chu trình. Điều này có thể xảy ra khi có các cạnh quay trở lại các đỉnh đã thăm, và BFS không ngăn chặn điều này.

Câu 10

Thuật toán thời gian tuyến tính để tính thành phần liên thông mạnh chứa một đỉnh v cho trước dựa trên DFS như sau:

1. Thực hiện DFS trên đồ thị ban đầu, bắt đầu từ đỉnh v.
2. Thực hiện DFS trên đồ thị chuyển vị của đồ thị ban đầu, bắt đầu từ đỉnh v.
3. Các đỉnh được thăm trong lần duyệt thứ hai của DFS sẽ tạo thành thành phần liên thông mạnh chứa đỉnh v.

[Để tính các thành phần liên thông mạnh của một đồ thị có hướng, ta có thể sử dụng thuật toán Kosaraju](https://vi.wikipedia.org/wiki/Th%C3%A0nh_ph%E1%BA%A7n_li%C3%AAn_th%C3%B4ng_m%E1%BA%A1nh).

**Bước 1: DFS trên đồ thị gốc:**

* + Thực hiện DFS trên đồ thị ban đầu và lưu thời gian kết thúc của mỗi đỉnh.
  + Sử dụng một stack để lưu các đỉnh theo thứ tự thời gian kết thúc giảm dần.

**Bước 2:Đảo ngược đồ thị (Reverse Graph):**

* + Tạo một đồ thị mới là đồ thị đảo ngược của đồ thị ban đầu. Điều này có nghĩa là đảo chiều mỗi cạnh trong đồ thị.

**Bước 3: DFS trên đồ thị đảo ngược:**

* + Sử dụng thứ tự thời gian kết thúc từ Bước 1 để chọn đỉnh xuất phát cho mỗi lần thăm.
  + Mỗi lần thăm một đỉnh, thực hiện DFS để xác định một thành phần liên thông mạnh.

Câu 13

Với *V*=1, chỉ có một đỉnh và không có cạnh nào. Điều này thoả mãn điều kiện không có cạnh song song.

Giả sử rằng mọi đồ thị có hướng với *V*=*k* đỉnh không chứa cạnh song song.

Xét đồ thị có hướng *G* với *V*=*k*+1 đỉnh.

Chúng ta có thể thêm một đỉnh mới vào *G*, tức là *k*+1-ésimo đỉnh. Có *k* cách để thêm một cạnh từ đỉnh mới này đến mỗi đỉnh khác trong *G* (đó là *k* cạnh đi vào đỉnh mới). Tổng cộng có 2*k* cạnh đi vào đỉnh mới.

Vì mọi đồ thị có hướng với *V*=*k* đỉnh không chứa cạnh song song, nên có 2*k ^2* đồ thị khác nhau với *V*=*k* đỉnh. Do đó, có 2*k*×2*k ^2* =2(*k*+1) ^2 đồ thị khác nhau với *V*=*k*+1 đỉnh.

Vậy, theo nguyên tắc quy nạp, mọi đồ thị có hướng *V* đỉnh không chứa cạnh song song.

Câu 18

Giả sử ta có một cây bao trùm *T* không min được từ thuật toán Boruvka.

Bây giờ, xét một lần lặp cụ thể khi một cạnh không được thêm vào *T*, điều này chỉ xảy ra nếu cạnh đó đã được xóa do đã tạo chu trình với cây hiện tại.

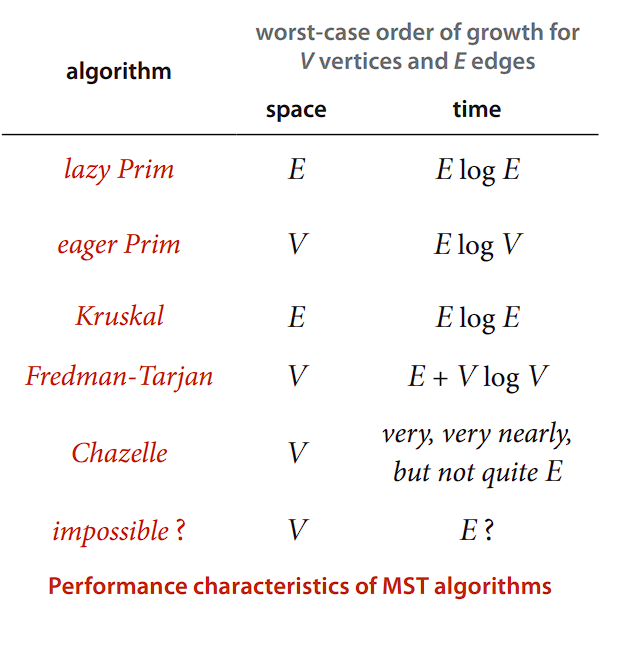
Nếu cây *T* không phải là MST, thì chắc chắn rằng có một cây MST *T*′ mà *T*′ có một cạnh có trọng số nhỏ hơn một trong những cạnh đã được xóa. Khi chúng ta thêm cạnh này vào *T*′, nó tạo ra một chu trình với một cạnh đã xóa. Điều này là không thể, vì chúng ta đã kiểm tra và xóa các cạnh này để đảm bảo không tạo chu trình.

Vậy nên, nếu *T* không phải MST, tồn tại một cây MST mà Boruvka không thể thêm vào. Điều này là mâu thuẫn, vì nếu có MST thì Boruvka chắc chắn sẽ tìm thấy nó.

Độ phức tạp của thuật toán trên phụ thuộc vào cách thức triển khai cụ thể và cách thức thực hiện xóa cạnh và kiểm tra liên thông. Trong trường hợp tồi nhất, độ phức tạp có thể là *O*(*E*log*V*) với *E* là số lượng cạnh và *V* là số lượng đỉnh. Điều này có thể xuất phát từ việc sắp xếp trọng số cạnh.

Chú ý rằng độ phức tạp này tốt hơn so với Kruskal's, đặc biệt là khi số lượng cạnh *E* lớn hơn và thường sử dụng cho đồ thị có cấu trúc đặc biệt như đồ thị thưa.

Câu 22



Câu 26

Để chứng minh rằng có thể tính các đường đi ngắn nhất trong một đồ thị có hướng với các trọng số không âm tại các đỉnh, ta có thể chuyển đổi vấn đề này thành việc tìm đường đi ngắn nhất trong một đồ thị có hướng với trọng số cạnh. Dưới đây là cách thức xây dựng đồ thị mới từ đồ thị ban đầu:

Giả sử chúng ta có một đồ thị có hướng  G = (V, E)  với tập hợp các đỉnh ( V ) và tập hợp các cạnh E . Mỗi đỉnh v trong V  có một trọng số không âm w(v).

Xây Dựng Đồ Thị Mới: Tạo một đồ thị mới  G' = (V, E') với cùng tập hợp đỉnh ( V ). Tuy nhiên, trọng số của các cạnh trong E' sẽ được xác định theo cách sau:

    - Với mỗi cạnh (u,v) thuộc E trong đồ thị ban đầu, chúng ta sẽ tạo một cạnh tương ứng trong E với trọng số là w'(u, v) = w(v). Điều này có nghĩa là trọng số của mỗi cạnh mới sẽ tương đương với trọng số của đỉnh đích trong đồ thị ban đầu.

Bây giờ, chúng ta có thể sử dụng các thuật toán tìm đường đi ngắn nhất trên đồ thị G’. Ví dụ, thuật toán Dijkstra hoặc Bellman-Ford có thể được áp dụng vì các trọng số cạnh trong G' không âm.

Kết quả tìm được trên G' sẽ tương ứng với đường đi ngắn nhất trên G dựa vào các đỉnh. Trọng số của một đường đi trong G sẽ tương đương với tổng trọng số các đỉnh trên đường đi đó.