Bộ bài tập do Samsung tài trợ
Năm học 2023-2024

*Chủ đề **Lý thuyết đồ thị***

Lưu ý: Trong các bài dưới đây, nếu không có định nghĩa cụ thể tại từng bài thì V là số đỉnh và E là số cạnh của đồ thị.

A. Đồ thị vô hướng

1. *Chu trình Euler và chu trình Hamilton.*
   Chu trình Euler là chu trình đi qua tất cả các cạnh, mỗi cạnh đúng một lần. Chu trình Hamilton là chu trình đi qua tất cả các đỉnh, mỗi đỉnh đúng một lần.
   Xét các đồ thị được cho bởi bốn tập cạnh sau:
   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
   0-1 0-2 0-3 1-3 0-3 2-5 5-6 3-6 4-7 4-8 5-8 5-9 6-7 6-9 8-8
   0-1 1-2 1-3 0-3 0-4 2-5 2-9 3-6 4-7 4-8 5-8 5-9 6-7 6-9 7-8
   4-1 7-9 6-2 7-3 5-0 0-2 0-8 1-6 3-9 6-3 2-8 1-5 9-8 4-5 4-7
   Đồ thị nào có chu trình Euler? Đồ thị nào có chu trình Hamilton?

2. *Đếm đồ thị*
   Có bao nhiêu đồ thị vô hướng khác nhau có V đỉnh và E cạnh (không có cạnh song song)?

3. *Phát hiện cạnh song song*
   Thiết kế một thuật toán tuyến tính để đếm số cạnh song song trong một đồ thị

4. *Chu trình lẻ*
   Chứng minh rằng một đồ thị là đồ thị hai mầu (bipartite) khi và chỉ khi nó không chứa chu trình độ dài lẻ. Đồ thị hai mầu là đồ thị mà có thể dùng hai mầu để tô mỗi đỉnh một mầu sao cho không có cạnh nào nối giữa hai đỉnh cùng mầu.
   Gợi ý: chứng minh bằng phản chứng.

5. *Biconnected*
   Một đồ thị được gọi là biconnected nếu mỗi cặp đỉnh đều được nối với nhau bởi hai đường đi không giao nhau. Trong đồ thị liên thông, điểm articulation là đỉnh mà khi xóa nó và các cạnh kề sẽ làm đồ thị mất tính liên thông. Hãy chứng minh rằng một đồ thị bất kì mà không có điểm articulation là đồ thị biconnected.
   Gợi ý: cho một cặp đỉnh s và t và một đường đi nối giữa chúng. Hãy sử dụng dữ kiện rằng không có đỉnh nào trên đường đi đó là điểm articulation để xây dựng hai đường đi không giao nhau nối s và t.

6. *Liên thông cạnh*
   Trong một đồ thị liên thông, cầu là một cạnh mà nếu xóa cạnh đó thì đồ thị đó bị tách thành hai đồ thị con không giao nhau. Một đồ thị không có cầu được gọi là liên thông cạnh (edge connected). Hãy cài một thuật toán xác định xem một đồ thị có tính chất liên thông cạnh hay không.
   Gợi ý: dựa DFS

7. *Xử lý ảnh*
   Hãy cài thuật toán floodfill trên ảnh để xác định các điểm ảnh kề nhau có cùng mầu.
   Gợi ý: dựa DFS hoặc BFS

B. Đồ thị có hướng

8. *Sắp xếp tô pô và BFS*
   Giải thích vì sao thuật toán sau không đảm bảo cho kết quả là một thứ tự tô pô: Chạy BFS, đánh dấu mỗi đỉnh theo khoảng cách tăng dần tới đỉnh nguồn của nó.

9. *Chu trình Euler có hướng*
   Viết một mô đun chương trình Euler nhận một đồ thị có hướng làm input và và tìm chu trình Euler trong đồ thị đó hoặc báo là không tồn tại.
   Gợi ý: Hãy chứng minh rằng một đồ thị có hướng G có một chu trình Euler khi và chỉ khi G liên thông và mỗi đỉnh đều có bậc ra bằng bậc vào.

10. *Liên thông mạnh*
    Hãy mô tả một thuật toán thời gian tuyến tính tính thành phần liên thông mạnh chứa một đỉnh v cho trước. Dựa trên thuật toán đó, hãy mô tả một thuật toán thời gian bậc hai đơn giản để tính các thành phần liên thông mạnh của một đồ thị có hướng.
    Gợi ý: dựa DFS

11. *Đường đi Hamilton trong đồ thị có hướng phi chu trình*
    Cho một đồ thị có hướng phi chu trình (DAG), thiết kế một thuật toán thời gian tuyến tính để xác định xem có tồn tại một đường đi có hướng đi qua mỗi đỉnh đúng một lần hay không.
    Gợi ý: Tính một sắp xếp tô pô và xem nếu có một cạnh nối giữa một cặp đỉnh liên tiếp trong thứ tự tô pô

12. *Thứ tự tô pô duy nhất*
    Thiết kế một thuật toán xác định xem một đồ thị có thứ tự tô pô duy nhất hay không.
    Gợi ý: một đồ thị có hướng có thứ tự tô pô duy nhất khi và chỉ khi có một cạnh có hướng nối giữa mỗi cặp đỉnh liên tiếp trong thứ tự tô pô (nghĩa là khi đồ thị có một đường đi Hamilton). Nếu đồ thị có hướng có nhiều thứ tự tô pô thì thứ tự tô pô thứ hai có thể thu được từ thứ tự thứ nhất bằng cách đổi vị trí một cặp đỉnh liên tiếp.

13. *Đếm đồ thị có hướng.*

Chứng minh rằng có tất cả 2^{V^2} đồ thị V đỉnh có hướng không chứa cạnh song song.
Gợi ý: Có bao nhiêu đồ thị có hướng chứa V đỉnh và E cạnh?

14. Đếm đồ thị có hướng phi chu trình
Có bao nhiêu đồ thị có hướng phi chu trình chứa V đỉnh?

15. Sắp xếp tô pô dùng hàng đợi.
Hãy cài thuật toán sắp xếp tô pô và sử dụng một mảng dùng chỉ số đỉnh để lưu giữ bậc vào (indegree) của mỗi đỉnh. Khởi tạo mảng và một hàng đợi chứa các đỉnh nguồn để duyệt một lần qua tất cả các cạnh. Sau đó thực hiện các bước sau cho đến khi hàng đợi rỗng:

- Lấy một đỉnh nguồn ra khỏi hàng đợi và đánh nhãn cho nó
- Với mỗi cạnh của đỉnh vừa được lấy ra, giảm phần tử mảng indegree ứng với đỉnh đích của cạnh đó đi 1
- Nếu phần tử mảng nào bị giảm về 0 thì thêm đỉnh ứng với ô đó vào hàng đợi.

C. Cây bao trùm nhỏ nhất

16. Rừng bao trùm nhỏ nhất
Hãy phát triển các phiên bản của các thuật toán Prim và Kruskal để tính rừng bao trùm nhỏ nhất trên đồ thị có trọng số mà không nhất thiết là đồ thị liên thông.

17. Thuật toán Vyssotsky
Hãy cài thuật toán tính cây bao trùm nhỏ nhất bằng cách áp dụng lặp đi lặp lại tính chất chu trình: Thêm từng cạnh vào cây, mỗi lần nếu phát sinh chu trình thì xóa cạnh có trọng số lớn nhất của chu trình đó.

18. Thuật toán xóa ngược
Cài một thuật toán tính cây bao trùm min theo cách sau: Bắt đầu bằng đồ thị chứa tất cả các cạnh. Sau đó duyệt các cạnh theo thứ tự trọng số giảm dần, với mỗi cạnh, kiểm tra xem xóa cạnh đó có làm đồ thị mất liên thông hay không, nếu không thì xóa nó đi. Hãy chứng minh rằng thuật toán này tính ra cây bao trùm min. Xác định độ phức tạp của thuật toán theo số bước so sánh trọng số cạnh.

19. Cạnh khó
Trong một cây bao trùm min, một cạnh khó (critical edge) là cạnh mà nếu xóa nó khỏi đồ thị thì sẽ làm tăng trọng số cây bao trùm min của đồ thị. Hãy trình bày cách tìm tất cả các cạnh khó trong một đồ thị trong thời gian tỷ lệ thuận với $E\log_2 E$.
Lưu ý: Có thể có các cạnh cùng trọng số (nếu không thì cạnh nào của cây bao trùm cũng là cạnh khó).

20. Hoạt hình
Viết một chương trình dùng đồ họa minh họa quá trình dựng cây bao trùm min.

21. Tập cạnh cho trước

Cho một đồ thị có trọng số G và một tập cạnh S của đồ thị (không chứa chu trình). Hãy mô tả thuật toán tìm cây bao trùm min của G có chứa tất cả các cạnh trong tập S

22. Prim vs. Kruskal

Hãy thực hiện thí nghiệm đo thời gian chạy thuật toán để so sánh hiệu năng giữa các phiên bản lười và bản dùng queue của Prim và thuật toán Kruskal

23. Thuật toán Boruvka

Cài thuật toán Boruvka tính cây bao trùm min bằng cách thêm dần từng cạnh vào một rừng gồm các cây. Mỗi lần, lấy cạnh có trọng số nhỏ nhất mà nối một cây với một cây khác và kết nạp cạnh đó vào cây bao trùm. Giả sử các cạnh có trọng số khác nhau.
Gợi ý: Sử dụng cấu trúc dữ liệu union-find.

24. Thuật toán Boruvka cải tiến

Cải tiến cài đặt trên bằng cách sử dụng danh sách liên kết đôi nối vòng tròn để biểu diễn các cây con của cây bao trùm min sao cho có thể merge các cây con và giữ thời gian tỷ lệ thuận với E ở mỗi lần lặp (khi đó không cần dùng cấu trúc union-find)

25. MST đồ thị lớn

Hãy tìm cách tính cây bao trùm min của đồ thị với kích thước lớn đến mức tại mỗi thời điểm chương trình chỉ có đủ bộ nhớ để lưu V cạnh.
Gợi ý: Áp dụng thuật toán Prim.

D. Đường đi ngắn nhất

26. Trọng số đỉnh

Hãy chứng tỏ rằng có thể tính các đường đi ngắn nhất trong một đồ thị có hướng với các trọng số không âm tại các đỉnh (trong đó trọng số của một đường đi là tổng các trọng số của các đỉnh) bằng cách xây dựng một đồ thị có hướng với trọng số cạnh.

27. Đường đi ngắn nhất đa nguồn

Hãy cài một mô đun chương trình áp dụng thuật toán Dijkstra để giải bài toán đường đi ngắn nhất đa nguồn trên đồ thị có hướng với cạnh có trọng số dương: Cho một tập đỉnh nguồn, tìm một rừng các đường đi ngắn nhất. Sau đó cung cấp hàm API trả về đường đi ngắn nhất từ nguồn bất kì tới mỗi đỉnh
Gợi ý: thêm một đỉnh phụ với cạnh trọng số 0 nối tới mỗi đỉnh nguồn, hoặc khởi tạo hàng đợi ưu tiên chứa tất cả các đỉnh nguồn

28. Đường đi ngắn nhất giữa hai tập đỉnh

Cho một đồ thị có hướng trọng số cạnh dương, và hai tập đỉnh phân biệt S và T. Hãy tìm đường đi ngắn nhất từ một đỉnh bất kỳ thuộc S tới một đỉnh bất kỳ thuộc T. Thuật toán của bạn cần chạy với thời gian tỷ lệ thuận với $E \log_2 V$

Gợi ý: đỉnh phụ hoặc khởi tạo hàng đợi ưu tiên chứa sẵn S

29. Đường đi ngắn nhất trong các đồ thị Euclid

Hãy chỉnh thuật toán Dijkstra để tăng tốc độ cho các đồ thị với các đỉnh là các điểm trên mặt phẳng hình học Euclid

30. Đường đi dài nhất tại DAG

Cài thuật toán tìm đường đi dài nhất trong đồ thị có hướng phi chu trình có trọng số.
Gợi ý: Dựa Dijkstra

31. Tất cả các đường đi ngắn nhất trong đồ thị có chu trình âm

Hãy chạy một phiên bản của thuật toán Bellman-Ford để xác định các trọng số pi[v] sao cho với mỗi cạnh v→w, trọng số cạnh cộng với hiệu giữa pi[v] và pi[w] là không âm. Sau đó sử dụng các trọng số này để đánh lại trọng số đồ thị, sao cho có thể sử dụng thuật toán Dijkstra để tìm tất cả các đường đi ngắn nhất trong đồ thị mới.

32. Tất cả các đường đi ngắn nhất trên một trục

Cho một đồ thị trục có trọng số (đồ thị vô hướng liên thông, mỗi đỉnh đều có bậc bằng 2 ngoại trừ hai đỉnh đầu và cuối có bậc 1), hãy thiết kế một thuật toán tiền xử lý đồ thị trong thời gian tuyến tính và có thể trả về khoảng cách của đường đi ngắn nhất giữa hai đỉnh bất kì trong thời gian hằng số.
Gợi ý: khi tiền xử lý thì tính tất cả các đường đi ngắn nhất để khi chạy hàm lấy khoảng cách giữa hai đỉnh thì chỉ trả về kết quả đã tính.

33. Đường đi ngắn nhất trên lưới vuông

Cho một ma trận NxN gồm các số nguyên dương, hãy tìm đường đi ngắn nhất từ ô (0,0) đến ô (N-1, N-1), trong đó độ dài đường đi là tổng các số nguyên trên đường đi.
Hãy làm lại bài toán với giả thiết bổ sung là bạn chỉ có thể đi sang phải hoặc đi xuống.

34. Đường đi đơn điệu ngắn nhất - Monotonic shortest path

Cho một đồ thị có trọng số, tìm một đường đi đơn điệu ngắn nhất từ đỉnh s tới mỗi đỉnh còn lại. Một đường đi được gọi là đơn điệu nếu trọng số của mỗi cạnh trên đường đi tăng dần hoặc giảm dần, đường đi phải là đường đi đơn (không lặp đỉnh)
Gợi ý: Hãy dãn cạnh theo thứ tự tăng dần của trọng số và tìm một đường đi tốt nhất; sau đó dãn cạnh theo thứ tự giảm dần của trọng số và tìm một đường đi tốt nhất.

35. Đường đi bitonic ngắn nhất - Bitonic shortest path

Cho một đồ thị có trọng số, tìm một đường đi bitonic ngắn nhất từ đỉnh s tới mỗi đỉnh còn lại (nếu có). Một đường đi từ s tới t được gọi là bitonic nếu như nó chứa một đỉnh trung gian x sao cho các cạnh trên đường đi từ s tới x có trọng số tăng dần, còn các cạnh trên đường đi từ x tới t có trọng số giảm dần. Đường đi không được lặp đỉnh.

Gợi ý: Với mỗi đỉnh, tìm đường đi bitonic ngắn nhất đến đó sao cho các cạnh trọng số đang tăng dần khi đến đỉnh đó bằng thuật toán Dijkstra. Từ đó, với mỗi đỉnh, tìm đường

đi bitonic ngắn nhất đến đó, sao cho các cạnh trọng số đã hết tăng và đang giảm dần khi đến đỉnh đó, cũng sử dụng thuật toán Dijkstra.

36. *Cạnh khó - critical edge*
    Phát triển một thuật toán với input gồm một đồ thị có hướng có trọng số và một cặp đỉnh s, t, thuật toán tìm một cạnh mà việc xóa bỏ cạnh đó sẽ làm tăng mạnh nhất độ dài đường đi ngắn nhất từ s đến t.

37. *Bellman-Ford nhanh*
    Hãy phát triển một thuật toán có thời gian chạy tốt hơn linearithmic cho bài toán đường đi ngắn nhất từ một nguồn cho đồ thị có hướng có trọng số với giả thiết bổ sung là các trọng số là các số nguyên có giá trị tuyệt đối không vượt quá hằng số C
    Gợi ý: Áp dụng thuật toán Dijkstra. Lợi dụng tính chất là khoảng cách lớn nhất giữa độ dài các đường đi đến các đỉnh đã thăm nhưng chưa tối ưu không thể vượt quá C, bằng cách thay vì sử dụng cấu trúc heap để quản lý các đỉnh, có thể sử dụng một mảng độ dài tối đa C, đánh số theo độ dài đường đi, và xếp các đỉnh vào vị trí tương ứng.

38. *Hoạt hình*
    Viết một chương trình dùng đồ họa minh họa quá trình dựng cây đường đi ngắn nhất theo thuật toán Dijkstra.

Tài liệu tham khảo:

1. Robert Sedgewick and Kevin Wayne, Algorithms, 4th edition.
2. Kenneth H. Rosen, Graph Theory and Its Applications, 2nd edition.