A

Đọc đề, chúng ta đều nghĩ đến solution đơn giản nhất:

Thử bỏ một cạnh, sau đó chạy dfs để kiểm tra xem đồ thị có chu trình hay không.

Để kiểm tra một đồ thị có chu trình hay không, đơn giản chúng ta chỉ cần duy trì nhãn $mark[u]$ với ý nghĩa u đã được thăm trước đó, và có đường đi từ u đến đỉnh đang xét hiện tại. Nếu đang thăm các cạnh kề của u. chúng ta thăm đến v, mà v đã được gán nhãn -> có chu trình.

Do số cạnh khá lớn, chúng ta không thể xét toàn bộ cạnh được. Hiển nhiên nếu có một chu trình, chúng ta sẽ phải bỏ một cạnh trong chu trình đó (để xóa chu trình đó đi). Mặt khác, nếu tồn tại chu trình, thì sẽ tồn tại chu trình có nhiều nhất n đỉnh (đò thị chỉ có n đỉnh). Chu trình này có n cạnh, thử bỏ các cạnh trong chu trình này, và chạy dfs để kểm tra xem có tồn tại chu trình nào khác không, chúng ta sẽ giải quyết được bài toán trong độ phức tạp $n * (n + m)$, với O(n+m) là độ phức tạp của dfs.

B

Bài này, trường hợp tệ nhất chúng ta có thể xây dựng đồ thị chứa ở input. Như vậy, chăc chắn sẽ có cách xếp thỏa mãn điều kiện bài toán.

Những bài đồ thị kiểu này (tìm số lượng cạnh nhỏ nhất để xây dựng đồ thị thỏa mãn điều kiện nào đó), thường chúng ta sẽ tìm "chiến lược tối ưu" và xây dựng đồ thị theo chiến lược đấy.

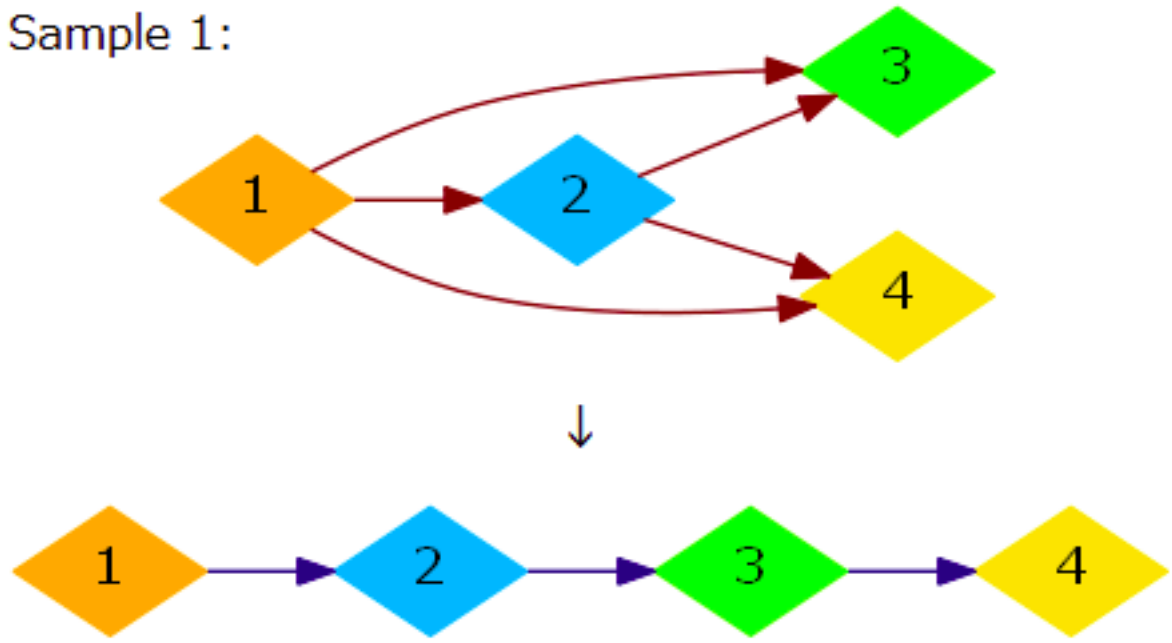
Hiển nhiên đồ thị ở input là đồ thị thỏa mãn. Chúng ta sẽ tìm cách biến đổi đồ thị này sao cho số cạnh cần giữ lại là ít nhất có thể. Gọi đồ thị này là G.

Nếu có 1 tập các đỉnh thuộc đồ thị G tạo thành một thành phần liên thông mạnh, đồng nghĩa với việc hai đỉnh bất kì trong tập đỉnh này phải đi đến được nhau. Đồ thị chúng ta cần xây dựng cũng phải giữ được tính chất này (nếu không thỏa mãn tính chất này -> có hai đỉnh nào đó không đến được nhau -> có một cặp thành phố quan trọng không thể đến được nhau).

Số cạnh ít nhất để tạo thành một thành phần liên thông mạnh n đỉnh là n cạnh (hiển nhiên, tạo thành 1 vòng tròn).

Có nhận xét này, giải quyết bài toán trở nên đơn giản. Chúng ta xây dựng một đồ thị G_1, với các đỉnh là các thành phần liên thông mạnh" trong đồ thị G. Đồ thị này là 1 DAG (DAG là gì có thể xem bài A).

Với một đồ thị DAG, chúng ta có thể sử dụng thứ tự Topo, xây dựng đồ thị theo một đường thẳng để đảm bảo thỏa mãn tất cả các cặp thành phố quan trọng.



Như vậy,với đồ thị G_1 có n đỉnh, chúng ta chỉ cần $n-1$ cạnh để đảm bảo các cặp đỉnh khác thành phần liên thông mạnh mà là cặp đỉnh quan trọng có thể đi đến nhau. Cần xét thêm trường hợp có nhiều cụm liên thông yếu trong đồ thị G_1, như vậy thì sẽ bỏ được một số cạnh nối. Ví dụ nếu đỉnh 1 bên trên không có cạnh nối đến bất cứ đâu, thì chúng ta không cần nối 1 và 2.

Còn lại các thành phần liên thông mạnh, với mỗi thành phần chúng ta cần sử dụng số cạnh đúng bằng số lượng phần tử trong thành phần liên thông mạnh đó. Riêng với trường hợp 1 thành phố là "1 thành phần liên thông mạnh" thì chúng ta không cần sử dụng cạnh nào.

Nhận xét: Bài này làm chủ yếu dựa vào các nhận xét lựa chọn như nào là tốt nhất. Việc xây dựng đồ thị DAG với các đỉnh đại diện cho 1 thành phần liên thông khá phổ biến, đặc biệt trong các bài cần quy hoạch động (Do chúng ta khó có thể quy hoạch động trên đồ thị có chu trình, mà thường dựa vào thứ tự Topo để quy hoạch động như trên dãy số).

C

Đọc đề bài này, chắc chắn chúng ta sẽ nghĩ ngay đến cách tiếp cận bằng quy hoạch động.

Các tiếp cận cơ bản nhất sẽ là $dp[i][j]$ khi đang xét một đoạn nhạc ở vị trí i, một đoạn nhạc ở vị trí j. Quy ước rằng chúng ta xây dựng các đoạn nhạc từ trái sang phải. Để tránh việc hai đoạn con có thể giao nhau. từ $dp[i][j]$, chúng ta chỉ thêm một phần tử ở vị trí lớn hơn j để đảm bảo kiểm soát được việc không dùng một phần tử vào cả hai bản nhạc. Như vậy dp[i][j] sẽ chuyển đến dp[i][x] nếu chúng ta thêm x vào sau bản nhạc kết thúc ở j, hoặc dp[j][x] nếu chúng ta thêm vào bản nhạc còn lại.

Trạng thái khởi tạo là dp[0][i] = 1, với mọi j.

Về cách cập nhận trạng thái quy hoạch động, khi đang xét dp[i][j] và thêm vào sau bản nhạc kết thúc ở j (bản nhạc kết thúc ở i làm tương tự), chúng ta chỉ cần thử thêm 3 vị trí sau:

- Vị trí x gần nhất bên phải của j sao cho $a[x] - a[j] = 1$.
- Vị trí x gần nhất bên phải của j sao cho $a[x] - a[j] = -1$.
- Vị trí x gần nhất bên phải của j sao cho $a[x] - a[j] \vdots 7$.

Các vị trí gần nhất chúng ta có thể dễ dàng khởi tạo trong $O(n^2)$ bằng rất nhiều cách.
Câu hỏi là tại sao chỉ cần xét 3 vị trí gần nhất, mà không phải các vị trí sau nó:

- Với hai trường hợp đầu tiên, do giá trị của a_x là cố đinh, do đó chọn vị trí gần nhất sẽ có lợi hơn.

- Với trường hợp thứ ba, giả sử tồn tại $y > x$ nào đó, $a[y] - a[j] \vdots 7$. Việc chọn a_y cũng sẽ không thể tốt hơn a_x, do nếu chọn a_y -> không thể chọn a_x nữa. Việc này rất lãng phí, do chúng ta có thể chọn a_x, và chọn a_y làm phần tử liền sau của a_x.

Như vậy, chúng ta giải quyết được bài toán trong độ phức tạp $O(n^2)$