Đây là một lời giải **rất khó,** nhưng đồng thời cũng mang rất nhiều giá trị về mặt kiến thức, nên các bạn có thể cân nhắc trước đọc. Nhưng ngoài ra thì các bạn có thể đọc lời giải cho một vài subtask nhỏ nếu như các bạn vẫn chưa giải được sau kì thi.

Subtask 1:

Trong subtask này, chỉ công ty A có các cạnh nối trên đồ thị, do đó bài toán quy về tìm một đường đi từ s đến t sao cho cạnh lớn nhất nằm trên đường đi là nhỏ nhất. Đây là một bài toán khá cơ bản mà các bạn có thể sử dụng cấu trúc dữ liệu Disjoint-set kết hợp với thuật toán Kruskal để giải.

Cụ thể, ta có thể sort lại các cạnh của đồ thị theo trọng số, lần lượt thêm các cạnh vào cho đến khi s và t liên thông. Trọng số của cạnh cuối cùng cũng chính là đáp số bài toán.

Để đọc thêm về Disjoint-set, các bạn có thể xem ở VNOI Wiki:
https://vnoi.info/wiki/algo/data-structures/disjoint-set

Subtask 2:

Có khá nhiều cách giải cho subtask này, nhưng mình sẽ trình bày cách để phục vụ cho subtask 4.

Dựa vào ý tưởng của subtask 1, ta có thể dễ dàng đoán rằng bài này có thể giải bằng tham lam. Bây giờ thay vì, chỉ sử dụng toàn cạnh từ A, các bạn có thể sử dụng thêm một vài cạnh từ B.

Vậy nếu chúng ta cố định cạnh lớn nhất có thể sử dụng của cả A và B, thì mọi cạnh bé hơn các cạnh đã chọn nằm trong 2 tập cạnh A, B đều có thể thêm vào mà không mất thêm chi phí nào.

Để ứng dụng lời giải của subtask 1, ta có thể giải bài toán theo các bước sau:

- Cố định một cạnh lớn nhất trong đồ thị B, thêm tất cả các cạnh bé hơn trong B vào Disjoint-set hiện tại.
- Lần lượt thêm các cạnh trong đồ thị A, dừng lại ngay thời điểm đầu tiên s liên thông với t và cập nhật đáp án (giống như subtask 1, chỉ khác kết quả sẽ được cộng thêm trọng số của cạnh B đã cố định).

Subtask 4:

Từ subtask 2, chúng ta lại thu về được một nhận xét quan trọng sau:

Gọi f(b) = a là trọng số cạnh trong đồ thị A nhỏ nhất để nếu thêm tất cả các cạnh trong đồ thi B với trọng số <= b và các cạnh trong đồ thị A với trọng số <= a thì s liên thông với t.

Nhận xét: f(b + 1) <= f(b).

Đây là một nhận xét khá rõ ràng, do thêm càng nhiều cạnh trong B vào thì cần càng ít cạnh trong A để s và t liên thông, tương đương với việc trọng số cạnh A lớn nhất cần thiết cũng sẽ nhỏ đi.

Thuật toán two-pointer:

Đến đây, ta có thể dễ dàng nhìn ra một ý tưởng sau:

- Đầu tiên thêm tất cả các cạnh trong đồ thị A vào đồ thị.
- Lần lượt tính các f(b) với b từ nhỏ đến lớn, mỗi lần thì thử xóa các cạnh A lớn nhất đến khi nào s và t vẫn còn liên thông.

Cản trở: Theo như mình biết thì đến hiện tại, chưa có một cấu trúc dữ liệu nào đủ mạnh để thêm và xóa cạnh bất kì cùng lúc phục vụ cho các truy vấn ở bên trên trừ Link-Cut Tree (thứ mà bạn chỉ cài trong 3 tiếng kì thi quốc gia nếu như bạn bị mất não).

Một cách tiếp cận khác: Nếu nhìn theo một mặt khác, cấu trúc dữ liệu Disjoint-set ở trên có thể giúp bạn trả lời hai truy vấn bao gồm thêm cạnh và hỏi tính liên thông của 2 đỉnh bất kì. Thứ duy nhất nó thiếu là thao tác xóa cạnh.

"Vậy nếu không xóa thì sao?"

Điều này là hoàn toàn có thể, do giới hạn của bài toán là N <= 50 000, nên không ai bắt bạn phải làm một thuật cực kì tối ưu mới có thể ăn hết điểm. Một lời giải dựa trên các nhận xét này đã được giáo sư PVH chia sẻ và các bạn có thể tham khảo ở đây: https://www.facebook.com/groups/VNOIForum/permalink/2843454175675398/?comment_id=2843697328984416

Lời giải của mình:

Như đã nói ở trên thì lời giải của mình cho bài toán này là chia để trị, và một nhận xét quan trọng cần thiết để thực hiện mình đã nói ở trên: f(b + 1) <= f(b).

Nhận xét thứ 2: thứ tự để tính f(b) không quan trọng.

Do đó mình có một tư tưởng chia để trị với hàm solve như hình 2

Lí do ta có thuật toán này, là bởi vì với một đoạn [L, R] bất kì và mid = (L + R) / 2 thì f(R) <= f(mid) <= f(L).

Vấn đề bây giờ là làm thế nào để tính f(mid) một cách nhanh chóng.

Nếu ta coi như trong Disjoint-set hiện tại đã có tất cả các cạnh B với trọng số < L và các cạnh A với trọng số < from. Thì việc tính f(mid) có thể thực hiện như sau:

- Thêm tất cả các cạnh B có trọng số từ L đến mid.
- Lần lượt thêm các cạnh A có trọng số từ from đến to và dừng lại khi s và t liên thông, trọng số của cạnh cuối cùng thêm vào chính là f(mid).

Tiếp theo, để hàm chia để trị xuống [mid + 1, R] và [L, mid - 1] thỏa mãn điều kiện đã giả sử ở bên trên, thì bây giờ ta phải xóa các cạnh đã thêm hiện tại khỏi Disjoint-set và thêm một số cạnh vào sao cho phù hợp.

Mấu chốt: Các cạnh ta cần xóa đi hiện tại là những cạnh cuối cùng vừa được thêm vào Disjoint-set.

Hay nói cách khác, ta có thể lưu lại những thay đổi gần nhất nhất trong cấu trúc Disjoint-set và "phiên bản cũ" của chính nó để khôi phục lại Disjoint-set giống như trước khi tính hàm f(mid).

Persistent Disjoint-set (with rollbacks):

Nếu như với disjoint-set thông thường, các bạn thường có 2 hướng chính sau:

- Union by rank
- Nén đường

Và có nhiều bạn còn làm cả 2.

Để ý rằng, nếu sử dụng phương pháp Union by rank, thì mỗi lần hỏi về tính liên thông của hai đỉnh bất kì, ta chỉ mất O(log N), do đó nén đường trong trường hợp này là không cần thiết.

Thứ hai, trong hàm thêm cạnh (join) thì chỉ có thông số của duy nhất 2 đỉnh bị thay đổi. Do đó, chi phí để ta "rollback" disjoint-set về K cạnh trước đó có độ phức tạp là O(K).

Quay lại với bài toán, giờ đây ta có thể thực hiện lời giải theo các bước sau:

- Tính f(mid)
- Rollback về như cũ, thêm các cạnh A từ from đến f(mid) và gọi solve(L, mid - 1, f(mid), to).
- Roll back về như cũ, thêm các cạnh B từ L đến mid và gọi solve(mid + 1, R, from, f(mid)).

- Roll back về như cũ tiếp.

Đánh giá độ phức tạp: Do mỗi lần ta chia đôi L, R nên sẽ có tối đa O(log |B|) tầng, và mỗi tầng, tổng số cạnh ta duyệt qua là O(|A| + |B|). Với mỗi cạnh duyệt qua, ta có thêm truy vấn về tính liên thông của s và t, nhờ Union by rank nên độ phức tạp cho mỗi truy vấn là O(log N).

Vậy tổng kết lại độ phức tạp của lời giải này là O(M log M log N).

Lời kết: Do bài tập này từ đầu đã ở mức advanced, nên để giải thích cặn kẽ từng chỗ thì sẽ rất dài, nên mong các bạn thông cảm. Mình cũng xin khẳng định lại là lời giải ở trên không hề đơn giản, nên dù bạn có đọc không hiểu ngay thì cũng rất bình thường. Mình rất hi vọng các bạn có thể nghiền ngẫm đủ lâu và học được cái gì đó từ bài viết này. :)

Kiến thức quan trọng (không phải cho VOI) về Persistent Dsu các bạn có thể search trên google và đọc các tài liệu tiếng anh.

Tham khảo:
https://vnoi.info/wiki/algo/dp/Mot-so-ky-thuat-toi-uu-hoa-thuat-toan-Quy-Hoach-Dong