Cặp đường thẳng – Pair Of Lines

Vì ta được dùng tối đa hai đường thẳng, nếu chọn được một đường rồi thì có thể kiểm tra được bằng cách kiểm tra xem các điểm còn lại có nằm trên một đường thẳng hay không.

Vậy thì chọn đường thẳng đầu tiên như thế nào?

Cách thứ nhất là sử dụng xác suất:

- Có hai đường thẳng nên có ít nhất một đường có nhiều hơn $\frac{1}{2}$ số điểm.
- Chọn ngẫu nhiên ra 2 điểm thì xác suất để hai điểm này cùng nằm trên đường chứa nhiều điểm hơn là khoảng $\frac{1}{4}$.
- Chọn được một đường thì ta có thể kiểm tra trong $O(n)$. Có thể làm khoảng 30, xác suất để tìm được đáp án nếu có khi đó ít nhất là $1 - \frac{3}{4}^{30} \approx 0.99982141791$.
- https://codeforces.com/contest/961/submission/43490377
- Cách này nhìn chung code đơn giản, chỉ cần một hàm kiểm tra.

Cách thứ hai là nhận thấy rằng nếu có 3 điểm không thẳng hàng, một trong hai đường trong kết quả sẽ chứa 2 trong số 3 điểm này. Vậy nếu tìm được 3 điểm không thẳng hàng thì chỉ cần kiểm tra 3 đoạn thay vì ≈ 30 đoạn như cách thứ nhất. Để tìm 3 điểm không thẳng hàng thì:

- Xuất phát với 1 đường thẳng. Loại đi tất cả các điểm trên đường thẳng này.
- Nếu mà tất cả các điểm đều nằm trên đường thẳng thì rõ ràng chỉ cần dùng 1 đường thẳng.
- Nếu không thì có thể chọn điểm bất kì nằm ngoài đường này, ta sẽ tìm được 3 điểm không thẳng hàng.
- Nhìn chung, thường sẽ chạy nhanh hơn nhưng code dài hơn so với cách thứ nhất.

Tô màu đồ thị - Graph Coloring

Nhận thấy là ta chỉ tô màu bằng mỗi đỉnh một lần vì khi tô bằng một đỉnh hai lần thì xem như không tô màu.

Bây giờ có hai cách tô màu, là tô tất cả cạnh thành màu đỏ hoặc tô tất cả cạnh thành màu xanh.

Ta giải mình trường hợp tô thành màu đỏ, tô thành màu xanh làm tương tự là được.

Khi tô thành toàn màu đỏ thì:

- Nếu cạnh $u - v$ đang đỏ, ta không được tô cả u và v hoặc là ta phải tô cả u và v.
- Nếu cạnh $u - v$ đang xanh, ta phải tô đúng một trong hai đỉnh u và v.

Vậy thì khi ta biết được u có được tô hay không, ta có thể biết luôn được tất cả các đỉnh v kề u có được tô hay không. Nếu ta cứ lặp lại quá trình thì sẽ biết được cả thành phần liên thông chứ u.

Thì các bài kiểu này gọi là tô màu đồ thị $0\ 1$.

Để giải bài này, ta có thể mô phỏng các đỉnh như sau:

- Một đỉnh u trên đồ thị ban đầu có thể được tô hoặc không. Ta sinh ra hai đỉnh u_0 và u_1 tương ứng với việc không tô hoặc là tô đỉnh u.
- Dựa vào điều kiện cạnh $u - v$, ta có thể biết được là nếu tô u hay không thì v sẽ như thế nào. Ta có thể nối các cặp u_i và v_j tương ứng bằng cạnh hai phía.
- Một thành phần liên thông các đỉnh u_i thì hoặc là đồng thời được thực hiện, hoặc là không. Điều này vì cạnh hai phía ở đây sẽ có quan hệ như phép tương đương, làm cái này cũng phải làm cái kia, không làm cái này cũng không làm cái kia. Vì mỗi cạnh có tính tương đương nên cả thành phần liên thông tương đương với nhau.
- Nếu tồn tại đỉnh u_0 và u_1 trong cùng thành phần liên thông thì dĩ nhiên là không có cách tô màu vì yêu cầu ở đây là tô u thì cũng không được tô u và ngược lại.
- Nếu tồn tại cách tô màu, có thể chứng minh được là nếu một thành phần liên thông chứ $u_i, v_j, w_k, \ldots$ thì có một thành phần liên thông

khác chứa u_{1-i}, v_{1-j}, w_{1-k}, … . Điều này thấy được vì nếu có cạnh $u_i - v_j$ thì cũng có cạnh $u_{1-i} - v_{1-j}$.

- Nếu có cách tô, mỗi thành phần liên thông trên đồ thị ban đầu sẽ ứng với hai thành phần liên thông đối nhau trên đồ thị các đỉnh u_i. Điều này có được vì chỉ khi có cạnh $u - v$ thì mới có thể có cạnh $u_i - v_j$.
- Với mỗi thành phần liên thông trên đồ thị ban đầu, ta có thể chọn một trong hai thành phần liên thông ứng với nó trên đồ thị các đỉnh u_i cho nó là đúng, cái còn lại là sai thì sẽ đạt được một cách tô màu thỏa mãn.
- Vì bài toán yêu cầu tô ít điểm nhất, ta sẽ chọn cách nào dùng ít điểm nhất trong hai cách, sau đó tổng hợp lại là được.

Để cài đặt, chú ý các chi tiết sau:

- Nối điểm kiểu $u_i \to v_j$ thực ra khá lằng nhằng, cộng thêm việc trong hàm DFS, BFS, DSU bị mất thêm một tham số. Vì thế, ta có thể định nghĩa $u_0 = u$ và $u_1 = u + n$ để code cho đơn giản.
- Nếu dùng DFS hay BFS thì ta sẽ phải lưu đồ thị dùng danh sách kề hay một cách nào đó. Có thể dùng DSU thì sẽ chỉ phải lưu mình gốc, thêm nữa code ngắn gọn hơn.

https://codeforces.com/contest/663/submission/32672418

Bài này là một bài về việc xử lí logic. Nhìn chung, kiểu này sẽ giải được tất cả các bài có dạng:

- Cho các thao tác. Với thao tác thứ i, có $a_i = 1$ khi và chỉ khi làm thao tác này. Có các quan hệ hai chiều là $a_i = x \Leftrightarrow a_j = y$, tìm một cách làm thỏa mãn.

Ngoài ra, xử lí logic còn có thể có nhiều yêu cầu hơn, các bạn muốn biết hãy tìm hiểu về 2SAT.

Lại thêm một bài so khớp xâu - Yet Another String Matching Problem

Đầu tiên cần thấy được trong bài này, các xâu chỉ sử dụng 6 kí tự từ a đến f. Thông thường thì các bài xâu sẽ dùng cả 26 kí tự trong bảng chữ cái tiếng Anh, nên cái này chắc chắn sẽ cần để giải quyết bài toán.

Bây giờ ta xét hai xâu s và t có $|s| = |t|$. Cách tối ưu để làm cho hai xâu này bằng nhau là như thế nào?

- Với mỗi s_i và t_i, đến cuối cùng thì hai kí tự này phải bằng nhau. Vì thế, ta có thể biểu diễn quan hệ này bằng một cạnh nối giữ kí tự s_i và kí tự t_i.
- Làm xong như vậy thì ta được một cái đồ thị có một số thành phần liên thông, ta cần phải biến tất cả các kí tự trong cùng một thành phần liên thông thành các kí tự giống nhau.
- Mỗi thành phần liên thông có x kí tự thì cách tốt nhất dĩ nhiên là biến $x - 1$ kí tự thành kí tự còn lại.

Vậy thì ta biết được là kết quả sẽ phụ thuộc vào cái đồ thị mà ta sinh ra được. Cụ thể hơn là sẽ phụ thuộc vào việc mỗi đỉnh nằm ở thành phần liên thông nào. Biết được điều đó thì ta cũng có chiến thuật để biến đổi các xâu luôn.

Trở lại điều kiện có 6 kí tự khác nhau. Dựa vào điều kiện này thì số cách ghép thành phần liên thông khá là bé, cụ thể là 203 cách. Chú ý, để được số cách bé thì phải làm như sau:

- Duyệt các kí tự $\mathrm{a} \to \mathrm{f}$.
- Khi duyệt một kí tự, có hai trường hợp như sau:
    - Kí tự này nằm trong một thành phần liên thông khác với các kí tự đã duyệt.
    - Kí tự này nằm cùng thành phần liên thông với các kí tự đã duyệt.

Vì số lượng khá bé, ta có thể thực hiện duyệt hết mọi cách. Với mỗi cách:

- Có được một chiến thuật là sẽ đổi từ kí tự nào sang kí tự nào, chi phí của cách này là bao nhiêu.

- Ta có thể sinh ra xâu S' và T' là xâu S và T sau khi được đổi bằng chiến thuật trên.
- Với mỗi xâu con của S', nếu xâu con này mà bằng T' thì chứng tỏ là chiến thuật hiện tại sẽ làm cho xâu con tương ứng trên S thành xâu T. Vì ta cần tìm chi phí nhỏ nhất, chi phí của mỗi xâu con sẽ là min chi phí trong các chiến thuật khác nhau.
- Có thể dùng hash, KMP, Z function, ... để tìm tất cả các lần T' xuất hiện trong S' trong $O(|S| + |T|)$. Làm vậy thì kiểm tra mất $O(|S| + |T|)$

Độ phức tạp tổng cả là $O\big(203 \times (|S| + |T|)\big)$. Tính ra chỉ có 5×10^7 trong trường hợp xấu nhất, dư sức giải được bài này.

Code dùng KMP:

https://codeforces.com/contest/954/submission/36520970

Bài này các tác giải cho 6 chữ cái khác nhau thực ra là để làm cách khác, tuy nhiên cách này lại cũng giải được. Điểm đặc biệt nhất là số lượng trạng thái. Không chỉ riêng trong bài này, ở trong nhiều bài thì khi ta tính ra số lượng trạng thái sẽ được giá trị khá bé/đủ bé để làm một thuật nào đó được. Ở bài này thì ta dùng để duyệt, tuy nhiên trong các bài khác thì có thể là có ít trạng thài nên quy hoạch động dựa trên các trạng thái đó, ...