MAC0345 - Desafios 2
Editorial Lista 3

Nathan Luiz e Enrique Junchaya

May 24, 2023

A. Forbidden Cities.
Tópicos: Block Cut Tree.

Esse problema é bem chato de codar. A ideia não é tão complicada. Faça a Block cut tree e também alguma estrutura que acha LCA (HLD ou binary lifting). Vamos separar em alguns casos:

- Caso 1: Se $C = A$ ou $C = B$:

  Resposta é não.

- Caso 2: Se C não for ponto de articulação.

  Resposta é sim. Pois o grafo continua conexo depois da remoção de C.

- Caso 3:

  Façamos a seguinte substituição

  ```
  A = if A é ponto de articulação, seu id na block cut tree:
                  else id da componente dele.
  ```

  Analogamente para o B e C. Se o vértice C estiver no meio do caminho de A para B, então não existe solução. Para checar isso, pegue o LCA_{AB}. Se o LCA_{ABC} estiver acima do LCA_{AB}, então C não está no caminho. Caso contrário, então $LCA_{ABC} = LCA_{AB}$. Tem que ver somente se $LCA_{AC} = C$ ou $LCA_{BC} = C$.

B. Strongly Connected Edges.
Tópicos: Dfs tree.

Se o grafo tem pontes, claramente não existe resposta. Se ele não tem pontes, pela construção que fazemos do tarjan, já que cada vértice possui `low[u] < prof[u]`, então se direcionamos as arestas arborescentes para baixo e as outras para cima, podemos sempre subir na árvore até chegar até a raiz e a partir daí ir para qualquer outro vértice.

Portanto, no final basta verificar se tem ponte e, se não tiver, direcionar as arestas na ordem dada acima.

C. Necessary Roads.
Tópicos: Pontes.

Aplicação direta de tarjan. Neste template, `vector<bool> bridges` é a resposta.

D. Necessary Cities.
Tópicos: Pontos de articulação.

Aplicação direta de block cut. Neste template, `gart[i].size ()` diz se um vértice é ponto de articulação ou não.

E. British Menu.
Tópicos : Componete Fortemente Conexa, PD.

O editorial foi copiado daqui.

Dado um digrafo $G = (V, E)$ tal que, para todo passeio fechado que não repete os externos como vértices internos, há no máximo 5 vértices distintos. Os vértices externos são os extremos do passeio, e os internos são os demais vértices. O objetivo é calcular um caminho de cardinalidade máxima nesse grafo. Esse problema é conhecido como o problema do Caminho Hamiltoniano que é NP-Difícil no caso geral. Com isso, devemos utilizar a estrutura do grafo para derivar uma solução eficiente para o problema. Para resolver o problema utilizaremos componentes fortemente conexas, uma aula sobre o tópico ser encontrada no canal no You Tube.

Lema 1: Seja G um digrafo em que todo passeio fechado, que não repete os extremos, tem no máximo k vértices distintos. Então, toda componente fortemente conexa tem no máximo k vértices.

Proof. Vamos provar por contradição. Suponha que há uma componente fortemente conexa U com $|U| > k$. Seja $v \in U$ um vértice tal que o maior conjunto de vértices distintos em um passeio fechado que não repete os extremos seja máximo e seja P_v esse passeio. Seja $u \in U - V(P_v)$ (sabemos que há pois $|U| > k \geq |V(P_v)|$). Como estão na mesma componente fortemente conexa, sabemos que há caminhos Q_{uv}, Q_{vu} que ligam u até v e v até u. Logo, $P_u = Q_{uv} \cdot P_v \cdot Q_{vu}$ é um passeio que fechado que só repete u nos extremos tal que $|V(P_u)| > |V(P_v)|$, uma contradição. □

De posse do Lema 1, uma ideia natural é obter o grafo de condensação das componentes fortemente conexas, isto é, o grafo em que cada componente fortemente conexa é um vértice. Seja D esse grafo, então, D é um DAG (grafo direcionado acíclico). Um caminho máximo em um DAG pode ser calculado por meio de uma programação dinâmica. Seja $N : V \to 2^V$ tal que $N_v \coloneqq \{u : (v, u) \in A(D)\}$, isto é, os vértices para os quais v "aponta". Então, temos a seguinte recorrência para a função f que calcula a cardinalidade do caminho (na verdade, o número de vértices) máximo começando em v.

$$f_v = 1 + \max_{u \in N_v} f_u.$$

Note que isso só é válido pois D não contém ciclos. Sabendo disso, basta encontrar uma forma de calcular o caminho máximo dentro de cada componente fortemente conexa. Pois podemos particionar um caminho máximo em partes entre componentes fortemente conexas e dentro das mesmas. Pelo Lema **??**, cada componente terá no máximo 5 vértices. Logo, podemos calcular o caminho máximo entre os vértices de cada componente fortemente conexa por meio de força bruta ou *backtracking*. Juntando essas duas informações conseguimos resolver o problema em $O(n + m)$, a complexidade de encontrar o grafo de condensação com Tarjan e Kosaraju.

Alguns detalhes de implementação devem ser ressaltados. Primeiramente, os caminhos máximos para cada par de vértices dentro de cada componente fortemente conexa podem ser calculadas por meio de uma busca em profundidade em que o se "desmarca" o vetor de visitados no final da execução da função no vértice, o que é bem simples de implementar. Além disso, quando calculando o caminho máximo no grafo de condensação, devemos salvar por qual vértice da componente forte esse caminho "entra", para que os caminhos sejam concatenados corretamente. Ademais, cuidado com a ordem das operações de concatenação dos valores, para que não seja calculado um passeio máximo. Por fim, adicionar um vértice artificial que aponta para todos os vértices (note que não quebra a propriedade) pode facilitar o processo.

F. Two Houses..
Tópicos: Interativo, SCC.

Lema: Sejam a, b duas cidades de maneira que $k_a \leq k_b$. a, b estão na mesma SCC se, e somente se, existe um caminho de b para a.

Proof. Para a ida, se elas estão na mesma SCC então existe um caminho de a para b e de b para a.
A volta é um pouquinho mais complicado.

- Caso 1: (Se tem uma aresta direta de a para b)

  Então, como existe caminho de b para a, eles estão na mesma SCC.

- Caso 2: (Se não tem uma aresta direta de a para b)

  Seja X_u o conjunto de cidades de maneira que existe uma aresta direcionada de todo elemento de X_u para u (os vértices que entram em u). Analogamente, defina Y_u os vértices que recebem aresta de u. Vamos provar que existe pelo menos um nó pertencente a Y_a e X_b. Temos:

  $|Y_a| + |X_b| = (n - 1 - k[a]) + k[b] = n - 1 + (k[b] - k[a]) > n - 2$

  Pelo princípio das casas dos pombos, já que b não pertence a Y_a e a não pertence a X_b, existe um elemento comum de Y_a e X_b. Portanto, seja x um elemento em comum de Y_a e X_b. Então, existe um caminho de a para x e de x para b. Como já sabemos que existe um caminho de b para a, então a e b estão na mesma SCC.

□

Portanto, podemos iterar por todos os pares a, b de maneira que $|k_a - k_b|$ é sempre o maior possível, e utilizar o lema acima para chegar se estão na mesma componente fortemente conexa.