Ancestral comum mais próximo entre dois vértices

Pedro Vítor Bortolli Santos

Instituto de Matemática e Estatística da Universidade de São Paulo

Objetivos

- Fornecer um material didático na Língua Portuguesa sobre o problema de encontrar o ancestral comum mais próximo entre dois vértices de uma árvore.
- Desenvolver habilidades em escrever um texto com um maior grau de formalidade, ao mesmo tempo que é desejável fornecer diversos exemplos práticos sobre o assunto.
- Contribuir com a evolução e crescimento da comunidade brasileira da Maratona de Programação.

Descrição do problema

- Dada uma árvore e dois vértices distintos **a** e **b**, encontrar o ancestral comum mais próximo entre eles, também conhecido como **LCA**.
- Define-se como o **LCA** entre dois vértices **a** e **b** aquele nó que está mais abaixo (ou seja, mais profundo na árvore) e que contém ambos **a** e **b** em sua subárvore.

Algumas soluções

- Fazer com que **a** e **b** estejam na mesma altura e, então, caminhar com ambos simultaneamente em direção à raiz da árvore até que se encontrem. O primeiro vértice de encontro é o **LCA**.
- Mesma solução acima, porém encurtando a quantidade de pulos usando o fato de que qualquer número pode ser representado por uma soma de sucessivas potências de dois distintas. Dessa forma, se o **LCA** está situado 11 níveis acima dos vértices **a** e **b**, podemos dar um pulo de tamanho 8, um de tamanho 2 e um de tamanho 1. Envolve um pré-processamento antes de realizar consultas.
- Planificar a árvore de forma que os vértices do caminho de **a** para **b** sejam representados em um intervalo contínuo. Assim, para obter o **LCA** basta consultar qual é o elemento cuja profundidade é mínima neste intervalo. Esta técnica, chamada de Passeio de Euler, possui como pré-requisito o conhecimento da estrutura de dados "Árvore de Segmentos".

Algoritmo simples

- Assumimos que já temos pré-calculados os vetores **profundidade** e **pai**. A computação destas listas se faz de forma básica: uma simples busca em profundidade (**DFS**) é capaz de construí-los.

Algoritmo 2 Determina o LCA entre dois vértices

```
1: função LCA_SIMPLES(a, b)
2:     se profundidade[a] < profundidade[b] então
3:         troca(a, b)
4:     enquanto profundidade[a] > profundidade[b]
5:         a ← pai[a]
6:     enquanto a != b
7:         a ← pai[a]
8:         b ← pai[b]
9:     devolve a
```

Figura 1:Pseudocódigo para encontrar o LCA entre dois vértices de uma árvore

- Complexidade de tempo $O(n)$.

Exemplo de um problema: Colônia de Formigas

- Uma colônia de formigas se organiza em túneis que conectam diferentes formigueiros cada um. É possível chegar em qualquer formigueiro partindo de qualquer um deles.
- Existem N formigueiros e $N - 1$ túneis que os conectam.
- Cada túnel i tem comprimento L_i.
- Dados dois formigueiros S e T, determine o menor caminho para sair do formigueiro S e chegar em T (menor soma de comprimentos dos túneis escolhidos para serem percorridos).

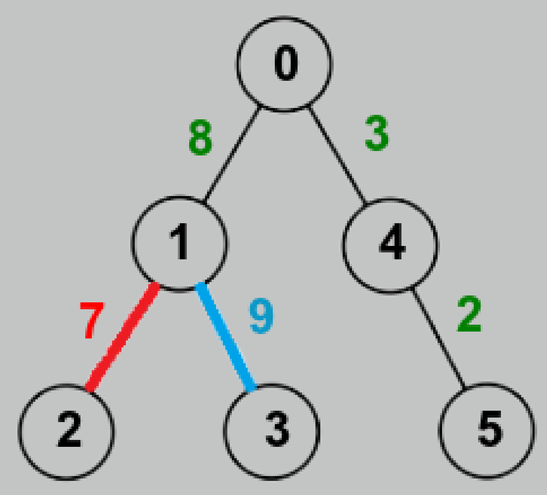

Figura 2:Exemplo de uma entrada para o problema

- No exemplo acima, o caminho do vértice **2** ao **3** (destacado em cores) possui comprimento total $\mathbf{7 + 9 = 16}$.

Solução eficiente

Primeiramente, como temos uma árvore sabemos que existe apenas um único caminho entre quaisquer dois vértices. Além disso, o **LCA** entre dois vértices sempre estará contido no caminho.
Podemos pré-calcular a soma acumulada dos comprimentos das arestas da raiz para cada vértice usando uma DFS, de forma que:

- Soma até 2: **15**
- Soma até 3: **17**
- Soma até 1: **8**

Note que ao caminhar de cada vértice em direção até a raiz passaremos pelo **LCA**. Assim, para obter o comprimento do caminho do **LCA** até cada vértice **v** basta obter a diferença da soma até **v** menos a soma até o **LCA**. Para visualizarmos isso no exemplo, temos que:

- $soma[3] - soma[1] = 17 - 8 = 9$
- $soma[2] - soma[1] = 15 - 8 = 7$

Por fim, tendo o comprimento de cada caminho, basta somá-los e este será o comprimento total do caminho inteiro de **2** até **3** - custo: $\mathbf{9 + 7 = 16}$.

- Complexidade de tempo para obter o comprimento total: $O(1)$.
- Complexidade para pré-calcular o vetor de soma acumulada: $O(n + m)$ (custo de tempo de uma DFS).
- Complexidade de tempo para computar o LCA: $O(log\ n)$ (utilizando o método mais eficiente estudado no trabalho).

Veja mais

Para explicações mais aprofundadas dos algoritmos, assim como as provas de suas corretudes e complexidades, e resoluções mais detalhadas de problemas, acesse a monografia completa:

https://linux.ime.usp.br/~bortolli/mac0499/