

Relatório - 2º Projeto ASA 2021/2022

Grupo: al096

Aluno(s): João Nuno Cardoso (99251) e José João Ferreira (99259)

Descrição das Soluções dos Requisitos

No âmbito da resolução do exercício de determinação do ancestral comum mais próximo entre 2 vértices proposto, decidimos desenvolver uma solução *bottom-up* de pesquisa DFS que começasse pelos vértices v_1 e v_2 , explorando subsequentemente os seus ancestrais e pintando-os, de acordo e de forma recursiva.

Para o 1º requisito, de determinação da validade da árvore, lemos, pelo *input*, as relações de parentesco. Guardamos, por cada vértice, uma referência para cada um dos progenitores, numa estrutura de dados chamada *vertex*. Ao tentar atribuir a um vértice um 3º progenitor, é acusada a invalidez da árvore. Dentro de *vertex*, também é guardado um booleano *clear*, indicador da existência ou não de um ciclo atingível a partir do vértice (note-se que esta propriedade é hereditária). Deste modo, ao realizar uma DFS recursiva por cada vértice, com um set que vai mantendo registo dos vértices percorridos, atribuindo *true* a *clear* para cada vértice através do qual a DFS chega somente a antepassados órfãos, e retornando de imediato *false* assim que um caminho vá ter a um vértice já descoberto, conseguimos, eficazmente, detetar a existência de ciclos. Equivalentemente a uma DFS típica, estamos na presença de um limite assintótico de grandeza $O(V+E)$. Contudo, se a DFS tiver início num vértice pertencente a um *loop*, o número de iterações é constante ($\Omega(1)$).

Por último, no que concerne o 2º requisito, guardamos também para cada vértice uma variável *color*, que pode ser *white*, *red*, *blue*, *purple* ou *black*. Para v_1 , realizamos uma DFS em que marcamos todos os seus antepassados, originalmente a branco, com a cor vermelha. De seguida, uma outra DFS, com início em v_2 , marca todos os seus antepassados brancos a azul. Ao encontrar um antepassado vermelho, marca-o a roxo, colorindo, seguidamente, todos os seus antepassados de preto, independentemente da sua cor. No final, sabemos que os vértices roxos restantes constituem a solução do problema, pois constam na interseção dos antepassados de v_1 e v_2 . Como se trata de uma DFS, a complexidade típica pertence a $O(V+E)$, conquanto na maioria dos casos apenas uma fração do grafo é explorada, ocorrendo ainda um número constante de iterações se o ancestral comum mais próximo for órfão e progenitor de v_1 e v_2 .

Análise Teórica

Numa árvore válida existem, no máximo, 2 arestas por vértice, logo, no pior caso, $|E| = 2|V|$.

- Leitura de arestas para *array* de vértices: $O(|E|) = O(2|V|)$
- Algoritmo que verifica se há *loops*: $O(|V|+|E|) = O(3|V|) = O(|V|)$, $\Omega(1)$
Este algoritmo percorre em média cada vértice e aresta uma única vez.
Se encontrar imediatamente o *loop*, corre apenas três iterações.
- DFS para determinar ancestral comum mais próximo: $O(|V|+|E|) = O(|V|)$, $\Omega(1)$
Em média, este algoritmo percorre uma fração do número de arestas e vértices do grafo. No melhor caso, encontra de imediato um antepassado órfão, no pior terá que percorrer todos os vértices e arestas, tal como uma DFS normal.

Complexidade de tempo: $O(|V| + |E|) = O(|V|)$, complexidade de espaço: $O(2|V|) = O(|V|)$.

Relatório - 2º Projeto ASA 2021/2022

Grupo: al096

Aluno(s): João Nuno Cardoso (99251) e José João Ferreira (99259)

Pseudocódigo de partes das lógicas por detrás das soluções apresentadas:

Verificação de loops:

```
for i=1 to #vertices do
  let path be a new set
  if (has_cycle(i, path, vertices)) then
    print("0")
    return
has_cycle(int v, set path, vertex vertices[])
  if vertices[v].clear == true then
    return false
  if path.contains(v) then
    return true
  else do
    path.insert(v)
    bool res = res
    for p in v.parents do
      res = res || has_cycle(p, path, vertices)
    vertices[v].clear = !res
  return res
end
```

Determinação do LCA:

```
DFS(int v, color c, vertex vertices[])
  if vertices[v].color == red & c == blue then
    c = purple
    vertices[v].color = c
  for p in v.parents do
    if p.color == {white || red} & c == blue do
      DFS(p, c, vertices)
    else if p.color != black &
      c == {purple || black} do
      DFS(p, black, vertices)
  end
```

Avaliação Experimental dos Resultados

Foram registados 10 resultados, um para cada número de vértices dados por *input* e de forma incremental, desde 1×10^4 até 1×10^5 , de 10000 em 10000, e calculados os tempos para cada instância. Para cada n^o de vértices, o n^o de arestas geradas foi, aproximadamente, o seu dobro e foram realizados 100 testes. Os *inputs* foram gerados com recurso ao `randGeneoTree` e a média de resultados foi calculada com recurso à ferramenta `hyperfine`. O tempo encontra-se em milissegundos (YYs) e o n^o de vértices na potência 10^4 (XXs).

Recordando a complexidade da solução apresentada, a curva do gráfico gerado está de acordo com aquilo que seria de esperar de um algoritmo com limite assintótico apertado de grandeza $O(|V| + |E|)$.

