# Relatório 2º Projeto ASA 2021/2022

Diogo Melita, 99202   Diogo Gaspar, 99207
Grupo: al038

## 1 Descrição do Problema e da Solução

Como segundo projeto para ASA, foi proposta a realização de um algoritmo que, dado um grafo dirigido $G = (V, E)$:

- Verificasse se $G$ corresponde a uma **árvore genealógica válida**: um grafo acíclico, onde cada nó tem no máximo dois antecessores.
- Dados dois nós, $u, v \in G$, devolvesse o **conjunto de ancestrais comuns mais próximos entre eles**, $LCA$.

Será importante, antes de qualquer outra coisa, definir ancestral comum mais próximo. Um nó diz-se **ancestral comum mais próximo** de dois nós $u$ e $v$ caso não exista nenhum nó que dele descenda e que seja ancestral comum de $u$ e $v$. Tendo isto em conta, a solução encontrada tem por base a *Depth-First Search*, DFS, para ambos os pontos acima referidos.

Para verificar a **validade** do grafo, é realizada uma DFS tal como abordado nas aulas teóricas pelo grafo transposto de $G$, marcando cada nó com **cores**. Se ao visitar as adjacências de um nó for encontrado um nó que está a ser atualmente visitado - estado *grey* - foi encontrada uma *back edge*, e, por consequência, **um ciclo**. A verificação da quantidade de antecessores é bastante mais trivial - se o vértice-destino do arco a ser adicionado já tiver 2 antecessores, o algoritmo pode parar, visto que a árvore não será válida.

A solução para o problema dos **LCA** é mais simples do que pode parecer à primeira vista, principalmente tendo em conta a definição referida mais acima. É importante realçar que todos os nós são, aqui, inicializados a *white*. Num primeiro momento, realiza-se uma DFS ao grafo transposto de $G$, partindo de $u$ (um dos vértices-argumento). Todos os vértices encontrados pela DFS são marcados com a cor *black*, recuperando aqui alguma lógica da DFS acima mencionada. É, de seguida, realizada uma segunda DFS ao mesmo grafo transposto, partindo agora de $v$ (o outro vértice-argumento). A procura marca um nó com *gray* caso atualmente não esteja no estado *black* e tenha sido encontrado na procura que partiu de $u$. Todos os nós seus antecessores são marcados a *black*, garantindo que, no final do algoritmo, os nós marcados a *gray* são necessariamente os únicos ancestrais comuns mais próximos de $u$ e $v$. Um vértice só pode ser marcado com *black* uma vez, evitando "subidas"desnecessárias pelo grafo.

## 2 Análise Teórica

Considere-se um grafo $G = (V, E)$. O custo espacial da leitura e tratamento de dados é, aqui, $O(|V|)$ - é mantido um vector de vectores com, no máximo, $|V|$ entradas, cada uma delas com um vector de, no máximo, dois elementos. A complexidade temporal da mesma secção é de $O(|E|)$, já que, na pior das hipóteses, são lidas todas as $|E|$ arestas do grafo.

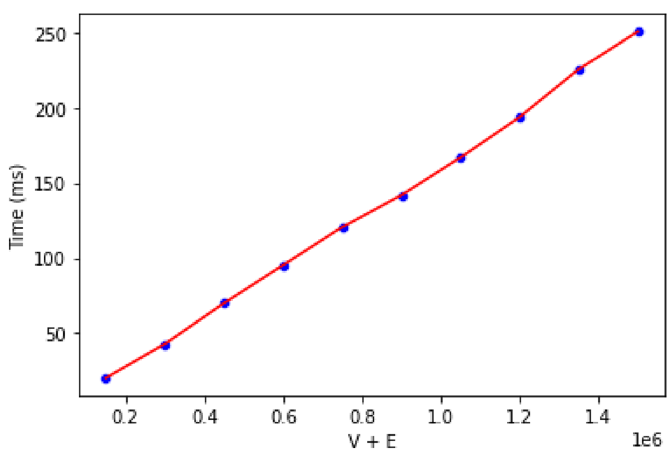
A verificação de ciclos corresponde a uma DFS pelo grafo transposto, com complexidade temporal $O(|V| + |E|)$ e espacial $O(|V|)$ (é mantido um vector com $|V|$ entradas).

Por fim, a determinação dos LCA consiste em duas outras DFS, uma primeira mais simples e uma segunda que passa, na pior das hipóteses, duas vezes por cada aresta, ambas com complexidade temporal $O(|V| + |E|)$. São guardados dois outros vectores com $|V|$ entradas, para guardar o estado de cada nó em cada procura, pelo que esta secção tem complexidade espacial $O(|V|)$.

Olhando para o algoritmo como um todo, é possível afirmar que este tem, então, complexidade temporal $O(|V| + |E|)$ e complexidade espacial $O(|V|)$.

## 3 Avaliação Experimental dos Resultados

Para analisar o algoritmo, foi utilizada a ferramenta `hyperfine` para *benchmarking*, tendo sido gerados grafos com $V + E \in [150000, 1500000]$ através da ferramenta fornecida pela docência, `randGeneoTree`.




(a) Solução - LCA (escala $V + E$)

Figura 1: Complexidade temporal da solução apresentada para o problema proposto

Os valores obtidos demonstram uma tendência linear, considerando o eixo das abcissas com escala $V + E$, pelo que a complexidade temporal proposta para o algoritmo fica, assim, comprovada.