Relatório 2º projecto ASA 2021/2022

Grupo: tp037

Aluno(s): Joana Mendonça (83597)

Descrição do Problema e da Solução

Este projeto consiste em encontrar o/s nó/s que fossem ancestrais mínimos comuns de dois dados vértices. Numa primeira fase, tive de gerar um grafo e verificar se este era uma DAG, ou seja, se não continha ciclos. De seguida, gerei a árvore transposta e efetuei a função BFS nos dois vértices dados. Isto permitiu obter todos os ancestrais de cada um dos nós.

Após geradas duas listas com ambos os ancestrais de cada vértice, combinei os ancestrais comuns numa única lista, e depois percorri os nós dessa mesma lista e verifiquei primeiro se tinham filhos. Caso não tivessem, o nó era um LCA. Caso tivessem, verificava se os filhos desses nós se encontravam na lista, também. Caso não pertencessem, os nós pais eram LCAs, mas caso pertencessem, removia o nó pai da lista, pois este não consistia numa LCA, dado que tinha um filho que poderia ser um potencial candidato a LCA.

Análise Teórica

- Leitura dos dados de entrada: simples leitura do input usando `scanf()`, com ciclo(s) a depender linearmente com o tamanho das strings lidas. Logo, O(n)
- Complexidade das funções `isCyclic` e `isCyclicAuxiliar`, que verificam se um grafo contém ciclos ou não é O(V+E).
- A complexidade da aplicação de duas `BFS` sobre ambos os vértices dados para encontrar todos os ancestrais desse nó é O(V+E)
- A função `mergeLists`, que utiliza `set_intersection` para combinar os valores comuns entre duas listas numa só. O(min(n, m))
- Função LCA que itera sobre a lista de `common_ancestors` e de seguida, itera sobre a lista de `*adj` do grafo, sendo que a complexidade depende do tamanho de ambas. O(n.m)
- Apresentação dos dados num ciclo a depender do tamanho da lista de `common_ancestors` final. O(n)

Complexidade global da solução: O(n) + O(V+E) + 2*O(V+E) + O(min(n, m)) + O(n.m) + O(n) = O(nm)