Relatório 2º projeto ASA 2021/2022

Grupo: al070

Aluno(s): Luís Freire D'Andrade (94179) e Joana Maria de Brito (96037)

Descrição do Problema e da Solução

Segundo o enunciado, o 2º projeto de ASA 2021/2022 consiste, dado um grafo dirigido e dois dos seus vértices, em determinar se o grafo forma uma árvore genealógica válida e, caso forme, o conjunto de ancestrais comuns mais próximos entre os dois vértices. A nossa solução, implementada em linguagem C++, recorre a uma classe personalizada composta por um inteiro (número de vértices), um vetor de vetores de inteiros (lista de adjacências) e uma matriz de inteiros (lista de adjacências do grafo transposto), para a representação do grafo dirigido em que os vértices são apontados pelos seus progenitores.

A resposta ao primeiro problema é facilmente determinada. Por definição, apenas são impostas duas restrições para um grafo dirigido poder formar uma árvore genealógica: todo o vértice **(1)** tem, no máximo, dois progenitores e **(2)** não é progenitor de nenhum dos seus ancestrais. Traduzindo estas limitações para teoria de grafos, é simples compreender que, num grafo dirigido, **(1)** o número de progenitores se trata do grau de entrada de um vértice e **(2)** um arco de um nó para um dos seus ancestrais forma um caminho fechado no grafo, por outras palavras, forma um ciclo. Logo, um grafo dirigido constitui uma árvore genealógica se **(1)** nenhum dos seus vértices apresentar um grau de entrada superior a 2 e se **(2)** não contiver ciclos (DAG). Para garantir que nenhum vértice excede o grau de entrada 2, basta para isso monitorizar, na formação do grafo dirigido, cada arco adicionado e as respetivas atualizações dos graus de entrada dos seus vértices associados. Já para a deteção de ciclos é suficiente confirmar que, numa floresta DFS do grafo dirigido, não existem arcos para trás em nenhuma das suas árvores.

O segundo problema já não é tão prontamente computado. Um vértice é, inerentemente, um ancestral comum mais próximo entre dois vértices se **(1)** é ancestral dos dois, e se **(2)** não existe nenhum outro vértice descendente do mesmo tal que seja também ancestral dos dois. Mais uma vez, passando esta definição para teoria de grafos, num grafo dirigido transposto, um ancestral comum mais próximos entre dois vértices, P1 e P2, é apenas todo o vértice **(1)** alcançável a partir de P1 e P2 (ou seja, um ancestral comum entre P1 e P2) de modo que **(2)** não é alcançável a partir de nenhum outro ancestral comum. Na solução proposta, para obter o conjunto de ancestrais comuns entre dois vértices num grafo dirigido, é inicialmente aplicada uma procura em largura primeiro (BFS) a partir de um deles, com o objetivo de identificar e marcar todos os vértices que é possível alcançar (os seus ancestrais). De seguida, é aplicada uma segunda BFS a partir do segundo vértice com vista a encontrar vértices que foram anteriormente marcados (ancestrais comuns). Quando um vértice é identificado como alcançável a partir dos dois vértices dados, o mesmo é adicionado a um conjunto de possíveis soluções, e todas as suas consecutivas adjacências (todos os ancestrais do vértice identificado) são removidas como possíveis soluções, ou, removidas do conjunto de possíveis soluções, caso já lá estivessem contempladas.

Análise Teórica

Nesta análise teórica, dado um grafo de input, considera-se V como o seu número de vértices, E como o seu número de arcos, e P1 e P2 como dois dos seus vértices.

- Leitura dos dados de entrada, construção da árvore genealógica e verificação da sua validade: simples leitura do input e apuração da inexistência de vértices com grau de entrada superior a 2, com um ciclo a depender linearmente de E. Logo, $\Theta(E)$

Relatório 2º projeto ASA 2021/2022

Grupo: al070

Aluno(s): Luís Freire D'Andrade (94179) e Joana Maria de Brito (96037)

- Verificação da inexistência de ciclos no grafo: procura de arcos para trás, com a aplicação de uma DFS (dois ciclos encaixados que geram, no máximo, |V| + |E| iterações). Logo, O(V + E)
- Identificação (e marcação) dos ancestrais de P1: procura de vértices alcançáveis no grafo transposto, com a aplicação de uma BFS a partir de P1 (dois ciclos encaixados que geram, no máximo, |E| iterações). Marcação de cada vértice alcançável (tempo constante). Logo, O(E)
- Formação do conjunto de ancestrais comuns entre P1 e P2: procura de vértices alcançáveis já marcados, com a aplicação de uma BFS a partir de P2 (dois ciclos encaixados que geram, no máximo, |E| iterações). Adição de cada vértice alcançável a um conjunto ordenado (tempo logarítmico), e marcação (tempo constante) ou remoção dos seus ancestrais do conjunto ordenado (tempo logarítmico). Logo O(E · log(V))
- Apresentação dos dados. O(1)

Complexidade global da solução: O(E · log(V))

Avaliação Experimental dos Resultados

Para a experiência, foram gerados 10 grafos de tamanho incremental. De seguida, foi cronometrado o tempo de execução do programa para cada um dos grafos gerados. Como resultado, foi originado o gráfico da Figura 1.

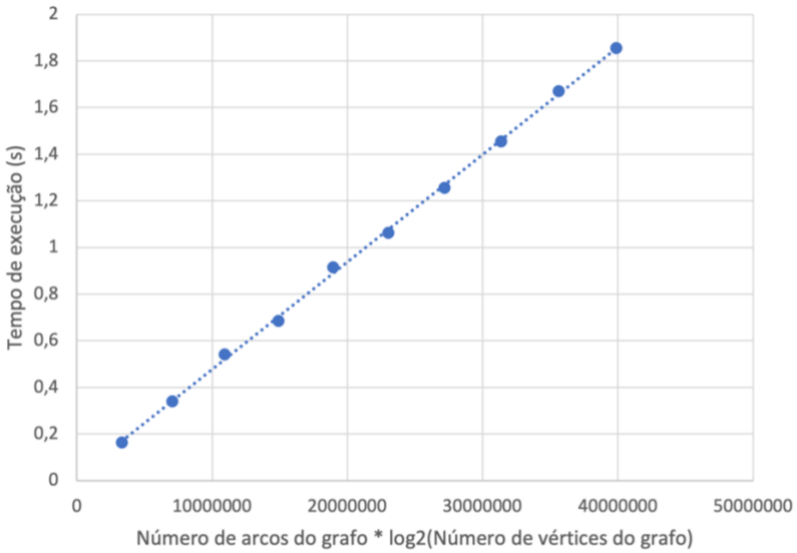

Figura 1 - Gráfico de Tempo de Execução em função de E · log(V)

Sendo que a linha de tendência linear do gráfico (com o tempo de execução em função da complexidade do respetivo problema) se revela bastante próxima de todos os pontos, pode-se concluir que o gráfico está concordante com análise teórica acima descrita, pois é possível observar que o tempo de execução do programa cresce linearmente com a complexidade apontada. Logo, a complexidade global da solução verifica-se: O(E · log(V)).