|  |  |  |  |
| --- | --- | --- | --- |
| **Número** | **Nome** | **Esforço (horas)** | **Contibuição ( %)** |
| 24058 | Rafael Rodrigues | 12h | 33.3% |
| 25391 | Bruno Roque | 12h | 33.3% |
| 25518 | Rafael Sousa | 12h | 33.3% |

**Descrição do Problema e da Solução**

O objetivo do problema é calcular as estimativas de quantos algoritmos computacionais (AC) que cada aluno do mestrado conhece, levando em consideração as relações de amizade entre eles. O algoritmo implementado para resolver esse problema é uma variação do Depth-First Search (DFS) adaptado para percorrer as relações de amizade e calcular as novas estimativas.

A vantagem de usar o algoritmo de busca em profundidade (DFS) é porque permite percorrer de forma eficiente as relações de amizade entre os alunos e atualizar as estimativas de conhecimento.

A desvantagem de usar o algoritmo de busca em profundidade deve-se ao facto de ser ineficiente em casos onde o grafo de amizades é muito grande e complexo, levando a um tempo de execução maior.

A implementação do algoritmo foi feita em Python

**Análise Teórica**

**Tabela 1.** Pseudocódigo e complexidade temporal da solução proposta.

|  |  |  |
| --- | --- | --- |
| **Pseudocódigo** | **Etapa** | **Execução** |
| Para qualquer estudante existente na lista de estudantes , caso o amigo não tenha sido visitado , repete-se a si mesmo até que todos tenham sido visitados | 1º- criamos uma função para o DFS | **Ω = N2** |
| Para todos os dados recebidos divide e junta em um vetor | 2º- temos um for que ira agrupar toda a informação dos alunos dentro de uma lista | **Ω = N** |
| Para todas as ligações de amizade acrescenta a AC’s conhecidas aos alunos respetivos | 3º- fizemos outro for mas agora para dar os conhecimentos de AC’s partilhados pelos alunos | **Ω = N** |
| Para qualquer estudante que não foi visitado então correr a função criada inicialmente | 4º- criamos mais um for com o objetivo de usar a função DFS criada anteriormente onde iremos agrupar todos os resultados dados pelas outras funções | **Ω = N3** |
| Para todos os resultados mostrar os resultados | 5º- output que teve de ser feito com um ciclo for | **Ω = N** |

**Referências**

[1] [5.1 Graph Traversals - BFS & DFS -Breadth First Search and Depth First Search - YouTube](https://www.youtube.com/watch?v=pcKY4hjDrxk&list=PLDN4rrl48XKpZkf03iYFl-O29szjTrs_O&index=62)

[2] [BFS DFS - Simplified - YouTube](https://www.youtube.com/watch?v=kyUpc_5705s&list=PLDN4rrl48XKpZkf03iYFl-O29szjTrs_O&index=82)

[3] <https://www.w3schools.com/python/python_intro.asp>

[4]Manual de Algoritmos disponibilizado pela professora.