Marina Godinho Domingues (18103928)
Matheus Bruhns Bastos (15100950)
Matheus Felipe Souza Valin (14200949)

# Atividade A2 - Relatório

Grafos (INE5413)

Professor: Rafael de Santiago

02 de Junho de 2019

# Conteúdo

# 1 Exercício 1 - Componentes Fortemente Conexas (Kosaraju)

A linguagem de implementação foi mantida como **Python**, dada compatibilidade com a implementação da primeira atividade da disciplina. A respeito de novas estruturas implementadas, foram feitas as seguintes escolhas:

- Stack: é uma fila criada a partir do método *dfsVisit()*, que serve para enfileirar os vértices do grafo através da finalização da sua expansão na busca em profundidade. A estrutura de dados utilizada foi o *deque*, devido à sua capacidade de realizar inserções no início da lista com complexidade de tempo $O(1)$, através do método *appendleft(item)*.

- Classes: é uma lista que contém as classes do grafo, sendo que um par de vértices pertencem à mesma classe se e somente se são parte da mesma componente fortemente conexa. Essa lista é de fato um objeto do tipo lista em Python, pois a principal utilização envolve o método *append(item)*, que para listas possui complexidade de tempo $O(1)$.

- Grafo Reverso/Transposto: foi necessário, para implementação do algoritmo de Kosaraju (para componentes fortemente conexas), a realização de um grafo reverso. Essa implementação foi realizada através de um método *getGrafoTransposto(grafo)*, que remonta o grafo com inversão de arcos. Para manter as estruturas otimizadas da classe Grafo, a implementação desse método tem complexidade $O(|V| + |A|)$, onde $|V|$ é o número de vértices e $|A|$ é o número de arcos do grafo original.

# 2 Exercício 2 - Ordenação Topológica

Para ordenação topológica foi utilizado o algoritmo baseado na busca em profundidade. A implementação utiliza uma versão modificada da estrutura utilizada no exercício anterior para constituição do grafo, uma vez que o grafo agora deve ser orientado. Esta foi a única modificação uma vez que a classe anterior já suportava grafos rotulados. A nova classe utilizada é chamada 'GrafoDirigido'. O Algoritmo resulta numa lista dos índices dos vértices, ordenada de acordo com a ordenação topológica. A partir dessa lista, é acessado o rótulo de cada um dos vértices, em ordem, e esses são exibidos na tela conforme requisitado, com uma seta -¿ "indicando o sequenciamento.

As estruturas de dados são semelhantes às utilizadas em outros algoritmos já citados (neste ou no relatório anterior). Vale ressaltar novamente que a estrutura de fila utilizada nesse algoritmo também é na verdade um *deque* ao invés de uma lista,

devido à necessidade de inserção de elementos no início da fila (implementado através do método *appendleft(item)*.

# 3 Exercício 3 - Árvore Geradora Mínima (Prim)

O algoritmo de Prim faz uso de uma estrutura de pilha, e para isso foi utilizada uma lista em Python, dado que todas as manipulações correspondem à remoção de itens do final da lista (que é executado com complexidade de tempo $O(1)$). Outros elementos utilizadas correspondem a estruturas já abordadas e justificadas, sem novos critérios ou variações.

# 4 Observação

É possível, também, encontrar o código, além das instruções para uso em:
`https://github.com/Matbbastos/ine5413-grafos`