12/09/2024

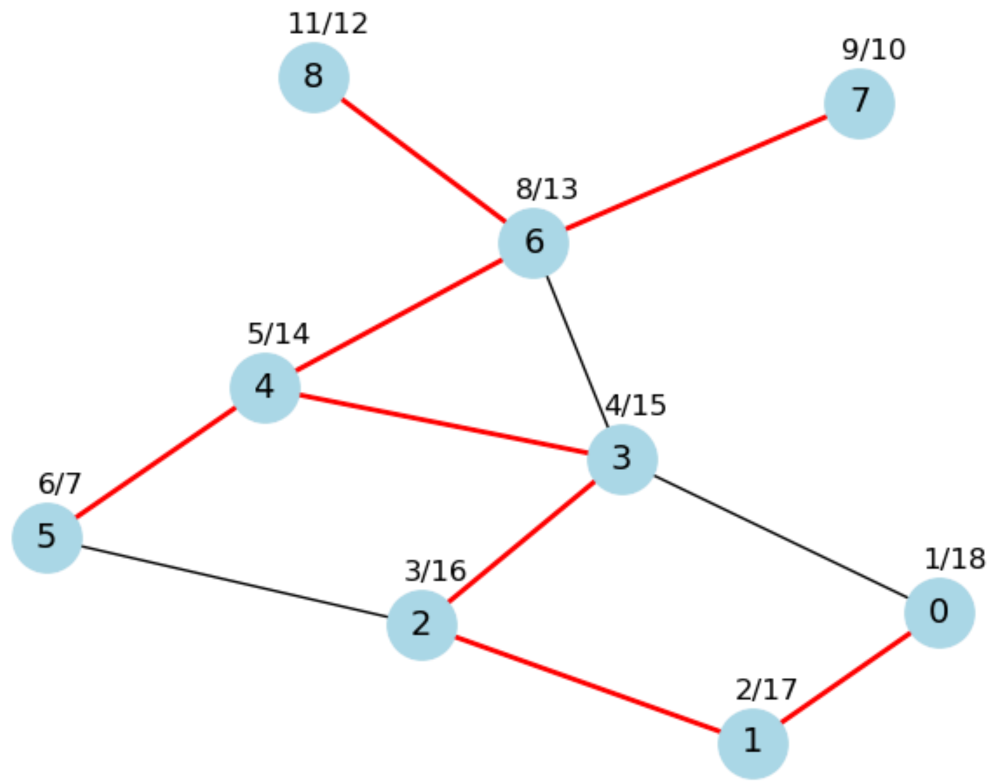
Aulão TGC: Primeira Prova

Fernando C. S. Dal' Maria

PUC Minas

Busca em Profundidade

Considere G = (V, E) um grafo **conexo**. A ideia central do algoritmo é explorar o máximo possível ao longo de cada ramificação antes de retroceder. A busca começa em um vértice *s* pertencente à V e se aprofunda no grafo até que todos os vértices de V tenham sido atingidos. Logo, considere S um conjunto que se inicia com *s* e enquanto S for diferente de V:



1 - A partir de *s* siga para um vértice vizinho *u* que não esteja em S, adicione *u* em S e repita o processo até alcançar um vértice sem vizinhos não visitados.
2 - Neste ponto, você deve retroceder (backtracking) até o último vértice que ainda possui vizinhos não contidos em S e continuar realizando os passos descritos no tópico anterior.

Fonte: Jon Kleinberg & Eva Tardos. Algorithm Design. ed. 1; p. 83; 2006.
Monitor: Fernando Campos Silva Dal' Maria

PUC Minas

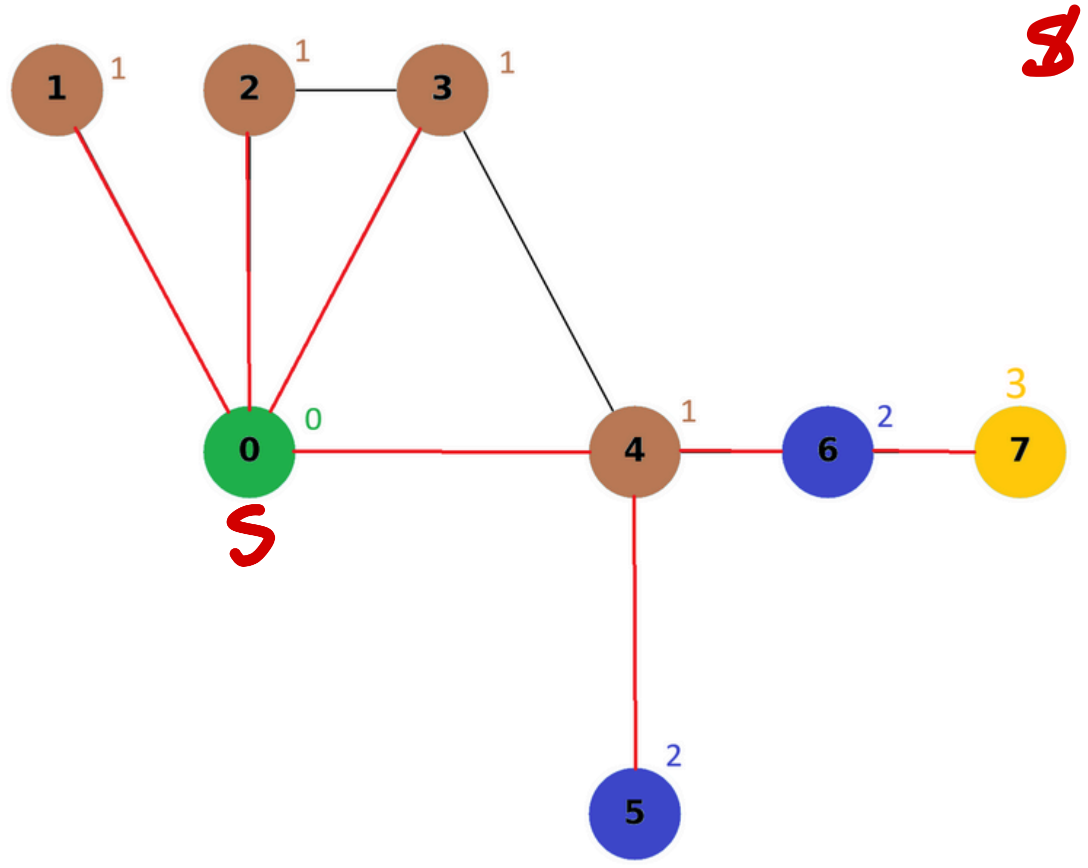
Busca em Largura

fila: 0

~~0 1 2 3 4~~

~~5 6 7~~

Considere G = (V, E) um grafo **conexo**. A ideia central do algoritmo de busca em largura é explorar todos os vértices de um mesmo nível de proximidade antes de passar para os vértices do próximo nível. A busca começa em um vértice *s* pertencente à V e segue para todos os vértices adjacentes antes de mover-se para os vértices do próximo nível. Dessa forma, considere S um conjunto e Q fila que inicialmente contêm apenas *s*. Enquanto S for diferente de V:



1 - Desenfileire o próximo vértice da fila. Considere esse vértice como *v*.

2 - Para cada vizinho *u* de *v*, caso *u* não esteja contido em S, adicione-o em S e enfileire-o em Q.

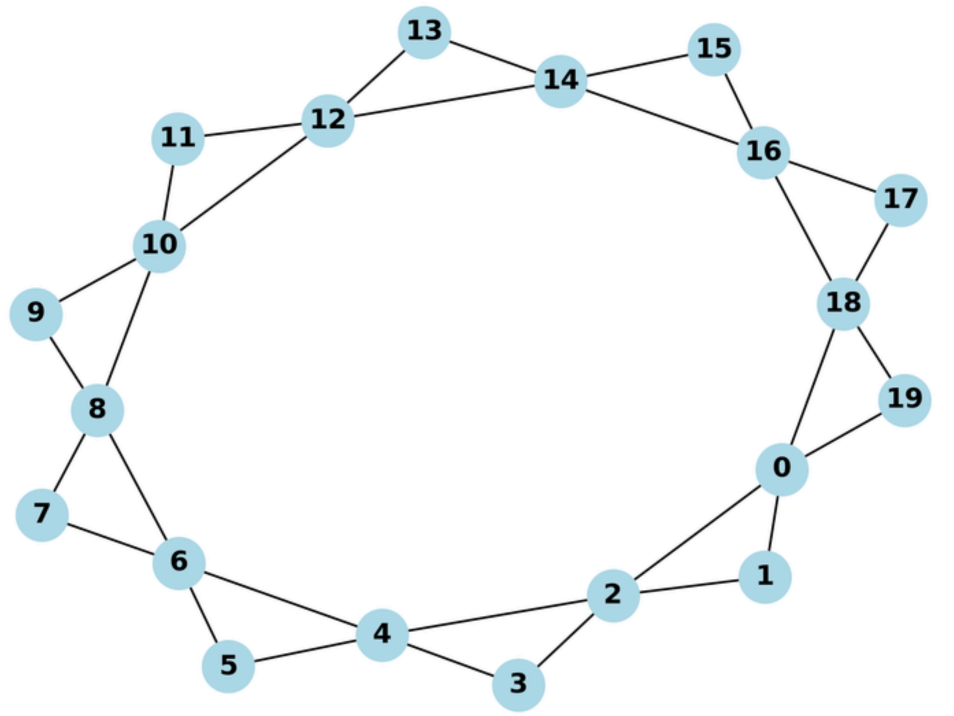
Fonte: Jon Kleinberg & Eva Tardos. Algorithm Design. ed. 1; p. 79; 2006.

Monitor: Fernando Campos Silva Dal' Maria

PUC Minas

Caminhamento Euleriano

Considere G = (V, E) um grafo **não direcionado e conexo**. Inicialmente é necessário que G apresente graus pares para todo *v* pertencente à V (condição para ciclo euleriano) ou que G contenha exatamente dois vértices de grau impar (condição para caminho euleriano). Sendo assim, o algoritmo para verificar se o grafo é euleriano ou semi-euleriano é:



1 - Realizar uma busca em profundidade, partindo de um vértice de grau impar sempre que possível.
2 - Durante a busca em profundidade priorizar aquelas arestas que não desconectam o grafo.
3 - Quando alcançar o vértice inicial ou não for mais possível progredir na busca o algoritmo se encerra.

Fonte: Douglas B. West. Introduction to Graph Theory. ed. 2; p. 26; 2001.
Monitor: Fernando Campos Silva Dal' Maria

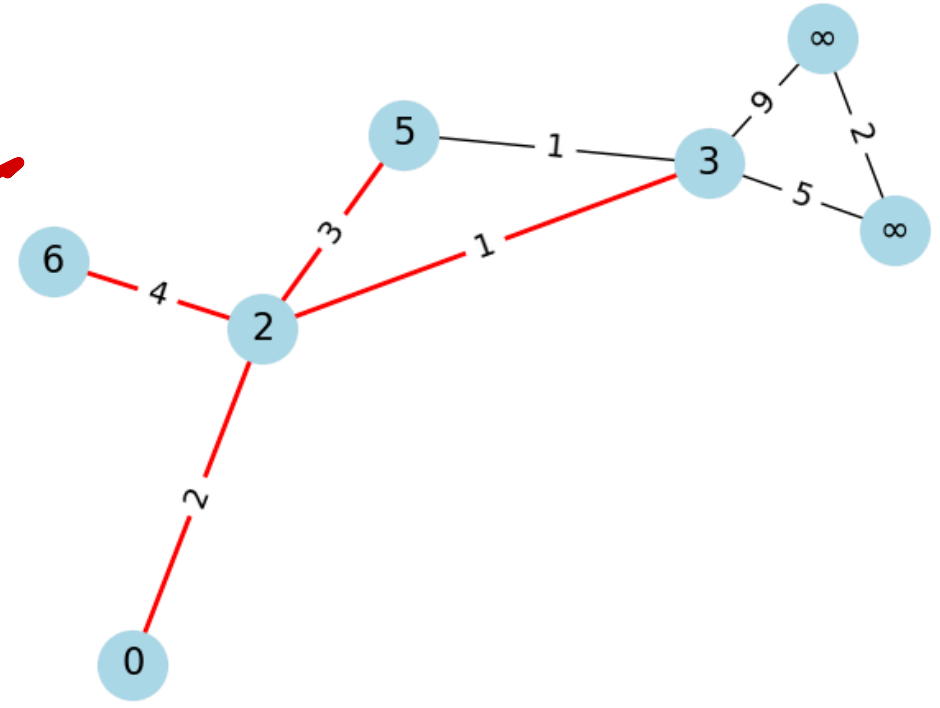
PUC Minas

Menor Caminho

Considere G = (V, E) um grafo **não direcionado, conexo e ponderado em arestas**. Seja *s* um vértice pertencente à V crie um vetor distância D contendo 0 para *s* e infinito para todos os demais. Além disso, considere S o conjunto de todos os vértices já visitados pelo algoritmo e Q uma fila de prioridade. Inicialmente, S e Q devem conter *s*. Dessa forma, siga a seguinte sequência de passos enquanto S for diferente de V.

-1

ou enquanto a fila não for vazia!



1 - Selecione em Q o vértice *w* com a menor distância em D e remova-o de Q.
2 - Para todos os vizinhos *u* de *w* não contido em S verifique se D[*u*] > D[*w*] + peso({*u*, *w*}).
3 - Em caso positivo atualize D[*u*] para D[*w*] + peso({*u*, *w*}), atualize Q para Q U {*u*} e S para S U {*u*}.

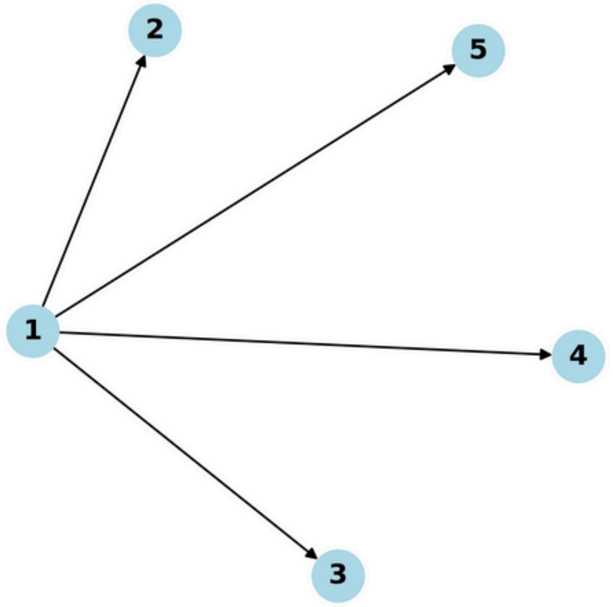
Fonte: Jon Kleinberg & Eva Tardos. Algorithm Design. ed. 1; p. 137; 2006.
Monitor: Fernando Campos Silva Dal' Maria

PUC Minas

Conceitos sobre Base

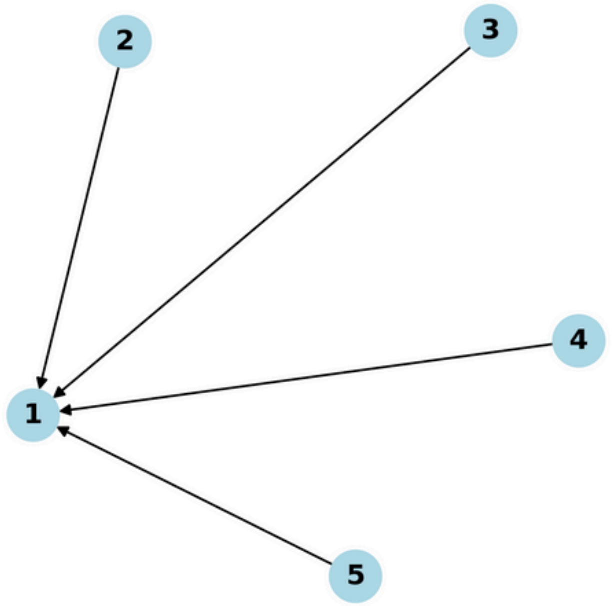
Base de um grafo dirigido G = (V, E):

- É um subconjunto B de V tal que não há caminho entre vértices de B e todo vértice não pertencente a B pode ser atingido por algum vértice de B.



Anti-base de um grafo dirigido G = (V, E):

- É um subconjunto A de V tal que não há caminho entre os vértices de A e todo vértice não pertencente a A pode atingir A por um caminho.

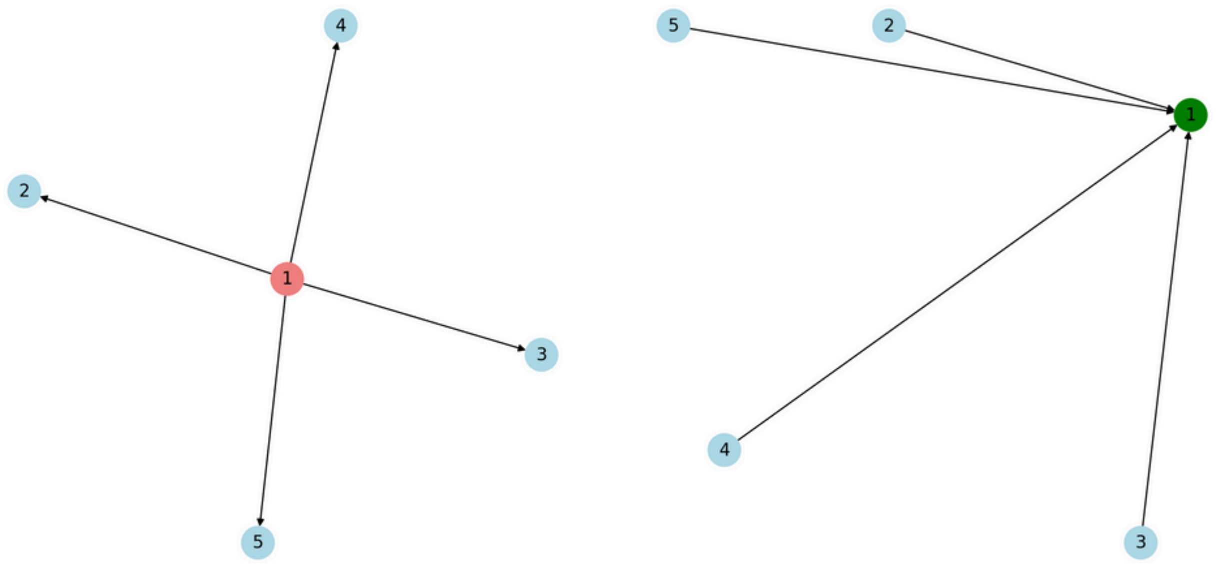


Fonte: Jamil, Silvio.
Monitor: Fernando Campos Silva Dal’ Maria

PUC Minas

Identificação de Base

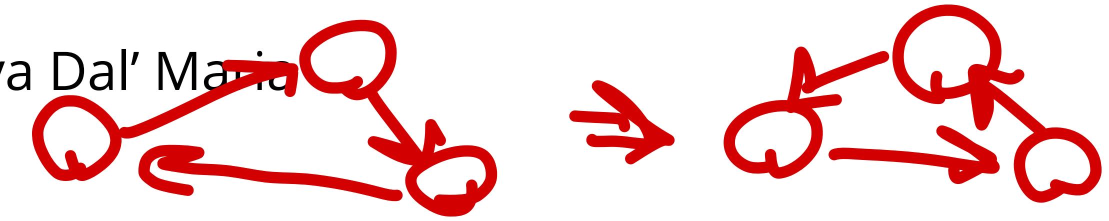
Considere G = (V, E) um **grafo dirigido**. Após calcular o grau de entrada de todos os vértices, identifique os vértices com grau de entrada igual a zero. Esses vértices não são alcançados por nenhum outro. Consequentemente, se houver outros vértices no grafo com grau de entrada maior que zero, eles serão alcançados diretamente ou indiretamente a partir desses vértices com grau de entrada zero.



1 - **Caso o grafo seja cíclico:** é preciso realizar uma abstração de modo que os vértice que compõem o ciclo sejam contraídos para um hipervértice. Quando o novo vértice tiver grau de entrada zero um dos vértices que compõem aquele ciclo deve ser selecioado.

2 - **Cálculo de anti-base:** obtenha o grafo transposto de G e aplique o algoritmo a ele.

→ mesmos vértices e arestas, mas com sentidos opostos.



Fonte: Jamil, Silvio.

Monitor: Fernando Campos Silva Dal' Maria

PUC Minas

SCC - Kosaraju

algoritmo para encontrar os componentes fortemente conectados (SCCs)

(DFS)

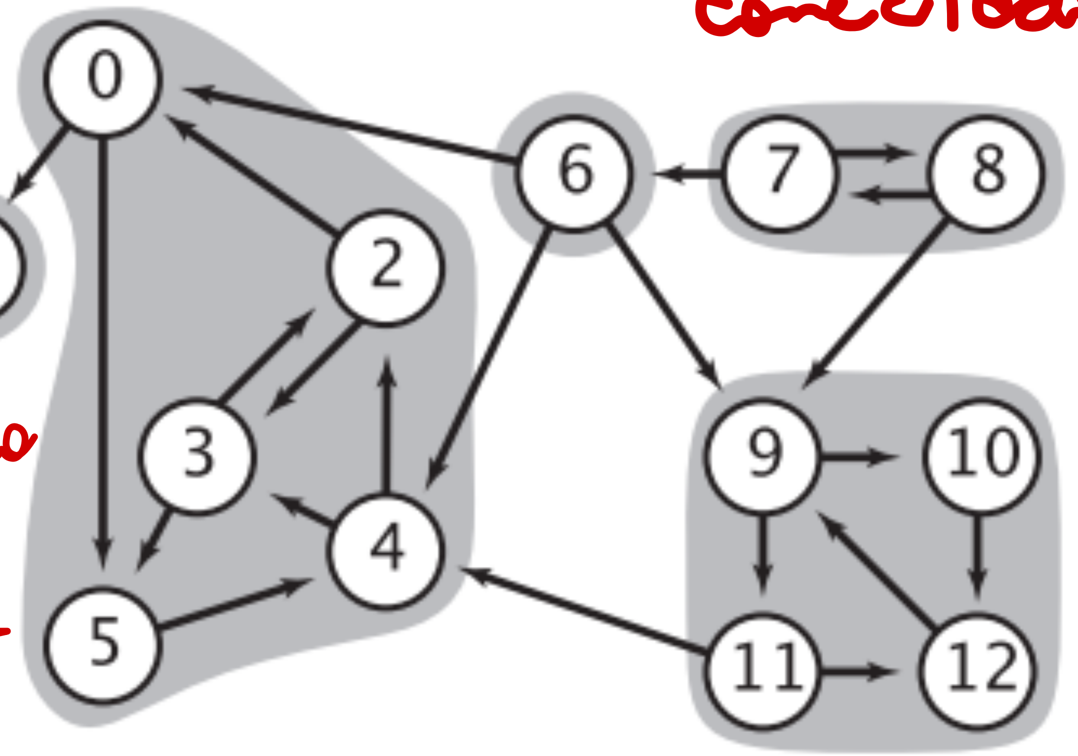
1 - Realizar uma **busca em profundidade** em G = (V, E), granvando os tempos de início e fim da visitação

↳ ao final disso, temos os vértices empilhados em ordem decrescente de tempo de término

2 - Na ordem inversa dos tempos de término (**ordem topológica**), realizar uma busca em profundidade no **grafo transposto**, T de G

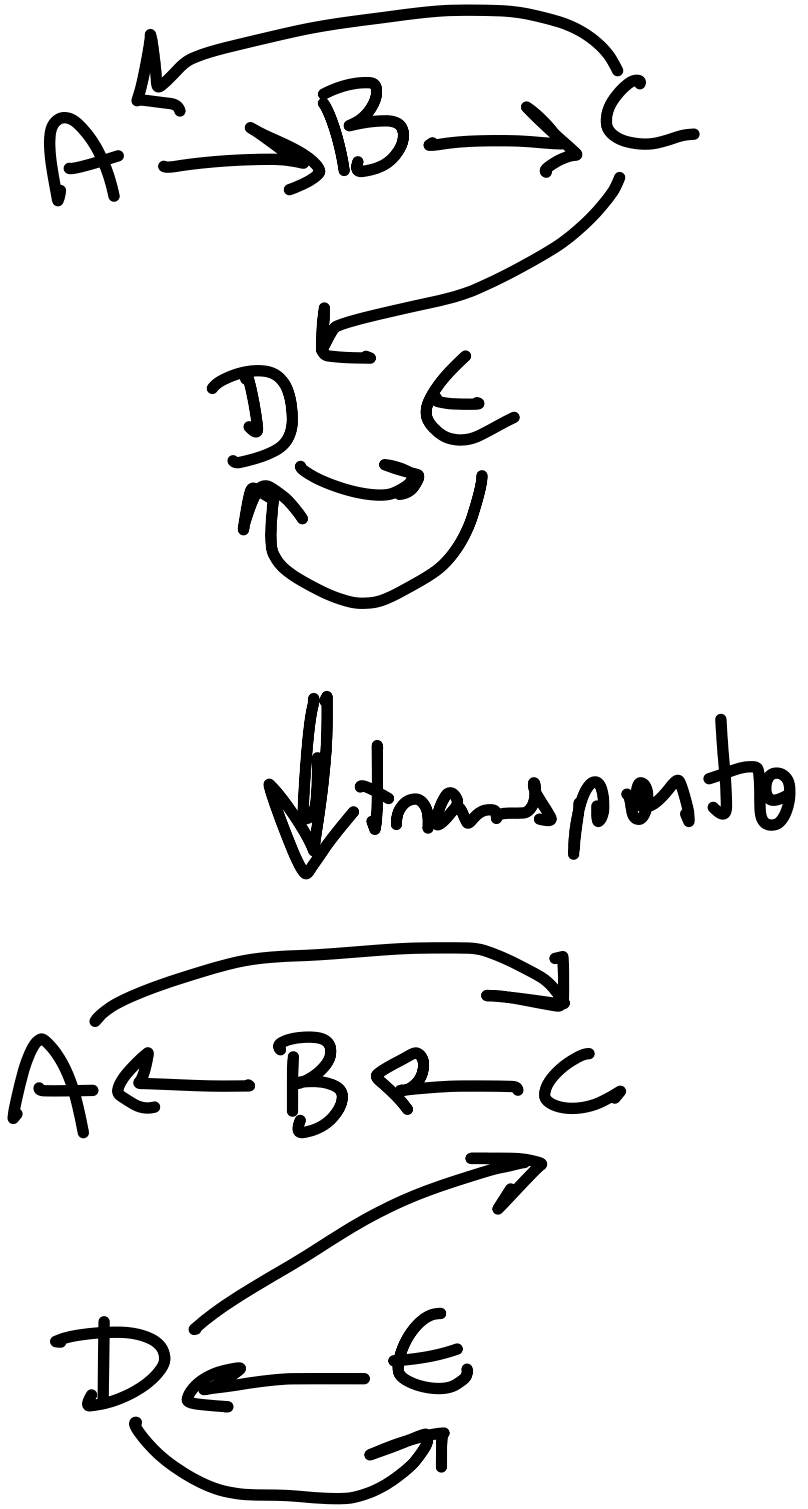
→ desempilhando os vértices da pilha criada no passo 1

3 - Para cada conjunto de vértices visitados em T, marque esse conjunto como um **componente fortemente conexo (scc)** em G.



Fonte: Robert Sedgewick & Kevin Wayne. Algorithms. ed. 4; p. 584; 2011
Monitor: Fernando Campos Silva Dal' Maria

PUC Minas



1- busca em prof. começando por A → A, B, C, D, E

|A| |B| |C| |D| |E| → (termina em D)

~~A~~ volta p/ A

2- busca em prof. na ordem decrescente da pilha no transposto → $\boxed{\{D, E\} \text{ e } \{A, B, C\}}$

|D| |E| ~~D~~ → até voltar p/ D

|A| |C| |B| ~~A~~ (voltou p/ A)

Obrigado!

fernandocsdm@gmail.com

PUC Minas