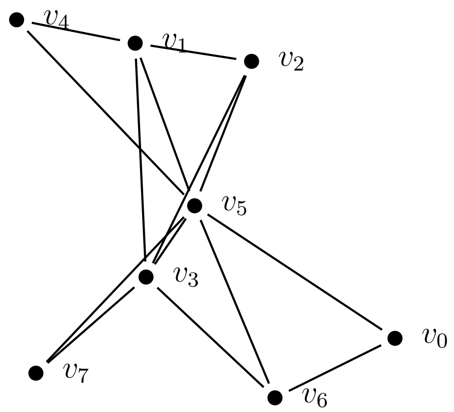
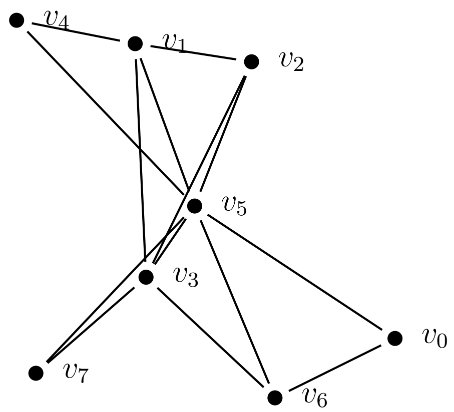
MCTA028-17 – Teoria dos Grafos
Centro de Matemática, Computação e Cognição
Universidade Federal do ABC
Profa. Carla Negri Lintzmayer

Lista 2: Buscas em grafos

1. Execute o algoritmo de BFS sobre o grafo exibido na figura abaixo. Para cada iteração do algoritmo, exiba a configuração corrente dos vetores *visitado*, *pred*, *D* e da fila.



2. Execute o algoritmo de DFS sobre o grafo exibido na figura abaixo. Para cada iteração do algoritmo, exiba a configuração corrente dos vetores *visitado*, *pred*, *ordem* e da pilha.

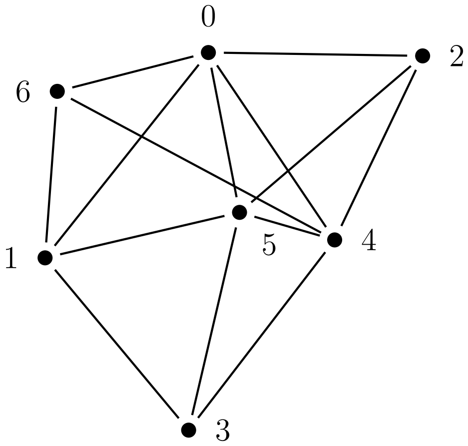


3. O que a árvore de BFS nos diz sobre a distância de um vértice x a outro y quando nenhum deles é a raiz?

4. Explique como podemos modificar a BFS e a DFS para eliminar o vetor *visitado*.

5. Faça um algoritmo baseado na DFS que, dado um grafo de entrada, identifique seus vértices de corte.

6. Os vetores abaixo representam os vetores *pred* obtidos após a execução dos algoritmos de Busca Genérica, BFS e DFS no grafo abaixo. Identifique o algoritmo utilizado para construir cada vetor e justifique a sua resposta.



(a)

0	1	2	3	4	5	6
1	3	5	3	3	3	1

(b)

0	1	2	3	4	5	6
4	0	5	5	4	1	1

(c)

0	1	2	3	4	5	6
0	0	5	1	3	3	4

(d)

0	1	2	3	4	5	6
6	6	0	1	3	3	6

7. Seja G um grafo conexo, seja T uma árvore de DFS de G, e seja $uv \in E(G)$. Prove que se $uv \notin E(T)$ e $ordem[u] < ordem[v]$, então u é um ancestral de v.

8. Prove que um grafo conexo G é acíclico se e somente se nenhuma aresta de retorno for encontrada por uma DFS sobre G.

9. Seja uv uma aresta de uma árvore de DFS, com $pred[v] = u$. Prove que essa aresta é de corte se e somente se não existe uma aresta de retorno xy onde x é um descendente de v e y é um ancestral de u (note que é possível ter $x = v$ ou $y = u$).

10. Escreva um algoritmo que execute em tempo $O(V)$ e que decida se um dado grafo contém ou não um ciclo. Neste exercício, você deve:

    (a) Fornecer um pseudocódigo;

    (b) Fornecer uma breve justificativa da correção do seu algoritmo (não precisa ser uma demonstração formal, apenas justifique o porquê do seu algoritmo funcionar, quais as propriedades que o mesmo mantém).

    (c) Analisar o tempo de execução.

11. Explique como podemos modificar a BFS para verificar se um grafo G é bipartido. Além da explicação, você deve fornecer um pseudocódigo que implemente a sua ideia. Caso G seja bipartido, sua função deve exibir uma bipartição de G preenchendo um vetor cor, indexado pelos vértice de G, tal que $cor[u] = cor[v]$ se e somente os vértices u e v estão na mesma parte da bipartição. Caso G não seja bipartido, sua função deve exibir um certificado de que G não é um grafo bipartido.

12. Um grafo é *biconexo*, ou 2-*conexo*, se não possui aresta de corte. Uma componente *biconexa* de um grafo G é um subgrafo biconexo maximal de G. Faça um algoritmo baseado em DFS que identifique as componentes biconexas de um grafo G em tempo $O(V+E)$. Para identificar cada componente, você deve listar as arestas pertencentes ao subgrafo que forma a componente.

13. Explique como usar a BFS para calcular a cintura de um grafo. A *cintura* de um grafo é o tamanho do menor ciclo contido nele.

14. Prove que o algoritmo a seguir encontra corretamente o diâmetro de uma árvore.

    - Execute a BFS a partir de um vértice w qualquer para encontrar um vértice u que está à maior distância de w.
    - Execute a BFS novamente, agora a partir de u, para encontrar um vértice v que está à maior distância de u.
    - Devolva $dist(u, v)$.