

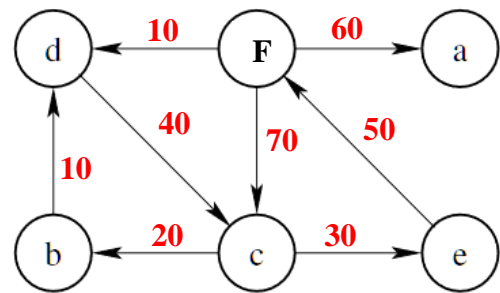
**Algoritmos e Estrutura de Dados II**  
**Prof. Hélder Pereira Borges**  
**Atividade de Grafos**  
**OBSERVAR AS REGRAS DESCRITAS NA UNIDADE 01**

**\*\*\* Atividade em Dupla \*\*\***

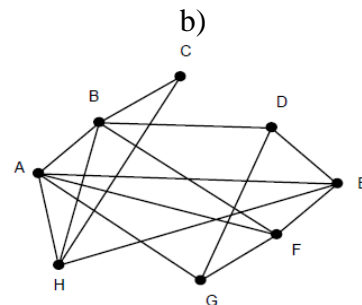
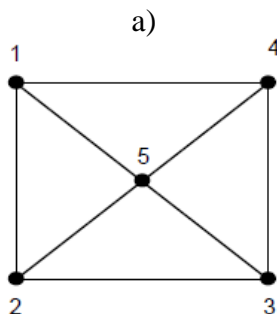
Desenvolver soluções para as questões que seguem. Além disto, cada dupla receberá o projeto de outra dupla, devendo avaliar a solução proposta, para então produzir uma resenha e comparativo entre as soluções, **atribuindo uma nota (0 a 10) a ambas**. O projeto com deve ser enviado no classroom da disciplina através da **Atividade Grafos - Projeto** e o arquivo de avaliação em **Atividade Grafos - Avaliação**. Os projetos serão distribuídos pelo professor para cada equipe avaliar.

**ATENÇÃO:** Para esta atividade, o nome do projeto deve ser apenas **ATV\_Grafos** e não deve haver seus nomes nos comentários no código.

1. Considerando um grafo orientado G, apresente a ordem dos vértices visitados, tanto na busca em **Largura** quanto em **Profundidade**. O vértice inicial pode ser especificado no código, ou seja, deve algo similar na main: **DFS (grafo, verticeInicial)**.

B;D;10 C;B;20 C;E;30 D;C;40 E;F;50 F;A;60 F;C;70 F;D;10		BFS: F A C D E B DFS: F A C E B D ou F A C B D E
--	---	--

2. Pesquise sobre o algoritmo de **Welsh-Powell** e o utilize sua lógica para colorir e determinar o número cromático de grafos como os que seguem.



3. Dado um grafo G, direcionado e ponderado, representando um mapa, com um conjunto de cidades, suas estradas e distâncias, desenvolva uma versão do **algoritmo de Dijkstra** que retorne a **maior** distância e a rota entre duas cidades quaisquer. As cidades origem e destino, podem ser especificadas no código, ou seja, deve algo similar na main: **Dijkstra (grafo, cidadeOrigem, cidadeDestino)**.

4. Dado um grafo qualquer, não direcionado, conexo e ponderado, desenvolva uma solução que resolva o Problema do Caixeiro Viajante a partir dos **algoritmos de Prim e Kruskal**.
5. Considerando um grafo dirigido qualquer, desenvolva um algoritmo que apresente uma **ordenação topológica** válida.