**Questão 1**

(**F**) Se for fornecido ao programa uma entrada de dados com dois elementos por linha, ele vai construir um grafo na memória utilizando dicionários. Nesse caso os pares de elementos são os valores das arestas;

(**F**) O programa pode ser usado sem modificações para a entrada de dados em aplicações que montam grafos normais e grafos dirigidos (dígrafos);

(**F**) O programa usa uma estrutura de dicionário para montar o grafo na memória. Nesta estrutura o nome do vértice é a chave e as propriedades ficam armazenadas em uma lista de adjacências pertencente à chave em questão;

(**V**) O programa usa uma propriedade que é uma representação em lista de adjacências para cada vértice. Nessa lista ficam armazenados os nomes dos vértices que estão ligados por uma aresta ao vértice dono da lista;

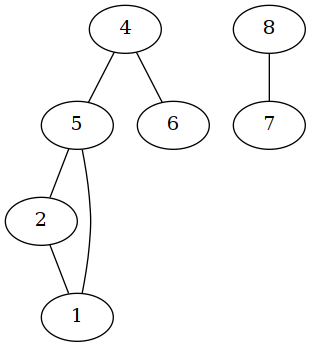
(**F**) O programa apresenta redundância nas linhas 10 e 14, pois as linhas 8 e 12 já adicionamo vértice armazenado na variável v2 na lista de v1 (linha 8) e o vértice armazenado em v1 na lista de v2 (linha 12);

(**V**) O programa é imune ao fornecimento duplicado de um mesmo par, desde que seja modifi cada a ordem dos vértices, ex.: 1 2 e 2 1;

(**V**) O programa não permite que seja informado um vértice solitário sem nenhuma aresta para outros vértices.

**Questão 2**

1. 4 5; 2 1; 1 5; 2 5; 4 6; 7; 8



**Lista de Adjacências:**

{1} = [2, 5]

{2} = [1, 5]

{5} = [4, 2]

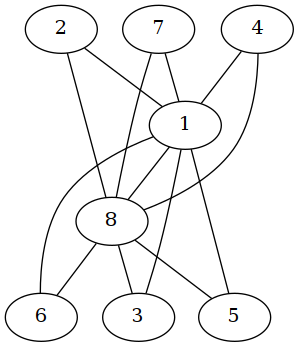
{4} = [5, 6]

{6} = [4]

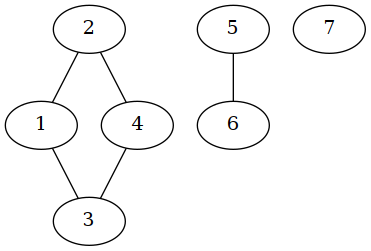
{7} = [8]

{8} = [7]

1. 1 2; 1 3; 1 4; 1 5; 1 6; 1 7; 2 8; 3 8; 4 8; 5 8; 6 8; 7 8; 8 1;



1. 1 2; 3 4; 1 3; 2 4; 5 6; 7



**Questão 3**

(**V**) Pode-se deixar o algoritmo independente de um valor teórico para infinito inicializando “INF” como sendo o número de vértices + 1;

(**V**) Pode-se utilizar o algoritmo para descobrir se o grafo é conexo, ou seja, a partir de qualquer vértice é possível chegar a qualquer outro vértice;

(**F**) Caso o algoritmo seja aplicado a um grafo desconexo, após a conclusão dele, todos os vértices serão pretos ou cinzas (aqueles por onde a recursividade não retornou);

(**V**)Após a conclusão do algoritmo da busca em largura, sabe-se a distância mínima (em número de vértices) do vértice de origem s para todos os outros vértices;

(**F**) O algoritmo não funciona em grafos que possuem ciclos fechados por utilizar um marcador cinza para os vértices já visitados previamente. Neste caso é mais correto manter o vértice branco**.**

**Questão 4**

(**F**) O algoritmo da busca em profundidade funciona de forma semelhante ao algoritmo de busca em largura. Ele primeiro acessa todos os vértices que estão a uma mesma distância do vértice de origem para só então passar para os vértices mais distantes;

(**F**) O algoritmo sempre atinge todos os vértices a partir do caminho mais curto a partir do vértice de origem;

(**V**) A condição de parada da recursividade utilizada no algoritmo é encontrar um vértice que é preto;

(**V**) O campo f tem seu valor atribuído após o vértice ter todos os vértices adjacentes visitados;

1. Caso o algoritmo da busca em profundidade seja aplicado a um grafo não conexo ele não conseguirá acessar todos os vértices do grafo, sendo necessário chamá-lo mais de uma vez, uma para cada “ilha” vértices.

**Questão 5**

**{2}= 1**

**{3}= 1**

**{8}= 2**

**{6}= 2**

**{7}= 2**

**{10}= 3**

**{5}= 1**

**{1}= 0**

**{4}= 1**

**{9}= 3**

**Questão 6**

**Obs:**

**Responder depois**

**Questão 7**

Resposta no Arquivo DFS.py

**Questão 8**