***def calcular\_grau(chave):***

***return len(grafo[chave][0])***

***def mostrar\_conjunto():***

***print(grafo)***

***def mostrar\_caminho(dicionario):***

***for key, value in dicionario.items():***

***print(key, value)***

***def depth\_dirst\_search(dicionario, inicio, fim):***

***pilha = [(inicio, [inicio])]***

***while pilha:***

***vertice, caminho = pilha.pop()***

***for proximo in set(dicionario[vertice][0]) - set(caminho):***

***if proximo == fim:***

***yield caminho + [proximo]***

***else:***

***pilha.append((proximo, caminho + [proximo]))***

***def sair():***

***return ""***

***def criar\_vertices(lista):***

***for x in range(len(lista)):***

***if not lista[x] in grafo:***

***for key in lista[x]:***

***grafo[key] = [[],[]]***

***def criar\_arestas(no\_1, no\_2, peso):***

***if no\_1 in grafo and no\_2 in grafo:***

***grafo[no\_1][0] += no\_2***

***grafo[no\_1][1] += peso***

***else:***

***print('Não existe vértices declarados para ligar este caminho')***

***grafo = {}***

***lista\_vertices = []***

***while (True):***

***print('Digite 1: Adicionar vértices e Arestas')***

***print('Digite 2: Calcular o grau de um dado vértice')***

***print('Digite 3: Mostrar os conjuntos de vértices e arestas')***

***print('Digite 4: Responder se um vértice é alcançável diretamente a partir de outro')***

***print('Digite 5: Depth Dirst Search:')***

***print('Digite 0: Sair')***

***argument = input()***

***if argument == '0':***

***print('Programa Finalizado')***

***break***

***elif argument == '1':***

***print('Adicione os vértices (em maiúsculo): ')***

***while (True):***

***elemento = input()***

***if elemento == '0':***

***break***

***else:***

***lista\_vertices.append(elemento)***

***criar\_vertices(lista\_vertices)***

***print('Adicione as arestas (em maiúsculo), para sair 0 e 0: ')***

***while True:***

***aresta\_x = input('De: ')***

***aresta\_y = input('Para: ')***

***if aresta\_x == '0' or aresta\_y == '0':***

***break***

***else:***

***valor = input('Peso: ')***

***criar\_arestas(aresta\_x, aresta\_y, valor)***

***elif argument == '2':***

***maior\_grau = max(grafo, key=calcular\_grau)***

***print(maior\_grau)***

***print(grafo[maior\_grau][0])***

***elif argument == '3':***

***mostrar\_conjunto()***

***elif argument == '4':***

***mostrar\_caminho(grafo)***

***elif argument == '5':***

***inicio\_vertice = input('Diga qual o vértice de início: ')***

***fim\_vertice = input('Diga qual o vértice de saída: ')***

***caminhos = list(depth\_dirst\_search(grafo, inicio\_vertice, fim\_vertice))***

***if caminhos: # Se a lista não estiver vazia***

***print('Todos os possíveos caminhos: ', caminhos)***

***print('Caminho mais curto: ', min(caminhos, key=len))***

***else:***

***print('Este caminho não é possível')***

***else:***

***print('Valor incorreto')***

***# https://pt.stackoverflow.com/questions/67593/grafo-caminhos-poss%C3%ADveis-python***

***# algoritmo de dijkstra***