Questão 1: A primeira questão consiste na implementação do algoritmo de Edmonds-Karp para encontrar o fluxo máximo de um grafo dirigido. Dessa forma, é necessário a utilização da implementação do grafo dirigido. A implementação está dividida em três funções: EdmondsKarp, getGrafoResidual e FordFulkerson. A função FordFulkerson é a principal e executa o algoritmo de Ford Fulkerson com base em um grafo G, um vértice fonte s de G e um vértice sorvedouro t. Essa função utiliza a função getGrafoResidual, que cria e retorna um novo grafo dirigido chamado de grafo residual Gf. A partir dessas variáveis, a função EdmondsKarp é chamada para encontrar caminhos aumentantes p utilizando listas para a implementação das estruturas de dados C, A e Q, sendo que a lista Q é, na verdade, uma fila porque os elementos sempre são inseridos no final e removidos do início. Dessa forma, a função FordFulkerson executa enquanto a função EdmondsKarp retornar algo que não seja “None”, sempre incrementando o valor de fluxo máximo. Quando a função EdmondsKarp retornar None, a função FordFulkerson imprime o valor do fluxo máximo.

Questão 2: Para a questão do Hopcroft-Karp, o arquivo contendo o grafo bipartido de entrada não possui a parte dos pesos das arestas, já que também se trata de um grafo não dirigido. Sendo que, ao ser lido, é formado um grafo não dirigido com dois atributos adicionais, a lista de vértices pertencentes a X e a lista de vértices pertencentes a Y. Ao todo há três funções que descrevem o algoritmo. A primeira, “HopcroftKarp”, possui os vetores “D”, para armazenar as distâncias de cada vértice, sendo que uma dessas posições (posição zero) armazena um vértice artificial que é utilizado em outra função, e “mate”, para guardar os casamentos entre os vértices. Além disso, essa primeira função também possui uma variável “m” que irá indicar a quantidade de emparelhamento que se obteve. Já a segunda função “BFS” realiza uma busca em largura com mais de um ponto de origem, sendo que esses pontos de origem são adicionados em uma estrutura de fila “Q”. Nessa função o vetor “D” é modificado e o vértice artificial “D[0]” é utilizado para indicar se foi encontrado um caminho aumentante alternante. Por fim, a função “DFS” descreve uma busca em profundidade que é responsável por concretizar os casamentos entre os vértices, ou seja, é onde o vetor “mate” é alterado. As funções “BFS” e “DFS” retornam booleanos para indicar se a execução continua e se a quantidade de emparelhamento é incrementada, respectivamente.

Questão 3: A implementação do algoritmo de coloração de vértices é composta de três funções: Lawler, obterIndependentesMax e obterCombinacoesVertices. A função Lawler é a principal, ela recebe um grafo não dirigido G e executa um laço de repetição que procura pela coloração mínima possível para ele. Para isso, ela utiliza uma lista X de quantidade de cores para cada combinação de vértices, uma lista de dicionários C que informam a cor de cada vértice para cada combinação e uma lista S que contém todas as combinações possíveis de vértices de G, feita pela função obterCombinacoesVertices, que utiliza a função “combinations” do módulo “itertools” do python para obter todas as combinações possíveis de 0 até |V| elementos e salvar em uma lista S. A função Lawler, então, executa o laço criando um novo grafo não dirigido G1 e montando as arestas em comum com G. Para obter os conjuntos de vértices independentes maximais, existe a função obterIndependentesMax, que, a partir das combinações de vértices S, obtida com a função obterCombinacoesVertices, procura por um conjunto de vértices em S que seja independente e quando isso acontece, retorna uma lista I de conjuntos de vértices independentes que são de mesmo tamanho, ou seja, conjuntos independentes maximais. Depois de executado o algoritmo de Lawler, é impressa a quantidade de cores através do último elemento de X e a cor de cada vértice a partir do último elemento da llista de dicionários C.