Relatório Atividade 3 – INE5413

Bryan Martins Lima – 18100521
Arthur Moreira Rodrigues Alves – 17202296

1. [Edmonds-Karp]

Para a implementação de todos os algoritmos requisitados foi necessário uma adpatação em relação a leitura dos arquivos, ignorando comentarios e modificando o algoritmo de leitura. Os métodos originais sobre número de arestas, numero de vértices e outros já implementados nos outros dois trabalhos continuam os mesmos. Ainda é utilizado o método sys.maxsize para indicar valores infinitos. O algoritmo inicia criando um grafo ponderado dirigido usando o arquivo *teste1.gr* como base; após isso ele passa esse grafo, o indice de um vertice de origem e o indice de um vertice destino para a funcao ed_k, definida no arquivo edmonds_karp.py. Dentro dessa função é definida a função fluxo, cujo o dominio são todas os arcos que se pode formar com os vértices do grafo, e a sua imagem ínicialmente apenas o valor 0. Após definida, o algoritmo atualiza os fluxos de todos os arcos presentes nos caminhos minimos que são obtidos usando a função bfs, também definida no arquivo; o fluxo é atualizado usando a menor capacidade residual encontrada em um caminho. Essa função bfs foi modificada para adicionar ao caminho apenas os vertices cujo a capacidade residual era maior que 0. Após ter atualizado o fluxo para todos os caminhos que eram possíveis de se criar entre os vértices passados como paramentro para a função ed_k, a função retorna a soma dos fluxos.

2. [Hopcroft-Karp]

A primeira adaptação necessária para este algoritmo foi a utilização de um dicionário para a construção do vetor de distâncias para a utilização do indice None, visto que se fosse um vetor não seria possivel utilizar este método. O resto das estruturas e lógica é semelhante ao algoritmo dado em aula.

3. [Coloração de Vértices]

Foi ultilizado o algoritmo de Lawler para se obter o número minímo de cores necessárias para se colorir o grafo. O algorimo cria um grafo não ponderado e não dirigido usando os dados do arquivo *teste2.gr*, que é o mesmo arquivo usado na questão 1, mas alterado para representar o tipo de grafo necessário nessa questão. Após criado o grafo é passado como parâmetro para a função lawler. Essa função cria uma lista com o indice de todos os vertices do grafo, e partindo dela gera todas as possiveis combinações(subconjuntos) de vertices do grafo passado como parametro. Essa função não cria conjuntos repiditos, nos quais os elementos aparecem em ordens diferentes. Os subconjuntos são codificados em listas, que depois são ordenadas em ordem reversa(não existe razão especial em ordená-los em ordem reversa) para ajudar na identificação das mesmos posteriormente. Todas essa listas são armazenadas em uma única lista, chamada *subconjuntos*. Após criados, é criada uma lista que é indexada seguindo a mesma ordem na qual os subconjuntos são indexados em *subconjuntos* e preenchida com sys.maxsize; a função dessa lista é armazenar a quantidade de cores necessárias para se colorir cada subconjunto. É definida que a quantidade de cores necessárias para se colorir o conjunto vazio é 0. Após isso o algoritmo percorre o conjunto dos subconjuntos do grafo, criando um conjunto de subconjuntos indepedentes maximais para cada elemento; após criados o algoritmo realiza a diferença de conjuntos entre o subconjunto e cada um de seus subconjuntos maximais, verificando qual a quantidade de cores necessárias para se preencher o subconjunto obtido por essa diferença e atualizando o valor de cores necessários para se preencher o subconjunto original conforme necessário. Após verificar todas as quantidade de cores necessarias, o algoritmo retorna a quantidade de cores necessárias para se preencher o grafo em si, que é o último elemento da lista de subconjuntos.

Com o número minímo obtido, o algoritmo percorre o grafo colorindo os vértices com as cores que são identificadas por números que vão de 1 até pelo menos o N, sendo N o número retornado pelo algorimo de Lawler.