Começarei a explicação por cada classe criada e que é importante para o funcionamento do trabalho.
MeuGrafo: Essa classe que implementa a interface Grafo contém os principais objetos e métodos que serão utilizados nos algoritmos de grafos.
Os principais objetos são: Matriz de Adjacência (Que é uma matriz de ArrayList para poder suportar grafos ponderados e direcionados com arestas paralelas), Matriz de Incidência (matriz comum), Lista de Adjacência (ArrayList de ArrayList), vertices (ArrayList contendo os vértices do grafo) e arestas (ArrayList contendo as arestas do grafo).
Os métodos a seguir foram adaptados para funcionar de maneira diferente dependendo da estrutura escolhida pelo usuário: AdicionarAresta (adiciona a nova aresta na estrutura já escolhida), existeVertice (verifica a existência de um vértice no grafo), buscaVertice (busca e retorna determinado vértice do grafo), buscaAresta (busca e retorna determinada aresta do grafo), existeAresta (verifica a existência de uma determinada aresta), grauDoVertice (Retorna o grau de saída de um determinado vértice do grafo direcionado), numeroDeVertices (retorna o número de vértices do grafo), numeroDeArestas (retorna o número de arestas do grafo), adjacentesDe (método importantíssimo, retorna os vértices adjacentes de determinado vértice), setarPeso (permite mudar o peso de determinada aresta), vertices (retorna o ArrayList de vértices do grafo) e arestas (retorna o ArrayList de arestas do grafo).
As classes **Aresta** e **Vertice** sofreram alterações mínimas do código-base fornecido, sendo a principal alteração um novo atributo em Aresta: tipo, para definir se a aresta é de árvore, retorno, avanço ou cruzamento.

MeusAlgoritmosEmGrafos: Essa classe que implementa a interface AlgoritmosEmGrafos contém os principais objetos e métodos que serão exibidos ao usuário. Objetos: vetor de descoberta, vetor de finalização, vetor pai, valor do fluxo máximo.

Agora irei explicar os métodos:

Carregar Grafo: Método que recebe o caminho do arquivo de entrada e o Tipo de Representação e faz a leitura com o FileManager e monta o grafo adicionando os vértices e arestas nos devidos atributos/objetos e retorna um objeto MeuGrafo.

Busca em Profundidade: Método que recebe um objeto MeuGrafo e um objeto Vertice origem, o funcionamento desse método foi totalmente baseado no pseudo-código apresentado em aula, portanto ele utiliza um vetor de cor para controlar os vértices não descobertos (valor 0), descobertos (valor 1) e finalizados (valor 2), além disso ele registra os tempos em um vetor de descoberta e um vetor de finalização, registra também o pai de cada vértice no vetor pai. Foi implementado um método buscaEmProfundidadeVisit que assim como visto nos slides em aula é uma função recursiva que descobre todos os vértices e arestas do grafo a partir de um vértice de origem, além disso ela classifica as arestas descobertas: caso um vértice foi descoberto a aresta é classificada como de árvore, caso o vértice descoberto seja um ancestral do vértice atual é classificada como de retorno, caso o vértice descoberto já é um descendente do vértice atual é uma aresta de avanço, caso não se encaixe em nenhum dos casos anteriores ela é classificada como aresta de cruzamento. Por fim, o método retorna uma coleção das arestas descobertas pelo método.

Busca em Largura: Método que recebe um objeto MeuGrafo e um objeto Vertice origem, o funcionamento desse método foi totalmente baseado no pseudo-código apresentado em aula, ele registra os tempos de descoberta em um vetor de descobrimento e os pais de cada vértice em um vetor pai, usa uma fila para priorizar os vértices a serem visitados e descobertos.

Árvore Geradora Mínima: Recebe um objeto MeuGrafo, utiliza um vetor de ArrayList de vértices para controlar e saber quais os vértices já fazem parte da árvore geradora mínima, a princípio é feito uma ordenação em ordem crescente das arestas que compõem o grafo, como na teoria o corte a princípio passa por todas as arestas, então o algoritmo começa pelas arestas de menor peso, e vai adicionando os vértices que ainda não fazem parte da árvore geradora mínima aos conjuntos, caso de fato o vértice não fazia parte dos conjuntos a aresta que o ligou é adicionada ao conjunto das arestas que fazem parte da AGM, o processo se repete até que todos os vértices façam parte da AGM, após isso é retornado o conjunto de arestas que compõem a AGM.

Caminho Mínimo: Recebe um objeto MeuGrafo e dois objetos Vertice, origem e destino. Foi implementado o algoritmo de Dijkstra para caminho mínimo, ou seja foram implementados as funções de extrairMínimo e Relaxa, que funcionam de maneira idêntica ao pseudo-código apresentado em aula. Após aplicar o algoritmo de Dijkstra é recuperado o caminho mínimo através do vetor pai, do vetor pai é feito são recuperados os vértices pais do destino até a origem, possibilitando a recuperação das arestas , assim é retornado as arestas que compõem o caminho mínimo de dois vértices.

Fluxo Máximo: Recebe um objeto MeuGrafo e dois objetos Vertice, origem e destino. É feita uma cópia profunda do grafo original em um que funcionará como uma rede residual, é aplicado então uma busca em largura modificada que prioriza arestas de maior peso a partir do vértice de origem, depois é extraído o caminho (se houver) entre os dois vértices de maneira similar ao caminho mínimo, após o caminho ser extraído é subtraído o peso mínimo do caminho de todas as arestas que o compõem e esse peso mínimo é adicionado na variável fluxo máximo, depois repete-se o processo inteiro desde a busca em largura até não houver mais caminhos possíveis entre os dois vértices (nesse algoritmo conforme ocorre as subtrações dos pesos das arestas, é considerado que arestas de peso 0 são bloqueadas e não podem fazer parte do caminho), após isso é retornado o valor da variável de fluxo máximo.