Avaliação Prática - Relatório & Modelagem

3) **Daniel Terra + João Kienen - Java**:

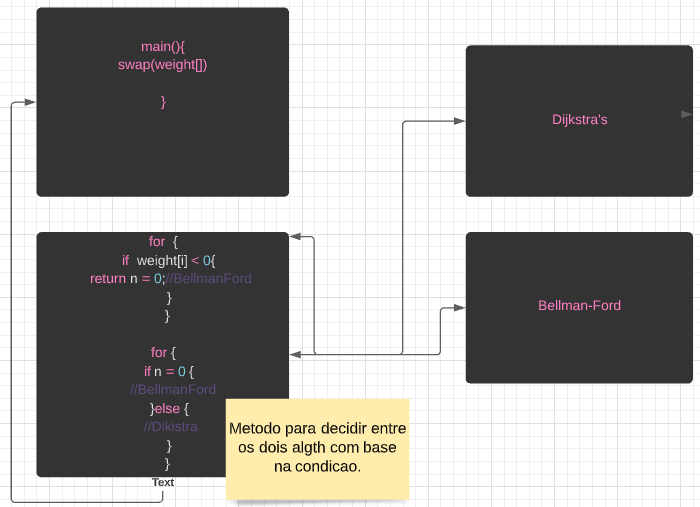
Dificuldade na implementação dos algoritmos. Sabendo que, teremos que usar todos os conhecimentos de POO em Java.

A dificuldade no avanço é trazer esses dois algoritmos implementados em Java e alternar seus usos baseado em um critério de seleção.

Planejamos criar uma Classe Dijkstra, Classe BFord, Classe Vértice e Classe Aresta.

Modelagem

A modelagem do problema irá partir do uso dos dois algoritmos de busca. Dependendo da quantidade de vértices e seus pesos de arestas. Sabendo que, o algoritmo de Dijkstra's não irá suportar números negativos. Será usado um critério de seleção entre os dois algoritmos, portanto, se houver arestas de pesos negativos iremos usar o algoritmo de Bellman-Ford para a resolução do problema. Por outro lado, o algoritmo de Bellman-Ford tem uma complexidade de 0(EV) limitando o tamanho do problema a ser tratado.



Algoritmos:

Dijkstra's: Algortimo para buscar o caminho de menor peso entre dois pontos. Em nosso problema irá suportar apenas números positivos de aresta.   
Complexidade de O(V \* log V + E), sendo V: vertices e E: arestas.

Bellman-Ford é um algoritmo de Single Source Shortest Path(SSSP). Fazendo com que encontre o menor caminho entre dois nos.  
Entretanto, não é ideal para a maioria dos problemas de (SSSP) devido a sua complexidade de 0(EV). Nesse caso, seria melhor usar o algoritmo de Dijkstra’s pois é mais rápido na ordem de 0((E+V) log(V)).