My graph is represented by a HashMap<String, HashMap<String, PriorityQueue<String>>>. The outer keys represent the nodes of the graph. The inner keys represent the child nodes of edges travelling from the outer key to inner key. The Priority Queues hold the labels associated with these edges. HashMaps allow for quick node and edge lookups and insertions due to their O(1) average lookup and insertion complexities . In addition, using a PriorityQueue to store labels allows for O(1) time complexity to find the minimum edge label between two nodes, which is a common graph operation.

The advantage of a collection of edges is that you don't have to deal with having node objects as well as associated edge objects.

The advantage of an adjacency list is that unlike an adjacency matrix, in order to find the children of a node you don't have to iterate through all nodes of the graph.

The advantage of an adjacency matrix is that you can find an edge an its label in O(1) time since you only have to access an index of an array.

I chose my representation since it offers good performance compared to other options and lends itself well to the typical functions of a graph, including node lookup/insertion, edge lookup/insertion, and finding the minimum edge between 2 nodes.