To implement BFS, we needed to add two helper functions. Since the Edge class was not hashable, we decided to convert edges to string in a way that the string could easily be read out and converted back into the edge that it represents. This allowed us to use an unordered map to store the label of each edge as well. The rest of the BFS structure follows the general pseudo-code that we were shown in class. At the end, we return a graph that only contains the discovery edges, or an MST of our dataset.

To implement Dijkstra's algorithm, we had to make some changes to the data structures we were familiar with. First, the vertices in the graph needed additional metadata, so we created a wrapper class to store it. We also needed a priority queue that could process our Dijkstra vertices, so we modified the heap class from lab_heaps. Instead of dealing with generic types, it is specialized to work with Dijkstra vertices. We also defined a custom priority function that favors Dijkstra vertices with lower distance values. Finally, the Dijkstra heap stores a map that points from each vertex to its corresponding Dijkstra vertex to allow for easy lookups.

For the landmark path algorithm, the function was a lot easier to implement than we initially expected. Since we're dealing with directed graphs, we found that the algorithm could be implemented with simply two calls to Dijkstra's algorithm: one from the starting vertex to the landmark, and another from the landmark to the ending vertex.

Another one of the algorithms we implemented was to find what new edges to insert into the graph to make it one strongly connected component such that we insert the minimum number of edges required. We used Kosaraju's algorithm to find the individual strongly connected components in the graph and identify representative vertices in the graph for each strongly connected component. We then iterated through the edges of the original graph to find out what strongly connected components were connected to each other in the graph, and used disjoint sets

to quickly identify this relation between SCCs in the graph. While iterating, we also found the SCCs of indegree zero and outdegree zero, because the minimum number of edges we needed to add was the maximum of the number of SCCs of indegree zero and the number of SCCs of outdegree zero. To only insert the minimum number of edges, we prioritized creating new edges that were outbound from an SCC of outdegree zero and inbound to an SCC of indegree zero such that the new edge would not create a cycle in the graph. If avoiding creating a cycle was impossible, our new edge was outbound from an SCC of outdegree zero and inbound to a different SCC of indegree zero. As a last resort, if there were any SCC vertices of indegree zero or SCC vertices of outdegree zero left, we just created a new edge connecting the SCC vertex to or from a different arbitrary vertex in the original graph.

The last algorithm we implemented was graph coloring. To do, we used an undirected and unweighted version of the original graph. We looped through all the vertices to fill the lowest possible color of a vertex so that its adjacent vertices don't contain the same color. The function returns a 2D vector with the indices as colors and values as a vector of adjacent vertices. We used a greedy algorithm to accomplish this. While this process isn't always the most efficient way, it does guarantee an upper bound of d + 1 on the number of colors generated where d is the degree of the graph.

Here is an example output from "playing" the game (The actual list of vertices is very long, so it's a slightly simplified version of the game).

```
sandpaper567@DESKTOP-HT4SRIL:~/desktop/Coding Stuffs/C++/PracticeProblems/asd6-adarshs3-sahaj2-vajrala3$ make && ./main
Makefile:27: -e  Looks like you are not on EWS. Be sure to test on EWS before the deadline. -e
clang++ main.cpp extractData.o dsets.o algorithms.o graph.o DijkstraVertex.o DijkstraHeap.o -std=c++1y -stdlib=libc++ -lc++abi -lm -o main

Num Edges Needed to Add to Play Reddi-spedia: 4205

Welcome to Reddi-Spedia. To begin, please select a starting location from the following list of subreddits:
lifeasannpc | girlsgonedopecoin | hommlol | gratefuldead | truechristian
Type your choice here (case sensitive): hommlol

Next choose a random destination subreddit:
ea_nhl | aquariums | sgsappeals | nordiccountries | luxembourg
Type your choice here (case sensitive): nordiccountries

The shortest path from r/hommlol to r/nordiccountries is the following path:
hommlol->leagueoflegends->python->learnprogramming->nocontract->aiowireless->abdl->abdlstories->mcmaster->queensuniversity->respiratorytherapy->iceland->nordiccountries
```