Final Project Report

1. Dataset:
    a. https://www.kaggle.com/datasets/nathanlauga/nba-games

2. Introduction:
    a. For my final project, I used the “games.csv” file, which includes all games from 2004 season to last updated date teams and some details like number of points, etc. Given the widespread interest in basketball and the NBA, this dataset presents a valuable opportunity to delve into in-depth analyses of player strategies, team dynamics, and the factors influencing success in professional basketball. By utilizing this dataset, one can conduct detailed player performance evaluations, identify patterns in individual and team statistics, and assess the impact of various performance metrics on overall team success.

3. Project Goal:
    a. The overall objective of the provided code is to analyze and visualize the betweenness centrality of nodes in a directed graph that represents game outcomes. The code achieves this objective through the following steps:
        i. Data Loading and Graph Construction (construct.rs): The code reads game data from a CSV file (games.csv) using the read_data function and constructs a directed graph based on the outcomes of the games. Teams are represented as nodes, and game results (wins) are represented as weighted directed edges in the graph.
        ii. Centrality Computation (function.rs): The code calculates the betweenness centrality for each node in the constructed graph using Dijkstra's algorithm. Betweenness centrality is a measure of a node's importance in facilitating communication between other nodes in the graph.
        iii. Sorting and Output (main.rs): After calculating centrality scores, the code sorts the nodes in descending order based on their centrality scores. It then prints the nodes along with their corresponding betweenness centrality scores. This provides a ranking of nodes based on their importance in the context of game outcomes.
        iv. Testing (test_sum function): The code includes a test function (test_sum) to ensure the correctness of a separate function (calculate_sum). This function checks whether the sum of a vector matches the expected sum, serving as a basic unit test.

v. In summary, the code's objective is to analyze the centrality of teams in the context of game outcomes, providing insights into which teams play a crucial role in the overall connectivity of the game results network. The modularity of the code allows for easy extension and maintenance, and the use of Rust and the petgraph library facilitates efficient graph manipulation and analysis.

4. Methodology
    a. Function.rs
        i. This module contains functions related to computing betweenness centrality in a directed graph. It employs Dijkstra's algorithm to calculate the shortest paths and then uses these paths to determine the betweenness centrality of each node. The code includes functions to find incoming and outgoing neighbors, as well as the core function betweenness_centrality that computes centrality scores.
    b. Construct.rs
        i. This module defines a Game structure representing game outcomes and provides functions for constructing a directed graph from a dataset stored in a CSV file. The graph is constructed based on game outcomes, with teams as nodes and the game results as weighted directed edges. The code includes functions to read data from a CSV file, validate game outcomes, and build the directed graph.
    c. Main Program
        i. The main program is the entry point of the application. It utilizes the functionality provided by the centrality.rs and construct.rs modules to analyze and print the betweenness centrality scores of nodes in the constructed graph. The program reads game data from a CSV file, constructs a directed graph, calculates betweenness centrality for each node, and then prints the nodes along with their centrality scores in descending order.
        ii. Additionally, the code includes a test function (test_sum) to verify the correctness of a separate function (calculate_sum). This test function checks whether the sum of a vector matches the expected sum.
        iii. In summary, the provided code is a Rust program that utilizes the petgraph library to analyze and visualize the betweenness centrality of nodes in a directed graph representing game outcomes. The modularity and separation of concerns make the code organized and maintainable.

5. Results and Observations
    a.

```
    Finished dev [unoptimized + debuginfo] target(s) in 0.10s
     Running `target/debug/DS210_FinalProject`
Node 1610612747: Centrality 1128.5998511943342
Node 1610612746: Centrality 961.2502162097703
Node 1610612750: Centrality 961.2168934217232
Node 1610612762: Centrality 851.6484181933937
Node 1610612754: Centrality 686.0769200625218
Node 1610612759: Centrality 674.7693890528208
Node 1610612755: Centrality 606.7344061375363
Node 1610612753: Centrality 596.5907425495917
Node 1610612765: Centrality 526.9937941716788
Node 1610612742: Centrality 426.72505583802734
Node 1610612745: Centrality 421.3700498333922
Node 1610612737: Centrality 409.9341435268572
Node 1610612758: Centrality 405.30339386481705
Node 1610612756: Centrality 396.7839167894836
Node 1610612743: Centrality 393.7028479605096
Node 1610612738: Centrality 363.44108851314405
Node 1610612740: Centrality 353.2570462830489
Node 1610612764: Centrality 338.7260221464936
Node 1610612752: Centrality 323.81225856140634
Node 1610612739: Centrality 311.0707392603048
Node 1610612751: Centrality 309.01598137384326
Node 1610612766: Centrality 304.3613274867801
Node 1610612749: Centrality 219.76097870247355
Node 1610612748: Centrality 218.46442410150425
Node 1610612741: Centrality 175.32840424634568
Node 1610612763: Centrality 143.92827177539883
Node 1610612757: Centrality 91.13464223652498
Node 1610612761: Centrality 88.8706905850262
Node 1610612760: Centrality 82.54200288223609
Node 1610612744: Centrality 2.2685787295150845
```

    i. The result represents the betweenness centrality scores for each node (team) in the directed graph constructed from the game outcomes.
        1. Node 1610612747: Centrality 1128.5998511943342
        2. Explanation: This team has the highest betweenness centrality in the graph.
        3. Example: Team 1610612747 plays a crucial role in connecting other teams in the game outcomes. It is often on the shortest paths between other teams.

    ii. Teams with higher centrality scores are more influential in connecting other teams in the network of game outcomes. They are often on the shortest paths between pairs of teams, indicating their importance in determining the overall flow of game results. Conversely, teams with lower centrality scores have less impact on the overall connectivity of the game outcomes network.

6. Test Function

```
    Finished test [unoptimized + debuginfo] target(s) in 0.08s
     Running unittests src/main.rs (target/debug/deps/DS210_FinalProject-760505c
172273744)

running 1 test
test test_centrality ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; fini
shed in 0.00s
```

7. Conclusion
    a. In conclusion, the project effectively employs graph analysis to reveal the betweenness centrality of teams in a sports league. The results highlight key teams that play pivotal roles in shaping the overall connectivity of game outcomes. The modular and well-tested code provides a solid foundation for similar analyses in different contexts, showcasing the power of Rust and the petgraph library in efficient graph-based computations.

8. Resources
    a. https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/
    b. https://depth-first.com/articles/2020/02/03/graphs-in-rust-an-introduction-to-petgraph/
    c. Lecture Notes.
    d. Chat GPT
        i. When encountering errors that seem unsolvable, I occasionally resort to writing them down as a way to find a solution.
            1. Ex). The test function is keep encountering erros. What is one way to solve this problem?