This Rust program analyzes the degree distribution of a graph constructed from a dataset of edges (node pairs). It also evaluates how well the graph's degree distribution fits a theoretical power-law model which is often observed in real-world networks. This database deepens my understanding of network structure, in particular how real-world networks exhibit specific properties such as degree distributions and power-law behavior. This analysis can reveal valuable questions about how networks are formed, how information or resources flow through them, and how to optimize or protect the network structure. In social networks, understanding the degree distribution can reveal influential individuals or “hubs”.

The Graph struct represents the graph using an adjacency list. The adjacency_list is a HashMap where each key is a node, and the corresponding value is a HashSet of its neighbors. Then use add_edge to add connections, degree_distribution to compute node degrees, and neighbors_at_distance_two to find nodes two hops away. build_graph_from_csv reads a CSV file where each line represents an edge between two nodes. It splits the line into two parts, trims the values, and adds the edge to the graph using the add_edge method. The function can help to ensure the graph structure is populated correctly and efficiently. This step converts raw data into a usable graph representation. degree_distribution calculates the number of connections for each node in the graph by iterating over the adjacency list. It can gather the number of nodes with the same degree into a HashMap, where keys are degrees, and values are counts. evaluate_power_law compares the observed degree distribution to a theoretical power-law distribution. It normalizes the degree counts into probabilities and calculates the theoretical probabilities. This determines if the network has a power-law structure.

The main function builds the graph from a dataset using build_graph_from_csv, computes the degree distribution with degree_distribution, and evaluates the power-law fit using evaluate_power_law. The results are printed in a user-friendly way, with explanations for both degree distribution and power-law fit, offering insights into the network's structure. Use tests to verify graph construction, degree computation and neighbor calculations. Empty graphs and fully connected graphs are also tested.