Greedy Project

Samuel Escalera Herrera

1. Implement the necessary code to solve the problem: https://github.com/SamuelEscalera/Algoritmia2_SamuelEscalera/tree/main/src/main/java/org/example/AudiesParty
    a. Make sure that you apply greedy algorithms to your solution:

       In my code, I am using the greedy algorithm in the findMaximalGuestList(double x) method of my GuestList class. Here is the implementation:

        1. In each iteration of the for loop, I start from a vertex in my graph and explore all its neighbors to form a list of guests that meet the given conditions.
        2. Within visitNeighbors, when I explore the neighbors of a given vertex, I choose those neighbors whose friendship weight is greater than my x threshold.
        3. Then, in calculateSum, I compute the sum of the friendship weights among the guests in the given list.
        4. At the end of each iteration, I compare the current list of guests with the maximum list up to that point, and if the current list is larger or has a larger sum of friendship weights, I update the maximum list with the current list.

       This approach is greedy because at each step, I locally choose the best available option (the neighbors that meet the condition) without considering the big picture of the graph.
    b. Do not forget to write clean code and follow best practices:

        - **Descriptive variable and method names**: I have used descriptive names for my variables and methods, which makes my code easy to understand.
        - **Documentation of methods and classes**: I have included Javadoc comments that explain the purpose of my classes and methods.
        - **Use of interfaces instead of specific implementations**: I have used the Graph interface instead of a specific implementation in my GraphCreated class. This is a good practice as it makes my code more flexible and allows to easily interchange different graph implementations in the future.
        - **Code organization:** I have divided my code into well-defined classes and methods, which makes it easier to understand and maintain.
2. Write a brief explanation of why you have chosen the greedy algorithm to solve the problem:
   I chose to use the greedy algorithm to solve the problem because of its simplicity and efficiency in this specific context. Here are the reasons behind my choice:

- Simplicity of implementation: In my case, I can iteratively build my maximal guest list by exploring neighbors that meet the friendship criteria and adding them to the list until there are no more neighbors that meet the criteria.
- Close to optimal solution: Although the greedy algorithm does not guarantee to find the optimal solution in all cases, in many situations it provides a solution that is close enough to the optimum.

3. Identify the time complexity of your solution:

   My algorithm has a linear O(n) complexity.

   In the worst case, where the complexity would be O(n2) quadratic and would be where each vertex is connected to all the others, however this case is not very common so the complexity in general would be O(n) linear.