```
import tkinter as tk
from tkinter import simpledialog, messagebox
import ttkbootstrap as ttk # type: ignore
import random
import math
from collections import deque

class Graph:
    def _init_(self):
        self.adjacency = {}

    def add_user(self, user):
        if user not in self.adjacency:
            self.adjacency[user] = set()

    def add_friendship(self, user1, user2):
        if user1 in self.adjacency and user2 in self.adjacency:
            self.adjacency[user1].add(user2)
            self.adjacency[user2].add(user1)

    def get_friends(self, user):
        return list(self.adjacency.get(user, []))

    def bfs_suggestions(self, user):
        visited = set()
        queue = deque()
```

```
        queue.append((user, 0))
        suggestions = set()

        while queue:
            current, level = queue.popleft()
            if level > 2:
                continue
            visited.add(current)
            if level == 2:
                suggestions.add(current)
            for neighbor in self.adjacency.get(current, []):
                if neighbor not in visited:
                    queue.append((neighbor, level + 1))

        # Excluir amigos directos y el propio usuario
        direct_friends = set(self.get_friends(user))
        suggestions -= direct_friends
        suggestions.discard(user)
        return list(suggestions)

class SocialNetworkApp:
    def _init_(self, root):
        self.root = root
        self.root.title("Red Social - Grafo de Usuarios")
        self.graph = Graph()
        self.positions = {}  # Para la visualización
```

```
        self.selected_user = None

        self.setup_ui()

    def setup_ui(self):
        # Estilos
        style = ttk.Style()
        style.theme_use('superhero')

        frame = ttk.Frame(self.root, padding=10)
        frame.pack(side="left", fill="y")

        self.canvas = tk.Canvas(self.root, bg="white", width=600, height=600)
        self.canvas.pack(side="right", fill="both", expand=True)

        ttk.Button(frame, text="Agregar Usuario", command=self.add_user).pack(fill="x", pady=5)
        ttk.Button(frame, text="Crear Amistad", command=self.create_friendship).pack(fill="x", pady=5)
        ttk.Button(frame, text="Ver Amigos", command=self.show_friends).pack(fill="x", pady=5)
        ttk.Button(frame, text="Sugerir Amigos", command=self.suggest_friends).pack(fill="x", pady=5)

        self.results_text = tk.Text(frame, height=10, wrap="word")
        self.results_text.pack(fill="both", pady=10)
```

```
        self.canvas.bind("<Button-1>", self.select_user)

    def add_user(self):
        name = simpledialog.askstring("Agregar Usuario", "Nombre del nuevo usuario:")
        if name:
            self.graph.add_user(name)
            self.positions[name] = (random.randint(50, 550), random.randint(50, 550))
            self.draw_graph()

    def create_friendship(self):
        users = list(self.graph.adjacency.keys())
        if len(users) < 2:
            messagebox.showinfo("Información", "Necesitas al menos 2 usuarios.")
            return

        user1 = simpledialog.askstring("Usuario 1", "Nombre del primer usuario:")
        user2 = simpledialog.askstring("Usuario 2", "Nombre del segundo usuario:")

        if user1 and user2:
            if user1 == user2:
                messagebox.showerror("Error", "No puedes conectar un usuario consigo
mismo.")
                return
            self.graph.add_friendship(user1, user2)
            self.draw_graph()
```

```
    def show_friends(self):
        if not self.selected_user:
            messagebox.showinfo("Información", "Selecciona un usuario en el grafo.")
            return
        friends = self.graph.get_friends(self.selected_user)
        self.results_text.delete(1.0, tk.END)
        self.results_text.insert(tk.END, f"Amigos de {self.selected_user}:\n" +
"\n".join(friends))

    def suggest_friends(self):
        if not self.selected_user:
            messagebox.showinfo("Información", "Selecciona un usuario en el grafo.")
            return
        suggestions = self.graph.bfs_suggestions(self.selected_user)
        self.results_text.delete(1.0, tk.END)
        self.results_text.insert(tk.END, f"Sugerencias para {self.selected_user}:\n" +
"\n".join(suggestions))

    def draw_graph(self):
        self.canvas.delete("all")
        # Dibujar aristas
        for user, friends in self.graph.adjacency.items():
            x1, y1 = self.positions[user]
            for friend in friends:
                if user < friend:  # evitar duplicar líneas
                    x2, y2 = self.positions[friend]
                    self.canvas.create_line(x1, y1, x2, y2, fill="gray")
```

```
        # Dibujar nodos
        for user, (x, y) in self.positions.items():
            color = "green" if user == self.selected_user else "blue"
            self.canvas.create_oval(x-20, y-20, x+20, y+20, fill=color)
            self.canvas.create_text(x, y, text=user, fill="white", font=("Arial", 10, "bold"))

    def select_user(self, event):
        for user, (x, y) in self.positions.items():
            if math.hypot(event.x - x, event.y - y) <= 20:
                self.selected_user = user
                self.draw_graph()
                break

if _name_ == "_main_":
    root = ttk.Window(themename="superhero")
    app = SocialNetworkApp(root)
    root.mainloop()
```