Relatório Acadêmico – Projeto de Ciência de Redes

Estrutura de Dados

COMPONENTE CURRICULAR: Estrutura de Dados

PROFESSOR: Cleyton Rodrigues

CURSO: Engenharia da Computação

ALUNO: Lucas Ribeiro Costa

Projeto de Ciência de Redes

1. Objetivo

Este projeto teve como finalidade aplicar os conceitos de grafos no contexto da Ciência das Redes. Para isso, foi construído um grafo de interações entre os personagens de *Dragon Ball Z*, segmentado por sagas. A proposta consistiu em representar os níveis de relacionamento entre os personagens — aliados ou inimigos — ao longo de cada arco do anime, analisando essas conexões por meio de estruturas de dados, em especial utilizando grafos e seus algoritmos clássicos.

2. Justificativa e Modelagem

- A base de dados principal está no arquivo `relacoes.csv`, contendo as colunas: origem, destino, tipo, saga e peso — representando o nível da relação (aliança ou inimizade).
- A segunda base, `personagens.csv`, foi criada para mapear cada nó à sua respectiva imagem `.png`.
- Nós: personagens mais relevantes de cada saga.
- Arestas: conexões de aliança ou inimizade entre os personagens.
- Pesos: intensidade de cada interação.
- Os grafos foram construídos separadamente para cada saga.

A escolha por esse universo temático se justifica pelo fato de que, ao longo do anime, Goku e seus aliados constroem diversas relações — de amizade e rivalidade — que se desenvolvem conforme a narrativa avança. A vasta gama de personagens e a complexidade de suas conexões serviram como excelente inspiração para representar essas interações por meio de grafos.

3. Estrutura e Dados de Implementação

- Foi utilizado as bibliotecas NetworkX para melhor manipulação de redes.
- O Grafo foi criado por uma lista de adjacência.
- Os arquivos CSV contém os nós e arestas.
- Trecho da lógica implementada para criar o grafo via Networkx:

```python
def construir_grafo_por_saga(relacoes, saga_nome=None):
    G = nx.Graph()
    for r in relacoes:
        if saga_nome is None or r["saga"].lower() == saga_nome.lower():
            G.add_node(r["origem"])
            G.add_node(r["destino"])
            G.add_edge(
                r["origem"],
                r["destino"],
                tipo=r["tipo"],
                peso=r["peso"]
            )
    return G
```

4. Algoritmo e Métricas Aplicadas

- Algoritmos utilizados no projeto:

- **Busca em Largura (BFS):** foi construida para identificar quais personagens estão mais próximos do Goku, destacando a interação direta no grafo.
- **Busca em Profundidade (DFS):** percorre o grafo explorando ao máximo cada ligação antes de realizar o caminho do retorno, Criada para mostrar até onde vai as conexões do Goku dentro de cada saga.
- **Algoritmo de Dijkstra (entre aliados):** criado para atender somente aos nós do tipo “aliado”, com Goku sendo a origem, Evidênciando no script o peso de cada nó, de modo inversamente proporcional, quanto mais forte a ligação, será menor o custo. Permitindo encontrar caminhos mais “fortes” em níveis de confiança com outros nós.

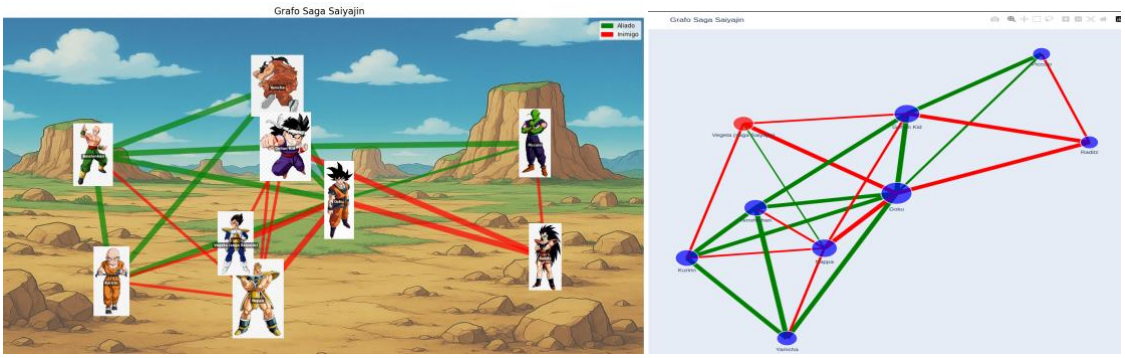
```python
def dijkstra_aliados(G, origem):
    G_aliados = nx.Graph()
    for u, v, data in G.edges(data=True):
        if data['tipo'] == 'aliado':
            custo = 1 / data['peso'] if data['peso'] > 0 else 10
            G_aliados.add_edge(u, v, weight=custo)

    if origem not in G_aliados:
        print(f"Origem '{origem}' não está conectada por alianças nesta saga.")
        return

    caminhos = nx.single_source_dijkstra_path(G_aliados, origem)
    distancias = nx.single_source_dijkstra_path_length(G_aliados, origem)

    print(f"\nCaminhos de aliados mais fortes a partir de {origem}:")
    for destino in caminhos:
        if destino != origem:
            caminho = " -> ".join(caminhos[destino])
            print(f"  {destino}: {caminho} (Custo: {distancias[destino]:.2f})")

def dfs(G, origem):
    visitados = list(nx.dfs_preorder_nodes(G, source=origem))
    print(f"\nBusca em profundidade (DFS) a partir de {origem}:")
    print(" -> ".join(visitados))

def bfs(G, origem):
    visitados = list(nx.bfs_tree(G, source=origem))
    print(f"\nBusca em largura (BFS) a partir de {origem}:")
    print(" -> ".join(visitados))
```

Além disso, pode perceber que os algoritmos de busca foram desenvolvidos para percorrer o grafo gerado pela estrutura do nx.Graph(), a biblioteca do Networkx.

5. Visualizações

-Para facilitar a visualização das redes foi utilizado duas bibliotecas:

- Visualização com Imagens(Matplotlib) e Interativa(Plotly):



Na primeira figura, os personagens foram representados por figuras em .PNG e, como pode perceber na imagem, foram posicionados conforme o layout do Matplotlib. Na segunda imagem, uma segunda versão foi criada em HTML, somente para fins de melhor interação sobre o grafo, proporcionando melhor visualização e entendimento sobre os pesos, qual nó possui mais ligações, quem é o nó central ou o último nó a ser percorrido, marcado em vermelho, sendo classificado como o oponente principal a ser combatido pelo nó principal (Goku).

Conforme solicitado, segue o link do GitHub do projeto: https://github.com/Lucasrc22/Projeto-ED-DBZ.git

6. Análise Crítica e Aprendizados

Ao realizar o desenvolvimento deste projeto, foi possível observar a importância de existir um nó central, a partir do qual, ao percorrê-lo, se pode chegar a um resultado que faça sentido, que, neste caso, foram as ligações de aliança e inimizade. Com isso, foi possível obter um melhor entendimento dos princípios de redes com grafos.