Relatório de Projeto: Análise de Redes em Labirintos Gerados Aleatoriamente

Autor: Arthur Pontes Ferreira **Repositório:** https://github.com/Arthurpf1448/Projeto_ED.git

1. Introdução e Definição da Rede

Este projeto aplica os conceitos da Teoria dos Grafos e Ciência das Redes para modelar e analisar um labirinto bidimensional gerado aleatoriamente.

A rede foi modelada da seguinte forma:

- **Nós (Vértices):** Cada célula `(i, j)` do grid do labirinto é um nó. As células podem ser de dois tipos: caminhos (`0`) ou paredes (`1`).
- **Arestas (Conexões):** Uma aresta não-direcionada e sem peso conecta dois nós se as células correspondentes são adjacentes (vertical ou horizontalmente) e **ambas são caminhos (valor 0)**.

2. Criatividade e Modelagem dos Dados

A criatividade do projeto está na geração procedural de um novo labirinto a cada execução, criando um problema dinâmico para análise. Para a modelagem, o labirinto de é tamanho N é mapeado para um grafo de N^2 vértices, onde cada célula (i, j) é convertida em um índice pela fórmula: índice = i * N + j.

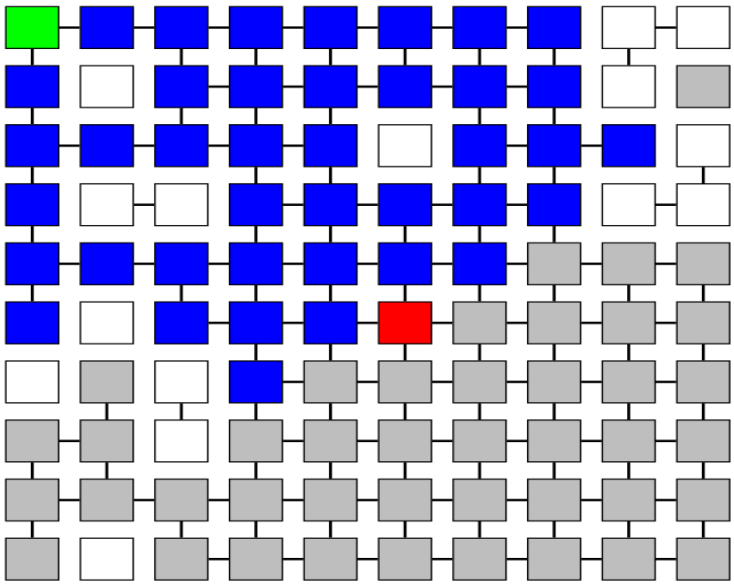
A estrutura de dados escolhida foi a **Matriz de Adjacência**, implementada na classe `Grafo` através de um `vector<vector<int>>`, onde `matriz_adjacencia[i][j] = 1` representa a existência de uma aresta.

Trecho de Código Corrigido (Modelagem):

```
void gerar_grafo(Grafo& g, const vector<vector<int>>& lab) {
    int lab_tam = lab.size();
    auto Get = [&](int i, int j) { return j + i * lab_tam; };
    // Adiciona as arestas baseadas na adjacência de células de caminho
    for (int i = 0; i < lab_tam; i++) {
        for (int j = 0; j < lab_tam; j++) {
            // Conecta com o vizinho de baixo se ambos forem caminhos
            if (i + 1 < lab_tam && lab[i + 1][j] == 0) {
                g.add_aresta(Get(i, j), Get(i + 1, j));
            }
            // Conecta com o vizinho da direita se ambos forem caminhos
            if (j + 1 < lab_tam && lab[i][j + 1] == 0) {
                g.add_aresta(Get(i, j), Get(i, j + 1));
```

```
        }
      }
    }
}
```

3. Visualização da Rede



A visualização da rede é realizada com o **Graphviz**, a partir de um arquivo `.dot` gerado pelo programa. As cores são usadas de forma estratégica para facilitar a interpretação da estrutura e da solução encontrada:

- **Nó de Partida:** Verde
- **Nó de Destino:** Vermelho
- **Caminho Encontrado:** Azul
- **Paredes:** Branco
- **Caminhos Livres:** Cinza

Essa codificação visual permite uma análise rápida e clara do labirinto e do resultado dos algoritmos de busca.

4. Algoritmos e Métricas Aplicadas

Dois algoritmos clássicos de busca foram implementados para verificar a conectividade entre uma célula de origem e uma de destino:

1. **Busca em Largura (BFS):** Explora o grafo em camadas, sendo ideal para encontrar o **caminho mais curto** em grafos não ponderados como o do projeto.

2. **Busca em Profundidade (DFS):** Explora cada ramo do grafo até o fim antes de retroceder. É eficiente para verificar a existência de um caminho, mas não garante que seja o mais curto.

Análise de Métricas da Rede:

- **Grau dos Nós:** Representa o número de movimentos possíveis a partir de uma célula. **Interpretação:** Nós com grau 1 são becos sem saída; grau 2 formam corredores; e grau 3 ou 4 são interseções ou áreas abertas, funcionando como pontos de distribuição de caminhos.

- **Componentes Conectados:** É um subconjunto de nós onde todos são alcançáveis entre si. **Interpretação:** Para que um labirinto tenha solução entre dois pontos, ambos devem pertencer ao mesmo componente conectado. Ou seja, deve haver um caminho entre eles.

5. Análise Crítica e Reflexão Final

Este projeto demonstrou com sucesso a eficácia da Teoria dos Grafos na resolução de problemas espaciais. Os principais aprendizados foram:

1. **Poder de Abstração:** A capacidade de traduzir um problema concreto como um labirinto em uma estrutura abstrata de grafo é uma ferramenta poderosa para a solução de problemas.

2. **Significado das Métricas:** Métricas como o grau dos nós revelam a importância estrutural de cada célula, distinguindo corredores de cruzamentos críticos sem a necessidade de simulações complexas.

3. **Visualização como Ferramenta de Análise:** A visualização gráfica não é apenas uma ilustração, mas uma ferramenta analítica que valida os resultados e facilita a compreensão da topologia da rede.

4. **Escolha da Estrutura de Dados:** A matriz de adjacência foi usada por sua simplicidade. Contudo, para labirintos muito maiores, uma **lista de adjacência** seria mais eficiente em memória, com complexidade de espaço O(V+E) em vez de O(V2).

Conclui-se que o projeto atingiu seus objetivos, reforçando a importância dos grafos como uma linguagem universal para descrever e analisar sistemas conectados.