Gerador de Labirintos

Lucas Hochleitner da Silva
hochlucassilva@gmail.com
@lucashoch

1.

O Objeto e o Problema

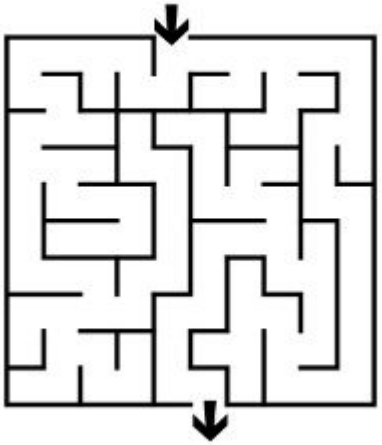
Um Labirinto?

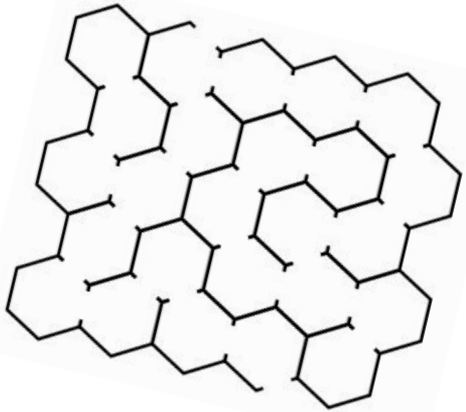
Labirinto

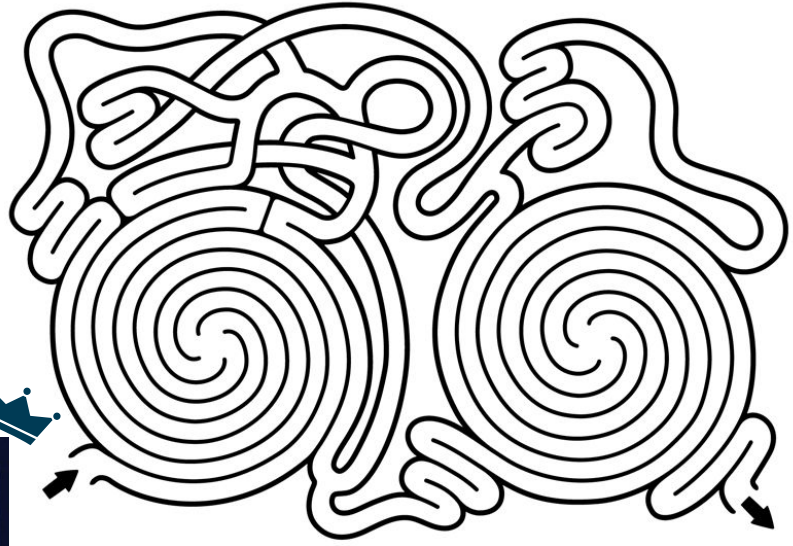
Um labirinto é constituído por um conjunto de percursos intrincados criados com a intenção de desorientar quem os percorre.

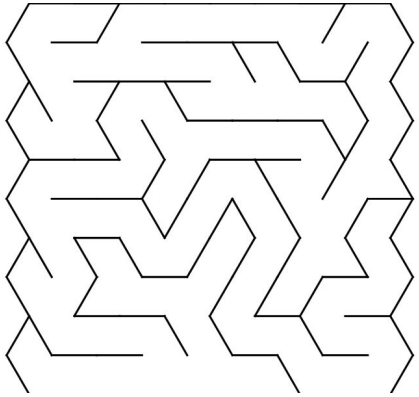
Várias classificações:

- Caminhos
- Planaridade
- Dimensão





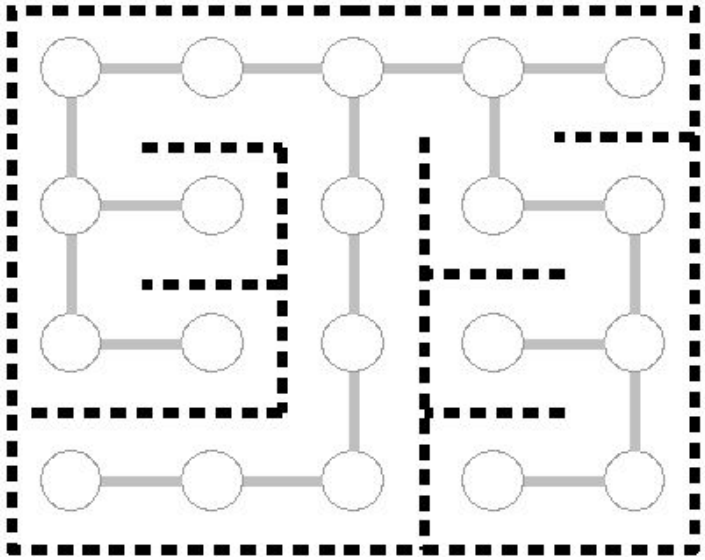


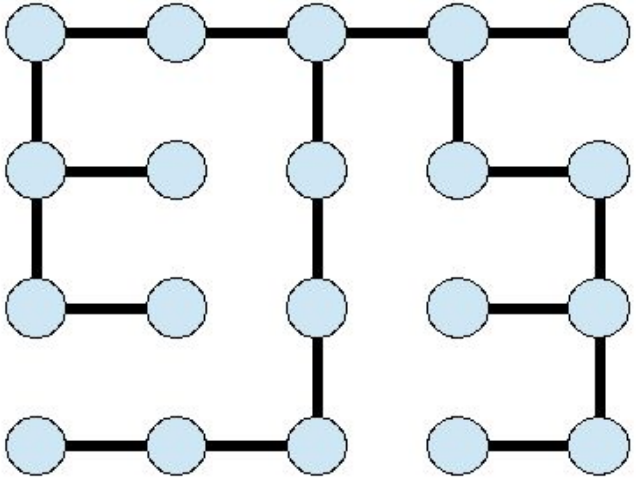


Labirintos Perfeitos

Labirinto Perfeito

Labirinto → Grafo → Árvore





2.
As Soluções

Criando Problemas

Abordagens Selecionadas

1. Busca Randômica em Profundidade
2. Algoritmo Randômico de Prim
3. Divisão e Conquista
    a. Constante
    b. Variável

Busca Randômica em Profundidade

- Recursive Backtracker
- Pilha de tamanho máximo N x M
- Percorre sempre toda célula duas vezes

Algoritmo

Inicialize a célula *atual* e marque como visitada

Enquanto existirem células que não foram visitadas:

- Se a célula *atual* tiver vizinhos não visitados:
  - Escolha aleatoriamente um dos vizinhos não visitados
  - Coloque a célula *atual* na pilha
  - Remova a parede entre a célula *atual* e a célula escolhida
  - A célula *atual* recebe a escolhida e marca como visitada
- Senão, se a pilha não estiver vazia:
  - Retire o topo da pilha e torne ele a célula *atual*

Busca Randômica em Profundidade

- Pode causar Stack Overflow dependendo do labirinto
- Resulta em labirintos com um fator de ramificação baixo (menos dead-ends), mas mais longos. Geralmente apresenta uma solução longa e bastante tortuosa.

Algoritmo Randômico de Prim

Modificação do algoritmo de Prim

- Arestas sempre com mesmo peso

Lista de tamanho máximo ⅔ N x M

Algoritmo

Escolha uma célula inicial e marque como visitada

Adicione os vizinhos da célula inicial na lista de células

Enquanto existirem células na lista:

- Escolha aleatoriamente uma célula *x* da lista
- Escolha aleatoriamente um de seus vizinhos visitados
- Remova a parede entre a célula *x* e o vizinho escolhido
- Adicione os vizinhos não visitados de *x* na lista de células

Algoritmo Randômico de Prim

- Também usa armazenamento proporcional ao tamanho do labirinto
- Resulta em labirintos com um fator de ramificação muito alto (muitas dead-ends), e muito curtas. Geralmente apresenta uma solução relativamente direta.

Divisão e Conquista

Divisão Recursiva

Focado em paredes, não em células

Não usa estrutura de armazenamento

Algoritmo mais rápido sem *bias* direcional

Fator constante ou aleatório

Algoritmo

Escolha um sentido (vertical/horizontal)

Escolha uma posição naquela direção (definida pelo fator)

Preencha toda a direção com paredes

Na parede criada, crie uma abertura numa posição aleatória

Repita o processo para o lado esquerdo e direito da parede criada

Condição de parada: Quando o tamanho da porção é igual a 0 ou 1

Divisão e Conquista

- Não usa armazenamento, mas usa recursão
- Resulta em labirintos com um padrão visual muito característico, dividido em câmaras. Com uma grande parede longa central e suas recursões. É uma forma de Labirinto Fractal
- Relativamente fácil de resolver identificando as câmaras

3.
Bônus: O Algoritmo A*

Provando que os Problemas têm Soluções

O Algoritmo A*

Encontra o caminho entre a entrada e a saída

Usa função de custo e avaliação

F = H + C

AO PROGRAMA

OBRIGADO!

Perguntas?

@lucashoch 
hochlucassilva@gmail.com