Backtracking Usando Pilha

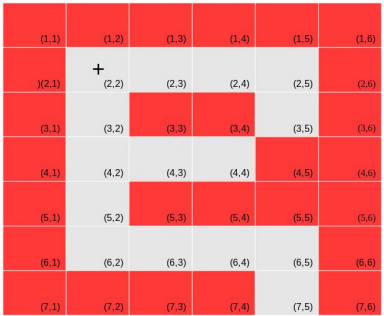
Daniel Oliveira

Departamento de Informática - UFPR

Janeiro 2021

Labirinto - Achar uma saída

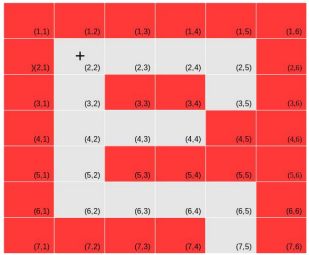
labirinto
Instância: (*labirinto*, *posicao*), onde *labirinto* é uma matriz de $n \times m$ e *posicao* indica o ponto inicial dentro do labirinto.
Resposta: Um caminho do ponto inicial até uma borda do labirinto



Labirinto - Recursão

busca_solucao(*labirinto*, *posicao*)

Se *fim_do_jogo*(*labirinto*, *posicao*)
    Devolva Verdadeiro
Senão
    Para *todo* x_l *pertencente ao domínio* D_k
        Insere x_l em *labirinto*
        Se *labirinto é válido E busca_solucao(labirinto*, x_l*)*
            Devolva Verdadeiro
        Remove x_l de *labirinto*
    Devolva Falso



Backtrack usando Pilha

- E se recursão não for possível ou a melhor opção?
  - Algumas linguagens/hardwares não suportam recursão ou são bem limitadas (e.g., GPUs)
  - Algum ganho de eficiência sem recursão
- Solução: Usar pilha para memorizar as decisões tomadas
  - Enquanto caminho no meu espaço de busca, empilho as decisões tomadas
  - Quando preciso fazer um backtrack (volta), desempilho uma decisão
- Diferente de recursão, preciso salvar também as decisões já tomadas por cada elemento considerado
  - Ao fazer o backtrack, preciso tomar uma decisão nova e não uma repetida

Backtrack usando Pilha - Algoritmo Geral

Solucao()

Inicializa P, *pilha* e demais estruturas
busca_solucao(P, *pilha*, ...)

busca_solucao(P, *pilha*, ...)

Enquanto *pilha nao vazia E solucao nao econtrada*
    Recupera elemento do topo da pilha
    Se *elemento ainda tem decisão válida dentro de* D_k
        Escolhe próxima decisão x_l tal que P é verdade
        Insere x_l em P
        Empilha x_l
    Senão
        Desempilha x_l
        Remove x_l de P
        Memoriza x_l como já analisado