PROJETO E OTIMIZAÇÃO DE ALGORITMOS

Prof. Michael da Costa Móra

Backtracking

Conceitos Gerais

- Estratégia de algoritmo
  - Abordagem para resolver um problema
  - Pode combinar várias abordagens
- Estrutura do algoritmo
  - Iterativo ⇒ executar ação em loop
  - Recursivo ⇒ reaplicar ação ao(s) subproblema(s)
- Tipo de problema
  - Satisfatório⇒ encontrar qualquer solução satisfatória
  - Otimização⇒ encontrarmelhorsoluções (vs. métrica de custo)

Uma pequena lista de categorias

- Muitos tipos de algoritmos devem ser considerados:
  - Algoritmos recursivos simples
  - Algoritmos de backtracking
  - Algoritmos de divisão e conquista
  - Algoritmos de programação dinâmica
  - Algoritmos gananciosos
  - Algoritmos de ramificação e limite
  - Algoritmos de força bruta
  - Algoritmos aleatórios

Backtracking

- Suponha que você tenha que fazer uma série de *decisões,* entre várias *escolhas,* onde
  - Você não tem informações suficientes para saber o que escolher
  - Cada decisão leva a um novo conjunto de escolhas
  - Alguma sequência de escolhas (possivelmente mais de uma) pode ser uma solução para o seu problema
- Backtracking é uma maneira metódica de experimentar várias sequências de decisões, até encontrar uma que “funcione”

Algoritmo de backtracking

- Com base na pesquisa recursiva em profundidade
- Abordagem
  1. Testa se a solução foi encontrada
  2. Se encontrar solução, devolva
  3. Senão para cada escolha que pode ser feita
     a) Faça essa escolha
     b) Re-inicie (recursivamente)
     c) Se a recursão retornar uma solução, retorne-a
  4. Se não restarem opções, retorne a falha
- Temos uma “árvore de busca”

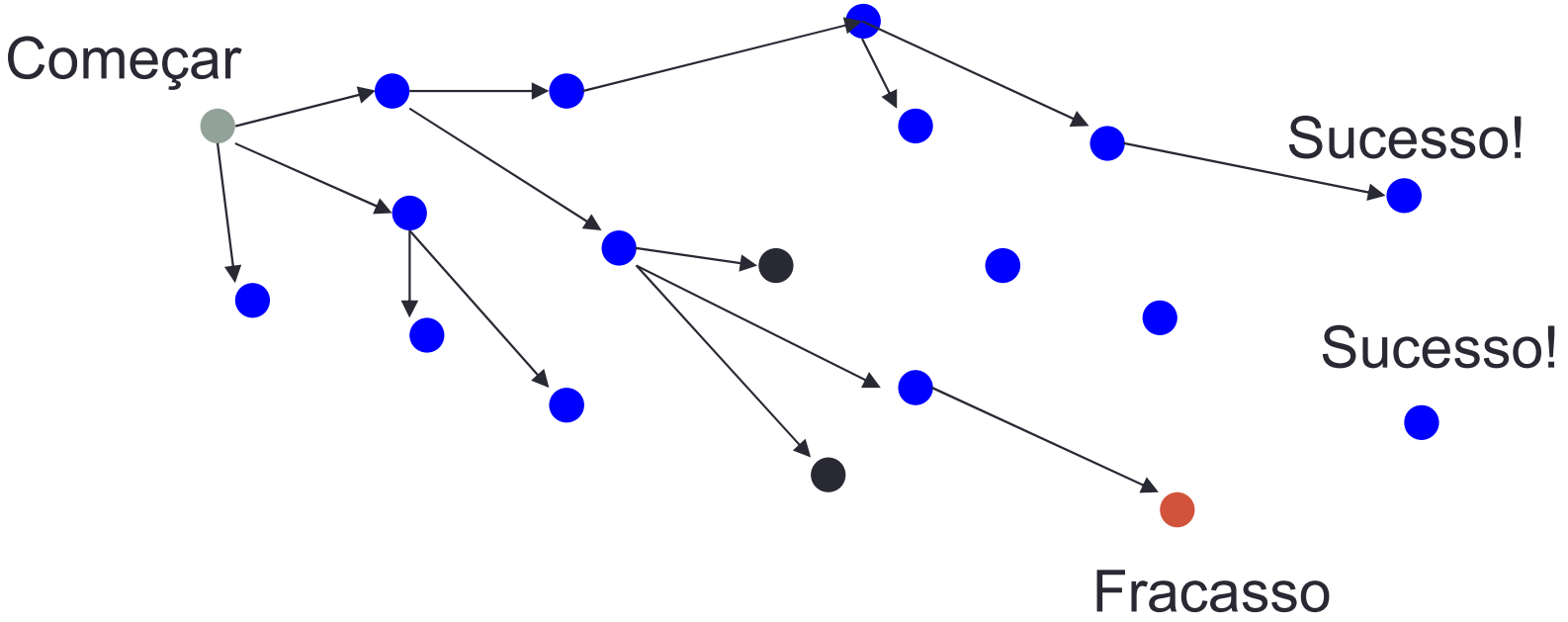
Algoritmo de Backtracking - Exemplo

- Encontre o caminho através do labirinto
  - Comece no início do labirinto
  - Se na saída, retorna true
  - Senão para cada etapa da localização atual
    - Encontrar caminho recursivamente
    - Retorne com o primeiro passo bem sucedido
    - Retorna false se todas as etapas falharem

Algoritmo de Backtracking - Exemplo

- Pinte um mapa com no máximo quatro cores
  - Se todos os países foram coloridos, retorne com sucesso
  - Senão para cada cor c de quatro cores e país n
    - Se o país n não for adjacente a um país que foi colorido c
      - Cor país n com cor c
      - Colorir recursivamente o país n+1
      - Se for bem sucedido, retorne o sucesso
  - Falha de devolução

Backtracking



O espaço do problema consiste em estados (nós) e ações (caminhos que levam a novos estados). Quando em um nó pode só pode ver caminhos para nós conectados

Se um nó apenas leva à falha, volte para seu "pai" nó. Tente outras alternativas. Se tudo isso levar ao fracasso então mais backtracking pode ser necessário.

Backtracking Recursivo

Pseudocódigo para algoritmos de backtracking recursivo

```
Se estiver em uma solução, retorne com
 sucesso
Para cada(escolha possível do estado/nó
 atual)
Faça essa escolha e dê um passo no caminho
  Use recursão para resolver o problema para o
   novo nó/estado
  Se a chamada recursiva for bem-sucedida,
   relate o sucesso para o próximo nível alto
  Saia da escolha atual para restaurar o
   estado no início do loop.

Falha
```

Backtracking

- Construa a árvore do espaço de estados:
  - Raiz representa um estado inicial
  - Os nós refletem escolhas específicas feitas para os componentes de uma solução.
    - Nós promissores e não promissores
    - folhas

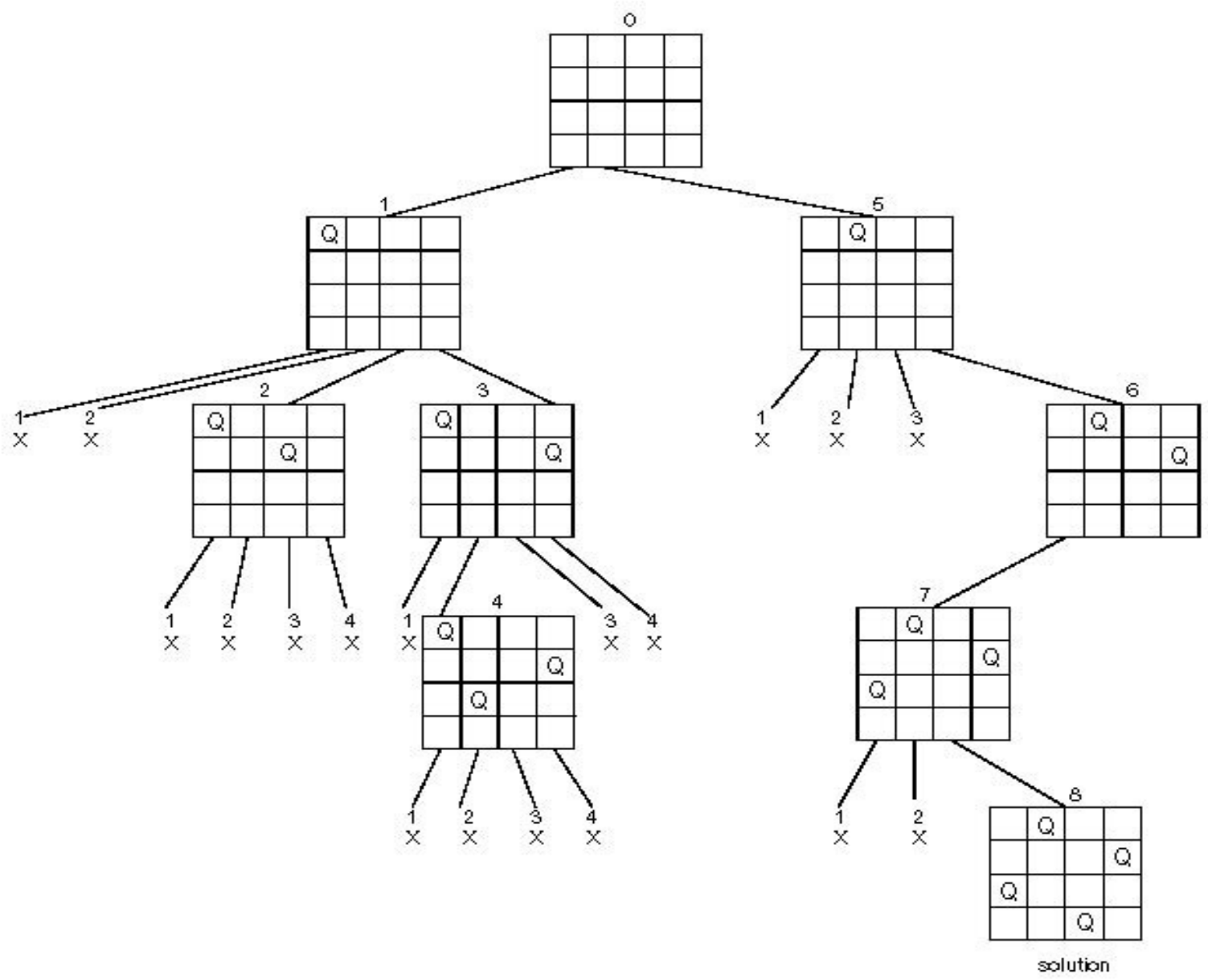
- Explore a árvore do espaço de estados usando pesquisa em profundidade

- "Podar" nós não promissores
  - Dfs para de explorar a subárvore enraizada em nós, levando a nenhuma solução e...
  - “recua” para seu nó pai

Exemplo: O Problema das n-rainhas

- Colocar *n* rainhas em um tabuleiro n por n de xadrez para que não haja dois deles na mesma linha, coluna ou diagonal

Árvore do Espaço de Estados do Problema das Quatro Rainhas



Uma solução possível: em Pseudo-Código

```
Create empty stack and set current position to 0

Repeat {

      loop from current position to the last position until
    valid position found //current row

      if there is a valid position {
         push the position to stack, set current position to 0
                                     // move to next row
      }

      if there is no valid position {
         if stack is empty, break // stop search
         else pop stack, set current position to next position
                                  // backtrack to previous row
      }

      if stack has size N { // a solution is found
         pop stack, set current position to next position
                             // backtrack to find next solution
      }

}
```

Exercícios

- Continue a busca de backtracking por uma solução para o problema das quatro rainhas para encontrar as demais soluções para o problema.