ALGORITMOS AVANÇADOS

Prof. Michael da Costa Móra

Backtracking

Conceitos gerais

- estratégia para design de algoritmos
  - Abordagem para resolver um problema
  - Pode combinar várias abordagens
- estrutura do algoritmo
  - Iterativo ⇒ executar ação no circuito
  - recursiva ⇒ reaplicar a ação a subproblema (s)
- tipo de problema
  - satisfatível ⇒ encontrar qualquer solução satisfatória
  - Otimização ⇒ encontrar melhor soluções (comparada a métrica de custo)

Uma pequena lista de categorias

- Muitos tipos Algoritmo devem ser considerados:
  - algoritmos recursivos simples
  - Backtracking
  - Divisão e conquista
  - algoritmos de programação dinâmica
  - algoritmos gulosos
  - Branch and bound
  - algoritmos de força bruta
  - algoritmos randomizados

backtracking

- Suponha que você tem que fazer uma série de *decisões,* entre os vários *escolhas,* Onde
  - Você não tem informações suficientes para saber o que escolher
  - Cada decisão leva a um novo conjunto de opções
  - Alguns seqüência de escolhas (possivelmente mais de um) pode ser uma solução para o seu problema
- backtracking é uma maneira metódica de experimentar várias seqüências de decisões, até encontrar um que “funciona”

backtracking Algorithm

- Baseado em pesquisa recursiva em profundidade
- Abordagem
  1. Testa se solução foi encontrada
  2. Se for encontrada solução, devolvê-la
  3. Senão para cada escolha que pode ser feita
     a) Fazer essa escolha
     b) Fazer a recursão
     c) Se recursão retorna uma solução, devolvê-lo
  4. Se não houver opções restantes, retornar falha
- Algumas vezes chamado de caminhamento na “árvore de busca”

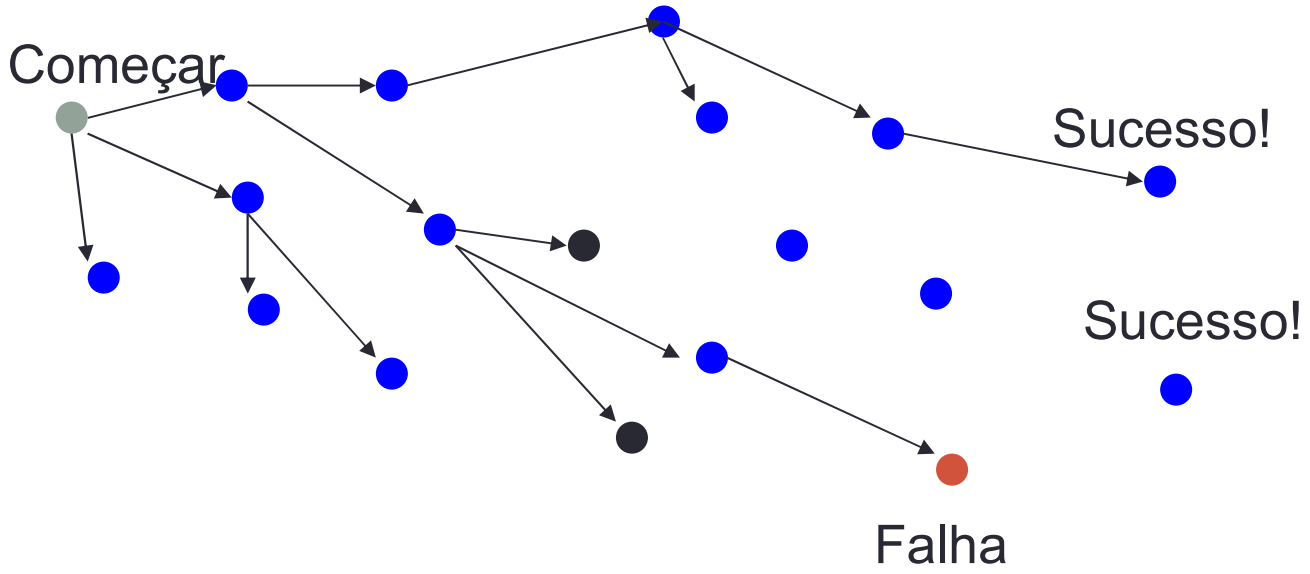
Retrocesso Algoritmo - Exemplo

- Encontrar o caminho através do labirinto
  - Comece no início do labirinto
  - Se na saída, retornar true
  - Senão para cada etapa da localização atual
    - Recursivamente encontrar o caminho
    - Retorne o primeiro passo bem sucedido
    - Retorne falso se todas as etapas falharem

Retrocesso Algoritmo - Exemplo

- Colorir um mapa com não mais de quatro cores
  - Se todos os países foram coloridos retorne sucesso
  - Senão para cada cor c de quatro cores e país n
    - Se o país n não é adjacente a um país que foi colorido c
      - Colori país n com a cor c
      - recursivamente colori país n + 1
      - Se for bem sucedido, retornar sucesso
  - falha de retorno

backtracking



Espaço de problema consiste em estados (nós) e ações (Caminhos que levam a novos estados). Quando em um nó pode só ver caminhos para nós conectados

Se um nó só leva ao fracasso voltar ao seu "pai" nó. Tente outras alternativas. Se todos estes levar ao fracasso em seguida, mais recuo pode ser necessário.

Backtracking recursiva

pseudo-código para algoritmos de retrocesso recursiva

If at a solution, return success
for(every possible choice from current state / node)
  Make that choice and take one step along path
  Use recursion to solve the problem for the new node / state
  If the recursive call succeeds, report the success to the next high level
  Back out of the current choice to restore the state at the beginning of the loop.
Report failure

backtracking

- construir o árvore de espaço de estado:
  - Root represents an initial state
  - Nodes reflect specific choices made for a solution's components.
    - Promising and nonpromising nodes
    - leaves

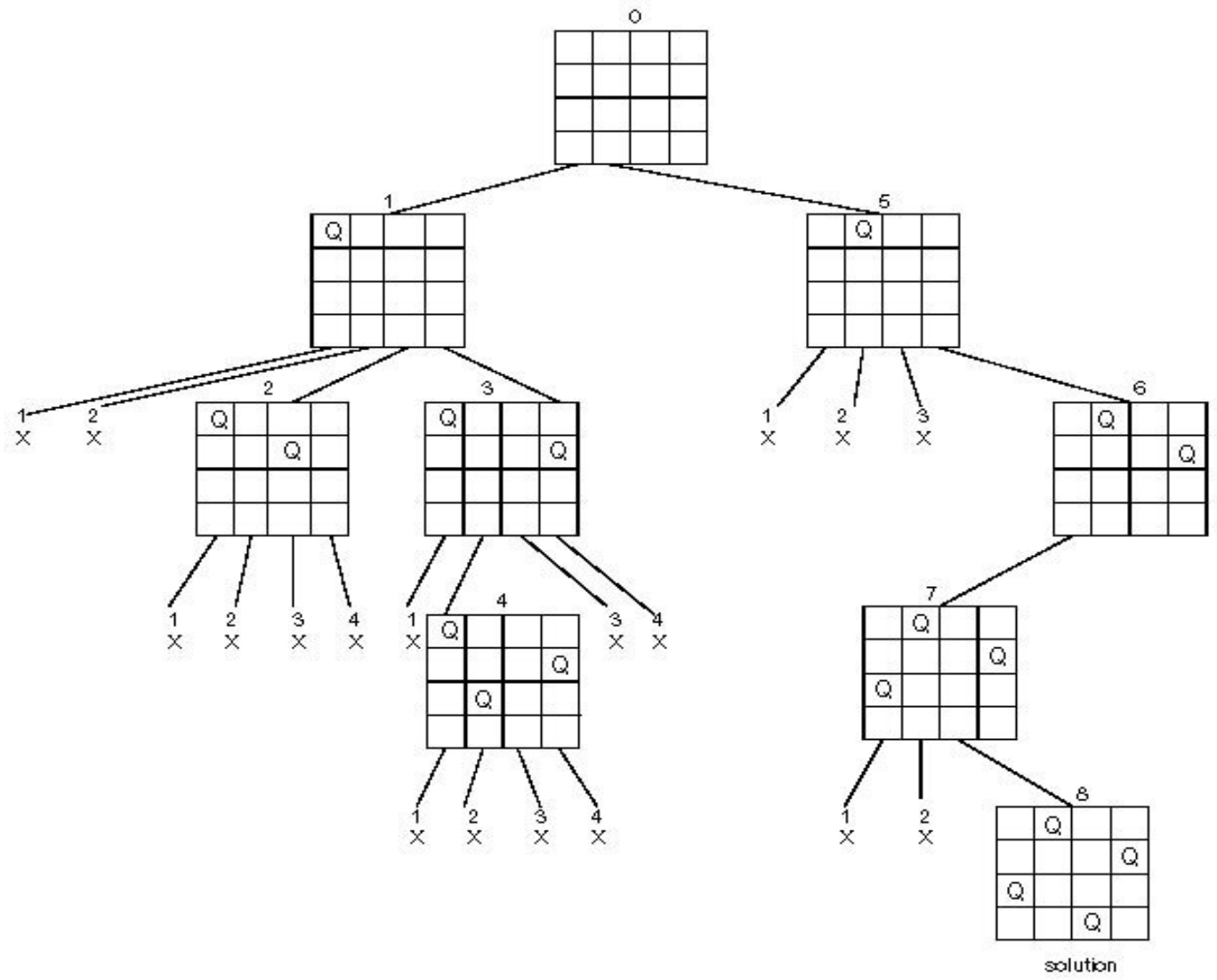
- Explore a árvore de espaço de estado usando busca em profundidade

  - "Prune" non-promising nodes
    - dfs stops exploring subtree rooted at nodes leading to no solutions and...
    - "backtracks" to its parent node

Exemplo: O *n*problema -Queen

- Lugar, colocar *n* rainhas em uma *n* por *n* tabuleiro de xadrez de modo que não há dois deles estão na mesma linha, coluna ou diagonal

Árvore Espaço Estado do Problema Quatro rainhas



O algoritmo de retrocesso

- Backtracking é realmente muito simples - nós "exploraramos" cada nó, como segue:
- "Explorar" nó N:
  1. Se N é um nó objetivo, voltar "sucesso"
  2. Se N é um nó folha, retorne "fracasso"
  3. Para cada filho C de N,
     3.1. Explorar C
        3.1.1. Se C foi bem sucedida, o retorno "sucesso"
  4. Return "fracasso"