Projeto de algoritmos: uma visão prática

Aula 06

Rafael Melo
Instituto de Computação
Universidade Federal da Bahia

Backtracking
(Tentativa e erro)

Motivação

- Problemas de busca combinatória podem ser abordados por força bruta (enumeração ou busca exaustiva):
  a. Listar o espaço de busca.
  b. Examinar cada possibilidade.
  c. Retornar as soluções que satisfazem as restrições do problema.
- Ineficiente quando o espaço de busca é grande.
- Como fazer melhor?

Motivação

- ***Backtracking***
  - Refinamento da força bruta
  - Aborda problemas complexos de busca combinatória
  - Usada para encontrar conjuntos que satisfaçam restrições especificadas
  - De maneira geral, é mais eficiente que busca exaustiva
    - Só gera soluções candidatas promissoras
    - Boa parte das soluções podem ser eliminadas sem serem explicitamente examinadas

Formalização do problema

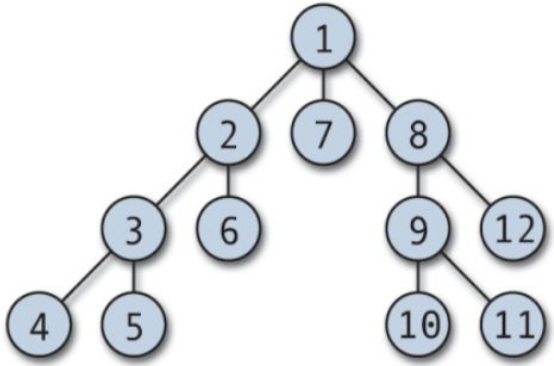
1. Problemas abordados por *Backtracking* possuem a forma:
   - Encontre um subconjunto $S \subseteq A_1 \times A_2 \times \ldots \times A_n$
   - onde A_i é um conjunto finito para $1 \leq i \leq n$
   - tal que cada elemento $(s_1, s_2, \ldots, s_n) \in S$ satisfaça certas propriedades.

Ideia geral

- As soluções são construídas um componente por vez.
- Avalia-se cada solução parcial como promissora ou não promissora.
- Se uma solução parcial é promissora, seleciona-se o próximo componente da solução.
- Caso contrário, o algoritmo retrocede (*backtracks*) e substitui o último componente da solução parcial por sua próxima opção.

Árvore de estados

- *Backtracking* induz uma árvore do espaço de soluções:
  - A raiz corresponde ao estado inicial (antes do início da busca por uma solução).
  - Um nó interno corresponde à uma solução parcial.
  - Um nó folha corresponde a uma solução parcial não promissora ou à solução final.
  - A busca é feita em profundidade (DFS).



Passos do *Backtracking*

- *Defina a representação da solução (normalmente um array).*
- *Estabeleça os conjuntos A_1 , . . . , A_n e a ordem em que seus elementos são processados.*
- *Estabeleça as condições que tornam uma solução parcial válida.*
- *Estabeleça um critério para identificar a solução final.*

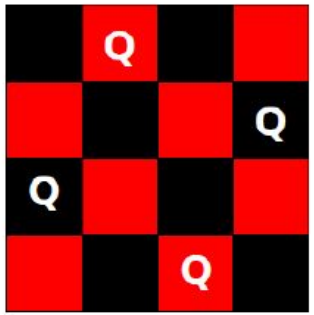
Passos do *Backtracking*

- O algoritmo induz uma árvore do espaço de estados.
- Cada nó representa enuplas parciais com os primeiros *i* componentes da solução parcial.
- Para cada $(x_1 , x_2 , \dots , x_i)$ válida, o algoritmo acha o próximo elemento em A_{i+1} que é consistente com
  - os valores de $(x_1 , x_2 , \dots , x_i)$;
  - as restrições do problema;
- e o adiciona à $(x_1 , x_2 , \dots , x_i)$ como seu *(i + 1)-ésimo* componente.

Problema das N rainhas

Problema das N rainhas

- Encontre todas as possibilidades de colocar *n* rainhas num tabuleiro de xadrez *n* × *n* tal que elas não possam se atacar:
  - cada linha contém exatamente uma rainha.
  - cada coluna contém exatamente uma rainha.
  - cada diagonal contém exatamente uma rainha.



Problema das N rainhas

- Quantos candidatos teríamos por força bruta para *n = 4*?
  - 16^4 = 65.536 possíveis formas de dispor 4 rainhas no tabuleiro.
- Busca exaustiva mais eficiente
  - coloca uma rainha em cada linha
  - 4^4 = 256 possibilidades de dispor 4 rainhas no tabuleiro.

Preliminares para o algoritmo

- Representação da solução: um array *X[0 . . n-1]* onde os indíces representam linhas e valores colunas
- Solução parcial *X[0 . . i]* é promissora se satisfazer:
  - rainhas devem estar em colunas diferentes.
  - rainhas não podem estar na mesma diagonal.
- Critério para definição de solução final: *i = n*.

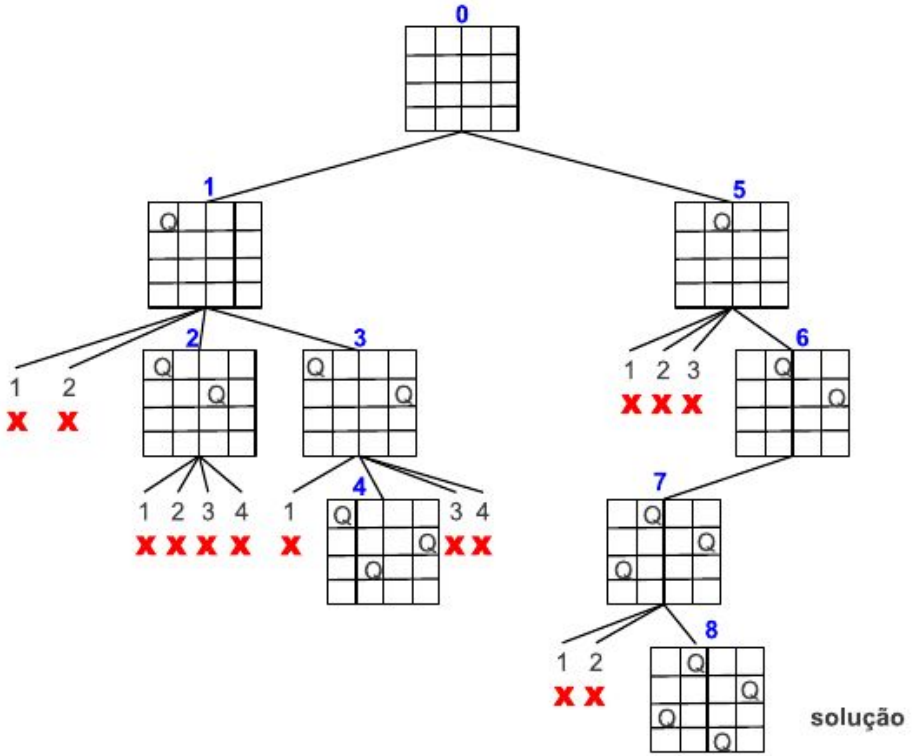
Algoritmo

```python
def VALID(X, linha, coluna):
    for i in range(linha):
        if X[i] == coluna or abs(X[i] - coluna) == abs(i - linha):
            return False
    return True

def N_RAINHAS(X, linha, n, solucoes):
    if linha == n:
        solucoes.append(list(X))
        return
    for coluna in range(n):
        if VALID(X, linha, coluna):
            X[linha] = coluna
            N_RAINHAS(X, linha + 1, n, solucoes)


n = int(input())
X = [0] * n
solucoes = []
N_RAINHAS(X, 0, n, solucoes)
for id, solucao in enumerate(solucoes):
    print("Solucao",id+1,"=", solucao)
```

Algoritmo



Atividades práticas

Contato:
rafael.melo@ufba.br
rafaelmelo.ufba@gmail.com