Documentação N-Rainhas Lisp

Pedro Loes

6/21/2021

Introdução

- O objetivo deste projeto foi desenvolver um algoritmo para resolver o problema das NRainhas na liguagem funcional Common Lisp. Este relatório apresenta a descrição do problema, a construção do algoritmo e a documentação do programa.

Problema

- O problema da N rainhas consiste em posicionar um número N de rainhas em um tabuleiro de xadrez.
- A peça rainha no jogo xadrez pode se movimentar um número ilimitado de casas nas linhas, colunas ou diagonais.
- O posicionamento deve ser realizado de forma que nenhuma rainha fique insegura, ou seja, nenhuma rainha pode estar na mesma linha, coluna ou diagonal que outra rainha.

Construção do Algoritmo

- O algoritmo foi construído com 3 funções:

  1. <**verifique**>

  – Verifica se a rainha da vez pode ser posicionada em determinada posição.

  2. <**posicione**>

  – Executa laço iterar sobre o espaço de busca.

  3. <**imprima**>

  – Imprime soluções encontradas pela programa.

- A ideia básica do programa é iterar sobre cada posição do tabuleiro N x N fornecido como argumento pelo usuário e verificar se as condições do posicionamento das N rainhas é seguro.
- O algoritmo itera sobre as linhas e colunas do tabuleiro procurando a primeira posição segura para cada rainha. Caso a posição inviabilize a solução final o algoritmo retorna e reposiciona a rainha anterior até encontrar uma solução onde as N rainhas estão seguras.
- A verificação de linhas e colunas é realizada comparando os indices da possível nova posição com os indices das rainhas já posicionadas.
- A verificação das diagonais é realizada calculando o valor absoluto de uma divisão.

1. Verifique

- A primeira função denominada <**verifique**>, verifica se uma posição em uma determinada casa do tabuleiro é segura.
- As linhas, colunas e diagonais são inspecionadas e a função retorna verdadeiro se as condições de segurança da posição forem satisfeitas.
- As funções <**cond**>, <**member**>, <**mapcar**>, <**car**>, <**cadr**>, <**lambda**> e <**abs**> da biblioteca base de liguagem Common Lisp foram utilizadas.
- A função recebe como parametros:

  1. A posição <**x**> no tabuleiro.
  2. A posição <**y**> no tabuleiro.
  3. A lista das posições das rainhas.

- A função de <**cond**> avalia duas condições:

  1. Se a rainha é membra da mesma linha que as rainhas anteriores.

  – A função <**member**> avalia se é verdadeiro o pertencimento da rainha a uma posição segura.
  – A função mapcar avalia cada posição em relação a lista de rainhas.
  – A função lambda recebe com paramentros a lista xy e aplica o mapcar para linhas e colunas.
  – A expressão lógica or avalia se <**x**> é igual a primeira posição ou <**y**> à segunda.

  2. Se a rainha é membra da mesma diagonal que a das rainhas anteriores.

  – A função <**member**> avalia se é verdadeiro o pertencimento da rainha a uma posição segura.
  – A função mapcar avalia cada posição em relação a lista de rainhas.
  – A função lambda recebe com paramentros a lista xy e a aplica o mapcar para diagonais.
  – A expressão lógica or avalia se o valor absoluto da divisão de <**x**> - a primeira posição por <**y**> - segunda posição de <**xy**> é igual a 1.

```
; Define função condição de segurança da rainha na posição x y
(defun verifique (x y rainhas)

  ; Verifica condição de posicionamento da rainha
  (cond

    ; Verifica se rainha da vez é membro da condição de linha
    ((member t (mapcar #'(lambda (xy)
                           (or (= x (car xy)) (= y (cadr xy)))) rainhas)) nil)

    ; Verifica se rainha da vez é membro da condição diagonal
    ((member t (mapcar #'(lambda (xy)
                           (= 1 (abs (/ (- x (car xy)) (- y (cadr xy)))))) rainhas)) nil)

    ; Retorna verdadeiro
    (t)
  )
)
```

1. Posicione

- A segunda função denominada <**posicione**>, itera sobre o tabuleiro.
- A função recebe como parametros:

  1. A posição <**x**> no tabuleiro.
  2. A posição <**y**> no tabuleiro.
  3. A lista das posições das <**rainhas**>.
  4. Número máximo <**max**> de rainhas e tamanho do tabuleiro.

```
; Define posicionamento recursivo da rainha em x e y na ordem: (1 1) ~ (max max)
(defun posicione (x y rainhas max)

  ; Condição de posicionamento seguro
  (cond

    ; Se verdadeiro Imprime tuplas (coluna linha) de posições da solução
    ((= max (length rainhas)) (print (list 'Solução rainhas)) (cdr rainhas))

    ; Caso contrario passa para proxima (coluna linha)
    ((or (> x max) (> y max)) (cdr rainhas))

    ; Verifica se pode posicionar a rainha
    ((verifique x y rainhas)

      ; Define conjunto, aplica laço recursivo com contador x + 1 e empilha rainha
      (setq rec (posicione (+ 1 x) 1 (append (list (list x y)) rainhas) max))

      ; Verifica condição de coluna
      (cond

        ; Condição de laço recursivo com contador y + 1 e conjunto rec verdadeiro
        ((equal rainhas rec) (posicione x (+ 1 y) rainhas max))
        (t rec)
      )
     )

    ; Executa a função de forma recursiva incrementando coluna y
    (t (posicione x (+ 1 y) rainhas max))
  )
)
```

3. Imprima

- A terceira função denominada <**imprima**>, verifica se uma posição em uma determinada casa do tabuleiro é segura.
- As linhas, colunas e diagonais são inspecionadas de forma recursiva e a função retorna verdadeiro se as condições de segurança da posição forem satisfeitas.

```
; Define função
(defun imprima (max rainhas)

    ; Executa o laço
    (posicione 1 1 rainhas max))

; Imprime solução com 4 rainhas
(print (list 'Solução (imprima 4 '())))
```

Conclusão