N-Rainhas Commom Lisp

Pedro Loes

6/23/2021

Introdução

- O objetivo deste projeto foi desenvolver um programa para resolver o problema das N-Rainhas na linguagem funcional Common Lisp. Este relatório apresenta a descrição do problema, explica o algorítimo, documenta o código e exemplifica o uso do programa.

Problema N-Rainhas

- O problema consiste em posicionar um número N de rainhas em um tabuleiro de xadrez $X_{N,N}$.
- A peça rainha pode ser movimentada um número ilimitado de casas nas linhas, colunas ou diagonais.
- Uma posição é segura se não está na mesma linha, coluna ou diagonal das posições ocupadas por outras rainhas.

Construção do Algoritmo

- O algoritmo foi desenvolvido com a implementação de 3 funções:

  1. <**verifique**>

  – Verifica se uma posição é segura.

  2. <**posicione**>

  – Executa laço para iterar sobre o espaço de busca.

  3. <**imprima**>

  – Imprime soluções encontradas pelo programa.

- O objetivo do programa é posicionar um número **n** de rainhas fornecido pelo usuário.
- O funcionamento do algoritmo consiste em iterar sobre as linhas e colunas do tabuleiro $X_{n,n}$ procurando uma posição segura para cada rainha.
- A verificação de segurança para linhas e colunas é realizada comparando os índices da possível nova posição com os índices de dominância das rainhas já posicionadas.
- A verificação de segurança nas diagonais é confirmada se o valor absoluto da divisão das distâncias entre as rainhas for diferente de 1.
- Caso uma posição selecionada inviabilize o posicionamento de qualquer rainha posterior, o algoritmo retorna e reposiciona as rainhas anteriores de forma recursiva até encontrar uma solução onde as **n** rainhas estarão seguras.

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 linguagem Common Lisp foram utilizadas.
- A função recebe como parâmetros:

  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 parâmetros 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 parâmetros 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 parâmetros:
  1. A posição <**x**> no tabuleiro.
  2. A posição <**y**> no tabuleiro.
  3. A lista das posições das <**rainhas**>.
  4. Número máximo <**n**> de rainhas e dimensão do tabuleiro.
- A função de <**cond**> avalia três condições:
  1. Se o tamanho da lista de rainhas posicionadas é igual ao máximo
  - Caso verdadeiro imprime tupla (coluna, linha) da maior para menor.
  2. Se <**x**> ou <**y**> é maior que o máximo
  - Caso verdadeiro passa para próxima rainha da lista rainhas.
  3. Se <**verifica**> permite o posicionamento.
  - Caso verdadeiro adiciona posição a lista rainha caso contrário chama a função verifica de forma recursiva nas posições <**x**> + 1 e <**y**> + 1.

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

  ; Condição de posicionamento seguro
  (cond

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

    ; Caso contrário passa para proxima (coluna linha)
    ((or (> x n) (> y n)) (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) n))

      ; 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 n))
        (t rec)
      )
     )

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

3. Imprima

- A terceira função denominada <**imprima**>, imprime os resultados do algoritmo no formato de tuplas da última coluna para a primeira.
- A função recebe como parâmetro <**n**> que define o número de rainhas e o tamanho do tabuleiro.
- Foram defindas 3 excessões para o parâmetro <**n**>:
  1. Verifica se <**n**> é número.
  2. Verifica se <**n**> é número inteiro.
  3. Verifica se <**n**> é número positivo.
- Executa a função posicione começando com <**x**> = 1 e <**y**> = 1 e o parâmetro <**n**>.

```
; Define função
(defun imprima (n)

    ; Verifica erro de exceção para entrada de caracteres
    (assert (numberp n) (n) "Erro: O parâmetro <n> precisa ser um número.")

    ; Verifica erro de exceção para número real
    (assert (integerp n) (n) "Erro: O parâmetro <n> precisa ser um número inteiro.")

    ; Verifica erro de exceção para entrada de número negativo
    (assert (not (< n 1)) (n) "Erro: O parâmetro <n> precisa ser um número positivo.")

    ; Executa a função posicione
    (posicione 1 1 '() n)
)
```

Exemplo

- Um exemplo da execução do programa é a resolução do problema com o parâmetro <**4**> para indicar a resolução do problema com 4 rainhas em um tabuleiro 4 x 4.

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

```
(SOLUÇÃO ((4 3) (3 1) (2 4) (1 2)))
(SOLUÇÃO ((4 2) (3 4) (2 1) (1 3)))
(SOLUÇÃO NIL)
```

- O compilador Common Lisp do servidor **rextester** permite executar o programa no sítio:
  - https://rextester.com

Referências

- Pratical Common Lisp
- www.tutorialspoint.com/lisp
- You Tube