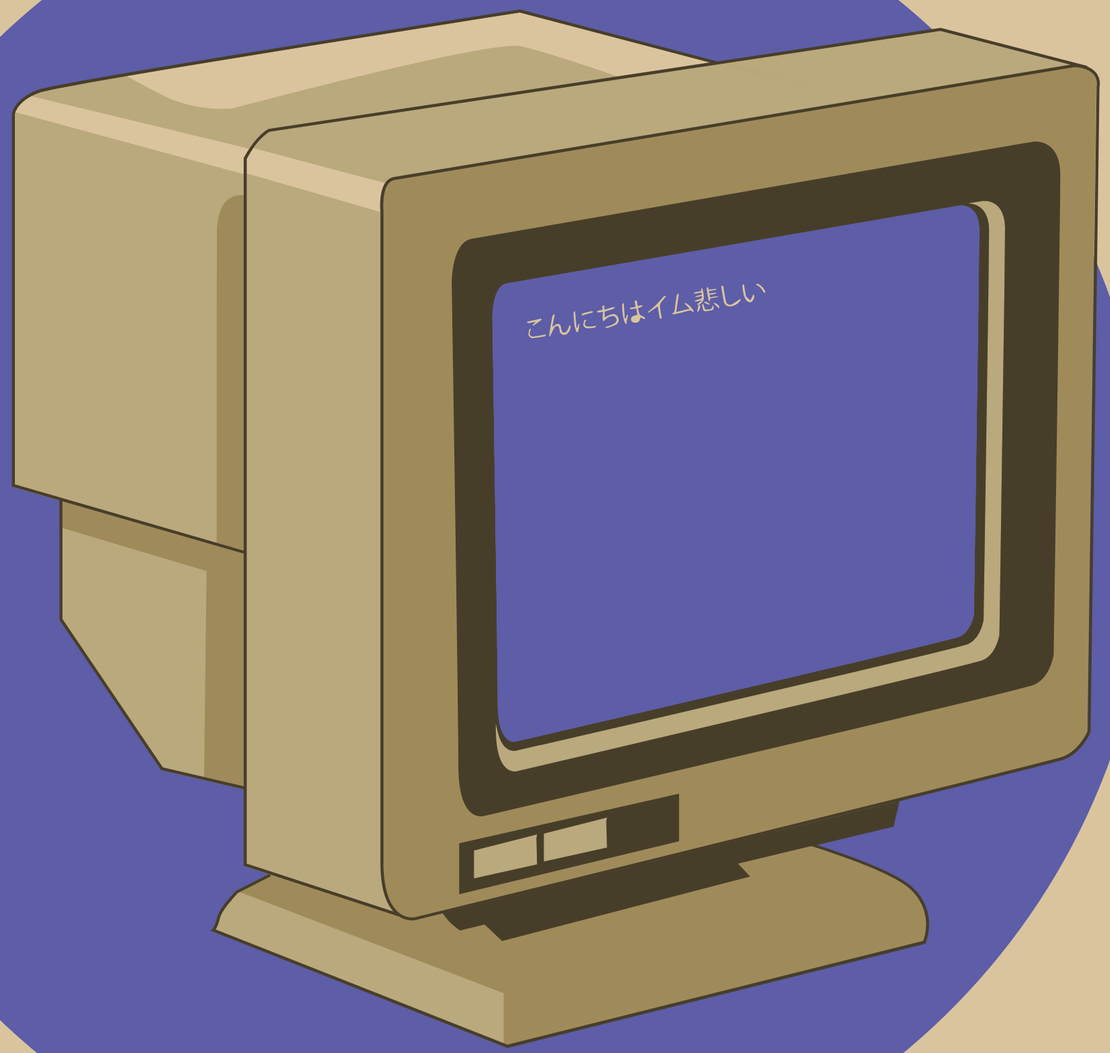


AED 2 - 2025.1

PROBLEMA DO CAVALO

(KNIGHT'S TOUR)

CAMILA VIDMONTIENE
CLARISSA BERLIM
ERALDO BOTELHO
FELIPE MITSUO
JOÃO COUTINHO

TRABALHO 6

O **Problema do Cavalo (ou Knight's Tour)** é um problema clássico da matemática e da ciência da computação baseado nas regras do jogo de xadrez.

Ele consiste em encontrar uma sequência de movimentos de um cavalo no tabuleiro de xadrez (8x8) de forma que ele visite todas as 64 casas exatamente uma vez, sem repetir nenhuma.

O cavalo, no xadrez, se move em um padrão em “L”: duas casas em uma direção (horizontal ou vertical) e depois uma casa perpendicular. Isso dá até 8 possibilidades de movimento a partir de uma posição, dependendo da borda do tabuleiro.

O desafio é encontrar uma sequência contínua que cubra todo o tabuleiro, o que é chamado de "passeio do cavalo". Quando essa sequência termina na mesma casa de onde começou, o passeio é chamado de fechado; caso contrário, é um passeio aberto.

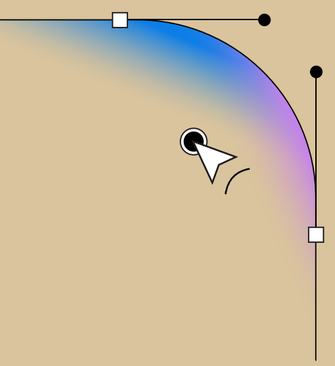
IMPLEMENTAÇÃO

O algoritmo desenvolvido tenta construir uma solução recursivamente, explorando possibilidades e voltando atrás (backtrack) quando não há mais movimentos válidos.

A função validos representa a chamada recursiva com tentativa e erro, o que caracteriza backtracking.

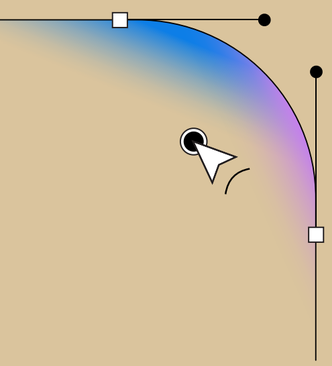
A heurística de Warnsdorff é aplicada na função sortPoss, que prioriza as casas com menos opções de movimento subsequente, reduzindo o número de tentativas inúteis e tornando o algoritmo mais eficiente.

CÓDIGO

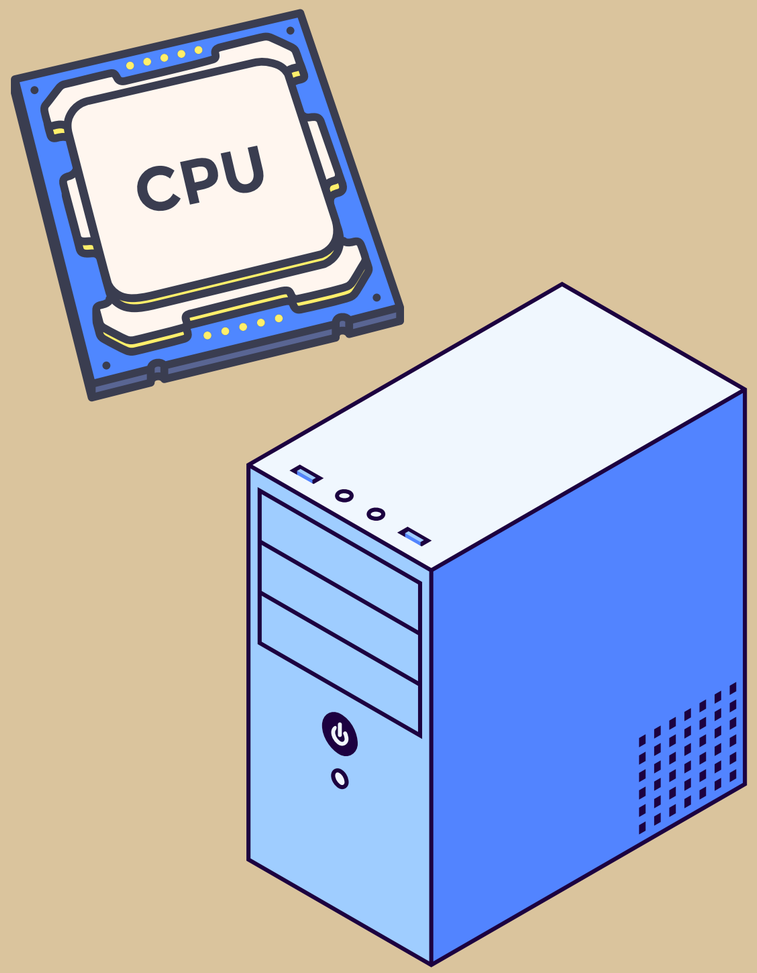


```python
from os import system as sys
import time

def possiveis(movimentos, casaAtual):
    possiveis = []
    for i in range(-2, 3):
        for j in range(-2, 3):
            casa = casaAtual+j+8*i
            if i*j != 0 and i != j and 0 <= casa//8 < 8 and 0 <= casaAtual%8+j < 8 and i+j != 0 and casa not in movimentos:
                possiveis.append(casa)
    return possiveis

def sortPoss(movimentos, poss):
    numeroDeBifurcacoes = lambda atual: len(possiveis(movimentos + [atual], atual))
    newPoss = sorted(poss, key = numeroDeBifurcacoes)
    return newPoss

def validos(movimentos, casaAtual):
    if len(movimentos) == 64:
        return movimentos
    poss = sortPoss(movimentos, possiveis(movimentos, casaAtual))
    if len(poss) != 0:
        for i in poss:
            movimentos += [i]
            caminho = validos(movimentos, i)
            if caminho:
                return caminho
            movimentos.pop()
    else:
        return None
```

CÓDIGO



```
31
32  def mostrarCaminhoParcial(movimentos, limite):
33      print(" "*3, end="")
34      for i in range(8):
35          print(f"{chr(97+i):^3}", end="")
36      print("")
37      for i in range(8):
38          print(f"{i+1:^3}", end="")
39          for j in range(8):
40              print(f"{movimentos[i*8+j] if movimentos[i*8+j] <= limite else " ":3}", end="")
41          print("")
42      print("")
43
44  def mostrarCaminhoTotal(movimentos):
45      for i in range(64):
46          sys("cls")
47          mostrarCaminhoParcial(movimentos, i+1)
48          time.sleep(0.15)
49
50  def main():
51      movimentos = validos([0], 0)
52      movimentos = {i:movimentos.index(i)+1 for i in movimentos}
53      mostrarCaminhoTotal(movimentos)
54      fim = input("Digite a tecla ENTER para sair.")
55
56  if __name__ == "__main__":
57      main()
58
```



COMPLEXIDADE

A complexidade teórica bruta do problema do cavalo, sem heurísticas, é:

- $O(8^{64})$ (pior caso), pois em cada uma das 64 casas o cavalo pode fazer até 8 movimentos diferentes.

No entanto, com a heurística de Warnsdorff, o número de caminhos testados é drasticamente reduzido. Em muitos casos, a solução é encontrada em tempo praticamente linear, porque a heurística evita ramificações desnecessárias.

Função	Complexidade aproximada
possiveis()	O(1)
sortPoss()	O(8n)
validos()	Ω(n) e O(8^n)

Como n = 64, o algoritmo com heurística pode terminar em tempo praticamente constante, apesar da complexidade teórica ser exponencial.

TEMPO DE EXECUÇÃO

Cavalo em A1:

```
    a  b  c  d  e  f  g  h
 1   1 16 31 40  3 18 21 56
 2  30 39  2 17 42 55  4 19
 3  15 32 41 46 53 20 57 22
 4  38 29 48 43 58 45 54  5
 5  33 14 37 52 47 60 23 62
 6  28 49 34 59 44 63  6  9
 7  13 36 51 26 11  8 61 24
 8  50 27 12 35 64 25 10  7

Tempo: 0.3533046245574951 segundos.
```

Cavalo em C2:

```
    a  b  c  d  e  f  g  h
 1   2 23  4 19 44 39 14 17
 2   5 20  1 40 15 18 45 38
 3  24  3 22 43 46 51 16 13
 4  21  6 59 50 41 48 37 52
 5  62 25 42 47 58 53 12 33
 6   7 28 63 60 49 34 55 36
 7  26 61 30  9 54 57 32 11
 8  29  8 27 64 31 10 35 56

Tempo: 0.4062228202819824 segundos.
```

Cavalo em D4:

```
    a  b  c  d  e  f  g  h
 1   3 40  5 22 19 26 29 24
 2   6 21  2 41 30 23 18 27
 3  47  4 39 20 59 28 25 32
 4  38  7 48  1 42 31 58 17
 5  11 46 37 60 49 62 33 56
 6   8 51 10 43 36 57 16 63
 7  45 12 53 50 61 14 55 34
 8  52  9 44 13 54 35 64 15

Tempo: 2.171722888946533 segundos.
```

OBRIGADO!