

Paradigmas de Projeto de Algoritmos

Backtracking - Tentativa e Erro

Professora:

Fátima L. S. Nunes

Tentativa e erro - conceitos

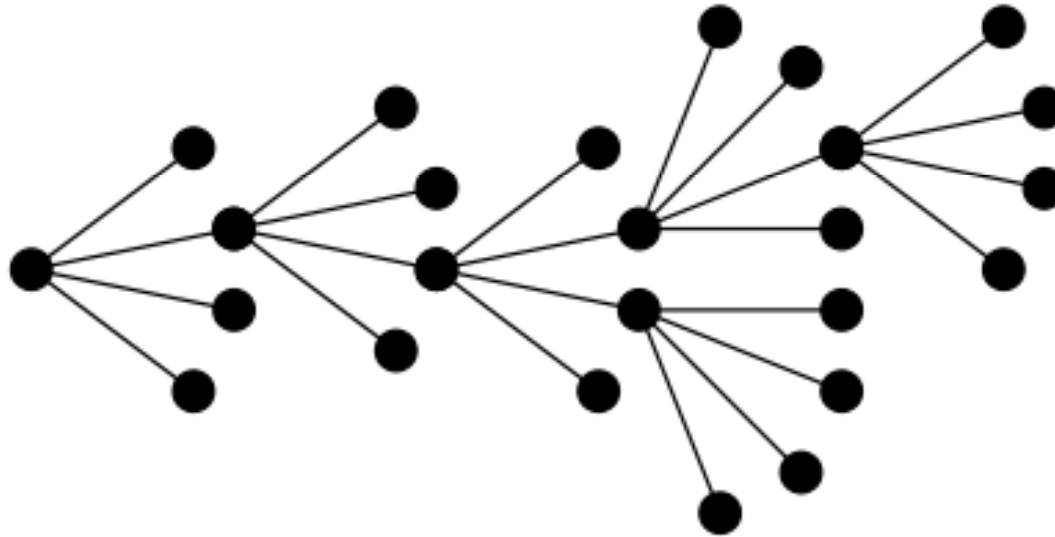
- Como o próprio nome diz:

Tentativa e erro - conceitos

- Como o próprio nome diz:
 - tentar enquanto estiver errado
- Técnica de projetos de algoritmos:
 - útil para problemas cuja solução consiste em tentar todas alternativas propostas;
 - idéia: decompor o processo em um número finito de subtarefas que devem ser exploradas;
 - processo geral pode ser visto como processo de tentativas que constrói e percorre uma árvore de subtarefas.

Tentativa e erro - conceitos

- processo geral pode ser visto como processo de tentativas que constrói e percorre uma árvore de subtarefas.



Tentativa e erro - conceitos

- Algoritmos com esta técnica não têm uma regra fixa de computação.
- Funcionamento geral:
 - passos para obtenção da solução final são tentados e registrados;
 - caso um passo não leve à solução final, é retirado e apagado do registro.
- Pesquisa na árvore de solução: muitas vezes tem crescimento exponencial.

Tentativa e erro - conceitos

- Voltando ao funcionamento geral:
 - passos para obtenção da solução final são tentados e registrados;
 - caso um passo não leve à solução final, é retirado e apagado do registro.
- **Como registrar e remover os passos com o que sabemos até agora?**

Tentativa e erro - conceitos

- Voltando ao funcionamento geral:
 - passos para obtenção da solução final são tentados e registrados;
 - caso um passo não leve à solução final, é retirado e apagado do registro.
- **Como registrar e remover os passos com o que sabemos até agora?**
 - **Solução 1:** use um array

Tentativa e erro – exemplo Labirinto

¶ Dado um labirinto, encontre um caminho da entrada à saída:

¶ em cada interseção, você só pode seguir 2 caminhos:

¶ ir à direita;

¶ ir acima.




¶ há obstáculos que provocam desvios;

¶ você não tem informação suficiente para escolher corretamente;

¶ cada escolha leva a outro conjunto de escolhas

¶ uma ou mais sequências de escolhas pode ser a solução.

Tentativa e erro – exemplo Labirinto

	0	1	2	3
0				
1		X		
2				
3			X	X

Origem = (3,0)





Destino = (1,3)

 ORIGEM

 DESTINO

X = obstáculos

Tentativa e erro – exemplo Labirinto

	0	1	2	3
0				
1		X		
2				
3				X

Origem = (3,0)

Destino = (1,3)

Move Direita (3,1)

Move Direita (3,2)

O que acontece???

 ORIGEM





 DESTINO

X = obstáculos

Array de registro:

3,1	3,2								
-----	-----	--	--	--	--	--	--	--	--

Tentativa e erro – exemplo Labirinto

	0	1	2	3
0				
1		X		
2				
3				X

Origem = (3,0)

Destino = (1,3)

Move Direita (3,1)

Move Direita (3,2)

O que acontece???

NÃO PODE

 ORIGEM




 DESTINO

X = obstáculos

Array de registro:

3,1	3,2								
-----	-----	--	--	--	--	--	--	--	--

Tentativa e erro – exemplo Labirinto

	0	1	2	3
0				
1		X		
2				
3			X	X

Origem = (3,0)

Destino = (1,3)

Move Direita (3,1)

Move Direita (3,2)

Não pode

Apaga último (3,2)

Volta anterior (3,1)

 ORIGEM





 DESTINO

X = obstáculos

Array de registro:

3,1									
-----	--	--	--	--	--	--	--	--	--

Tentativa e erro – exemplo Labirinto

	0	1	2	3
0				
1		X		
2				
3			X	X

Origem = (3,0)

Destino = (1,3)

Move Direita (3,1)

Move Direita (3,2)

Não pode

Apaga último (3,2)

Volta anterior (3,1)

Move Acima (2,1)

 ORIGEM






 DESTINO

X = obstáculos

Array de registro:

3,1	2,1								
-----	-----	--	--	--	--	--	--	--	--

Tentativa e erro – exemplo Labirinto

	0	1	2	3
0				
1		X		
2				
3			X	X

Origem = (3,0)

Destino = (1,3)

Move Direita (3,1)

Move Direita (3,2)

Não pode

Apaga último (3,2)

Volta anterior (3,1)

Move Acima (2,1)

Move Direita (2,2)

 ORIGEM







 DESTINO

X = obstáculos

Array de registro:

3,1	2,1	2,2							
-----	-----	-----	--	--	--	--	--	--	--

Tentativa e erro – exemplo Labirinto

	0	1	2	3
0				
1		X		
2				
3			X	X

 ORIGEM

 DESTINO

X = obstáculos

Array de registro:

3,1	2,1	2,2	2,3						
-----	-----	-----	-----	--	--	--	--	--	--

Origem = (3,0)

Destino = (1,3)

Move Direita (3,1)

Move Direita (3,2)

Não pode

Apaga último (3,2)







Volta anterior (3,1)

Move Acima (2,1)

Move Direita (2,2)

Move Direita (2,3)

Tentativa e erro – exemplo Labirinto

	0	1	2	3
0				
1		X		
2				
3			X	X

 ORIGEM

 DESTINO

X = obstáculos

Array de registro:

3,1	2,1	2,2	2,3	2,4					
-----	-----	-----	-----	-----	--	--	--	--	--

Origem = (3,0)

Destino = (1,3)

Move Direita (3,1)

Move Direita (3,2)

Não pode

Apaga último (3,2)

Volta anterior (3,1)

Move Acima (2,1)







Move Direita (2,2)

Move Direita (2,3)

Move Direita (2,4)

Não pode (estoura coluna)

Tentativa e erro – exemplo Labirinto

	0	1	2	3
0				
1		X		
2				
3			X	X

 ORIGEM

 DESTINO

X = obstáculos

Array de registro:

3,1	2,1	2,2	2,3						
-----	-----	-----	-----	--	--	--	--	--	--

Origem = (3,0)

Destino = (1,3)

Move Direita (3,1)

Move Direita (3,2)

Não pode

Apaga último (3,2)

Volta anterior (3,1)

Move Acima (2,1)

Move Direita (2,2)

Move Direita (2,3)







Move Direita (2,4)

Não pode (estoura coluna)

Apaga último (2,4)

Volta anterior (2,3)

Tentativa e erro – exemplo Labirinto

	0	1	2	3
0				
1		X		
2				
3			X	X

 ORIGEM

 DESTINO

X = obstáculos

Array de registro:

3,1	2,1	2,2	2,3	1,3					
-----	-----	-----	-----	-----	--	--	--	--	--

Origem = (3,0)

Destino = (1,3)

Move Direita (3,1)

Move Direita (3,2)

Não pode

Apaga último (3,2)

Volta anterior (3,1)

Move Acima (2,1)

Move Direita (2,2)

Move Direita (2,3)

Move Direita (2,4)

Não pode (estoura coluna)

Apaga último (2,4)

Volta anterior (2,3)

Move Acima (1,3)

CHEGOU (1,3)

Exercício – exemplo Labirinto

Faça o algoritmo para encontrar o destino, partindo da origem, para os seguintes labirintos:

	0	1	2	3
0		X		
1				●
2		X		
3	●	X	X	X

	0	1	2	3
0				
1				●
2	X	X		
3	●	X		

● ORIGEM

● DESTINO

X = obstáculos

Array de registro:

--	--	--	--	--	--	--	--	--	--

Exercício – exemplo Labirinto

Faça um algoritmo geral para encontrar o destino, partindo da origem, para os labirintos vistos anteriormente:

Exercício – exemplo Labirinto

Faça um algoritmo geral para encontrar o destino, partindo da origem, para os labirintos vistos anteriormente:

```
Enquanto não chegar ao destino e houver movimentos possíveis
  Enquanto movimento for válido
    Move direita
    Se fracasso
      Apaga último
      Volta posição anterior
    Move acima
  Fim Se
Fim Enquanto
Fim Enquanto
Se acabou movimentos possíveis
  Imprime "Impossível chegar ao destino"
Senão
  Imprime "Chegou ao destino"
Fim Se
```

Movimento válido =
Se não está ocupado E
Se está dentro das coordenadas E
Não acabou movimentos possíveis

Exercício – exemplo Labirinto

Faça um algoritmo geral para encontrar o destino, partindo da origem, para os labirintos vistos anteriormente:

```
Enquanto não chegar ao destino e houver movimentos possíveis
  Enquanto movimento for válido
    Move direita
    Se fracasso
      Apaga último
      Volta posição anterior
    Move acima
  Fim Se
Fim Enquanto
Se acabou movimentos possíveis
  Imprime "Impossível chegar ao destino"
Senão
  Imprime "Chegou ao destino"
Fim Se
```

Movimento válido =
Se não está ocupado E
Se está dentro das coordenadas E
Não acabou movimentos possíveis

Exercício – exemplo Labirinto

Alterar o algoritmo anterior considerando que para cada movimento agora é possível andar em todas as direções, nesta ordem: direita, acima, esquerda, abaixo.

Exercício – exemplo Labirinto

Alterar o algoritmo anterior considerando que para cada movimento agora é possível andar em todas as direções, nesta ordem: direita, acima, esquerda, abaixo.

```
Enquanto não chegar ao destino e houver movimentos possíveis
```

```
    Enquanto movimento for válido
```

```
        Move direita
```

```
        Se fracasso
```

```
            Apaga último
```

```
            Volta posição anterior
```

```
            Move acima
```

```
            Se fracasso
```

```
                Apaga último
```

```
                Volta posição anterior
```

```
                Move esquerda
```

```
                Se fracasso
```

```
                    Apaga último
```

```
                    Volta posição anterior
```

```
                    Move abaixo
```

```
            Fim Se
```

```
        Fim Se
```

```
    Fim Se
```

```
    Fim Enquanto
```

```
Fim Enquanto
```

```
Se acabou movimentos possíveis
```

```
    Imprime "Impossível chegar ao destino"
```

```
Senão
```

```
    Imprime "Chegou ao destino"
```

```
Fim Se
```


Tentativa e erro - conceitos

- Voltando ao funcionamento geral:
 - passos para obtenção da solução final são tentados e registrados;
 - caso um passo não leve à solução final, é retirado e apagado do registro.
- **Como registrar e remover os passos com o que sabemos até agora?**
 - Solução 1: use um array
 - Solução 2: recursividade!

Tentativa e erro - recursividade

- Registrando e removendo **com recursividade**:
 - Exemplo: fatorial do número 5

```
fatorial (n)
se n < 3
    retorna n
senão
    retorna n*
        fatorial (n-1)
fim se
```

```
n = 5
retorna n * fatorial (4)
```

**registra dados chamada 0
faz chamada recursiva**

```
n = 4
retorna n * fatorial (3)
```

**registra dados chamada 1
faz chamada recursiva**

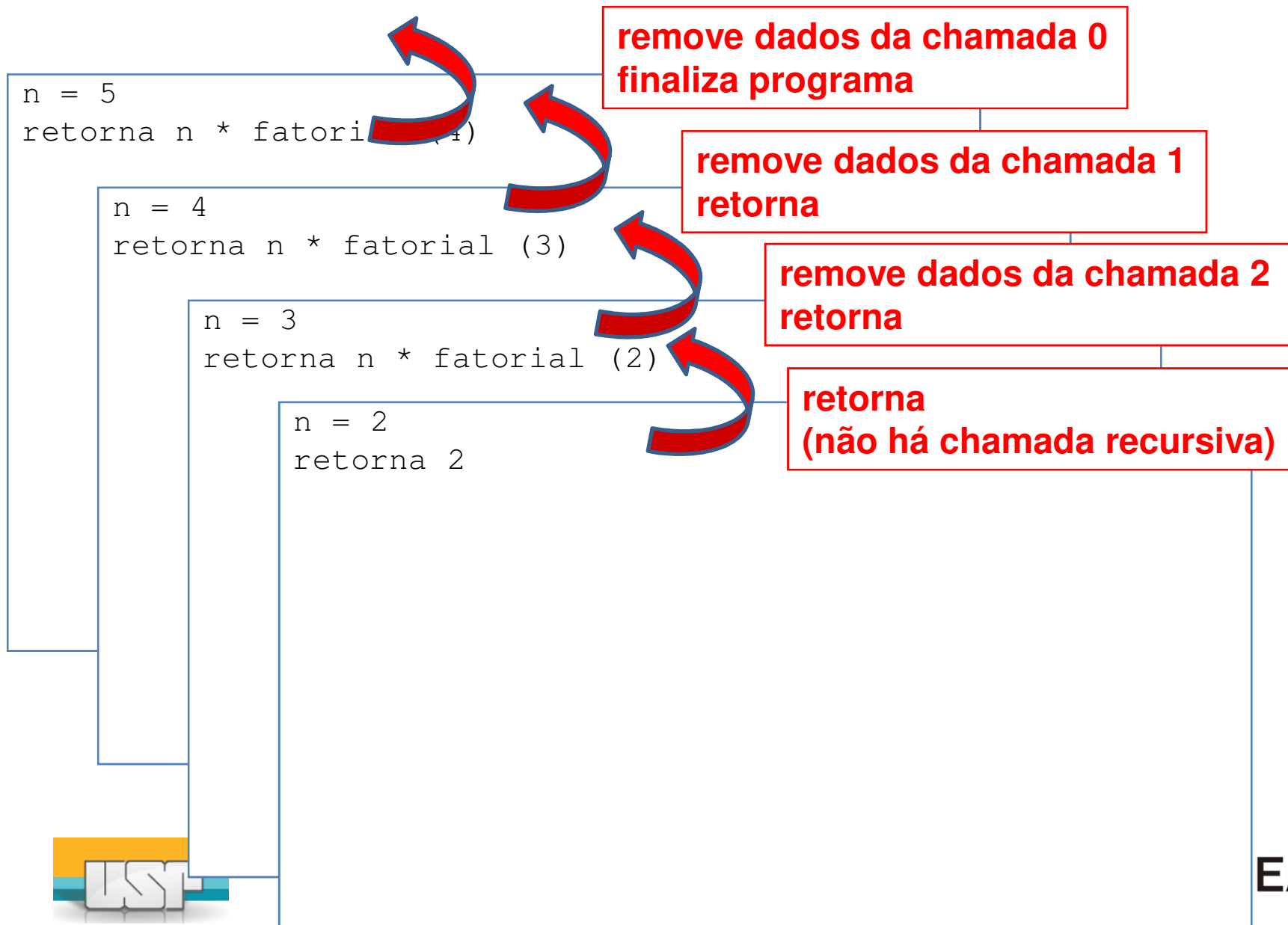
```
n = 3
retorna n * fatorial (2)
```

**registra dados chamada 2
faz chamada recursiva**

```
n = 2
retorna 2
```

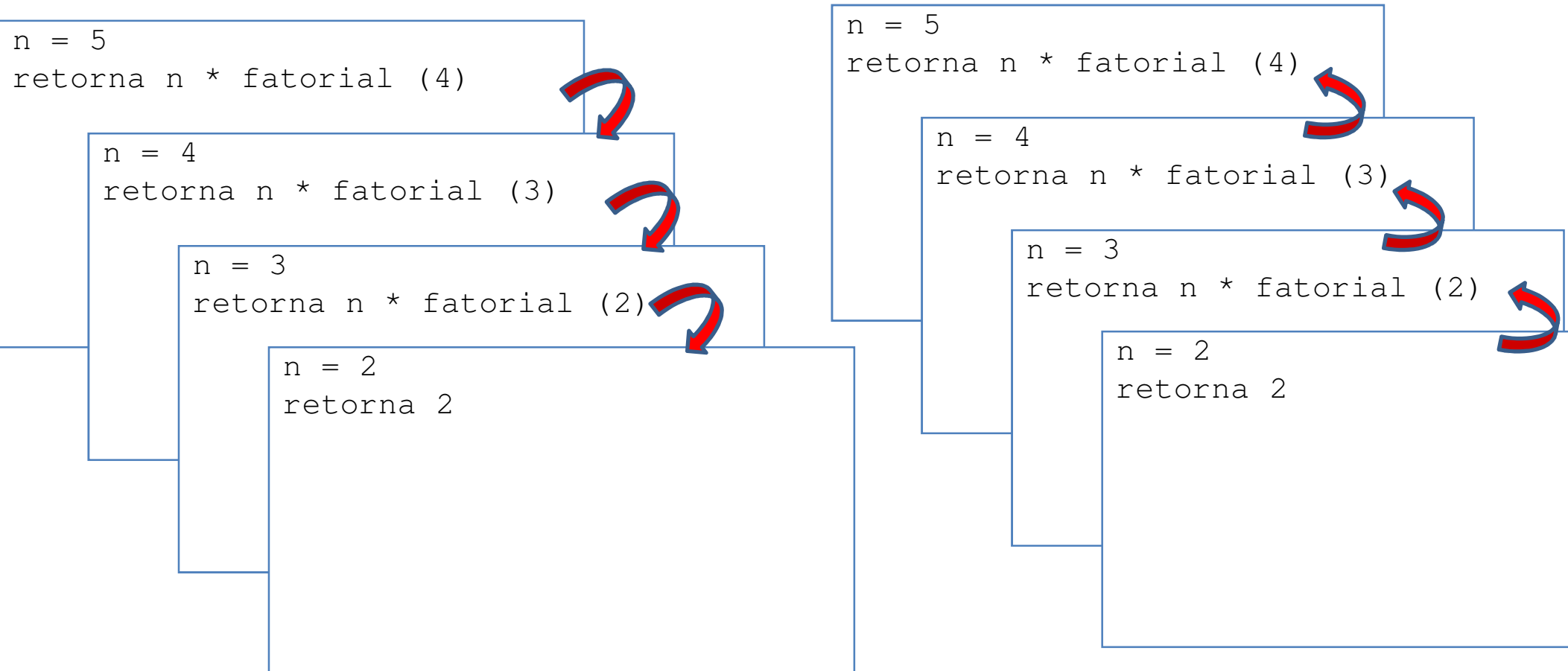
Tentativa e erro - recursividade

- Registrando e removendo **com recursividade**:



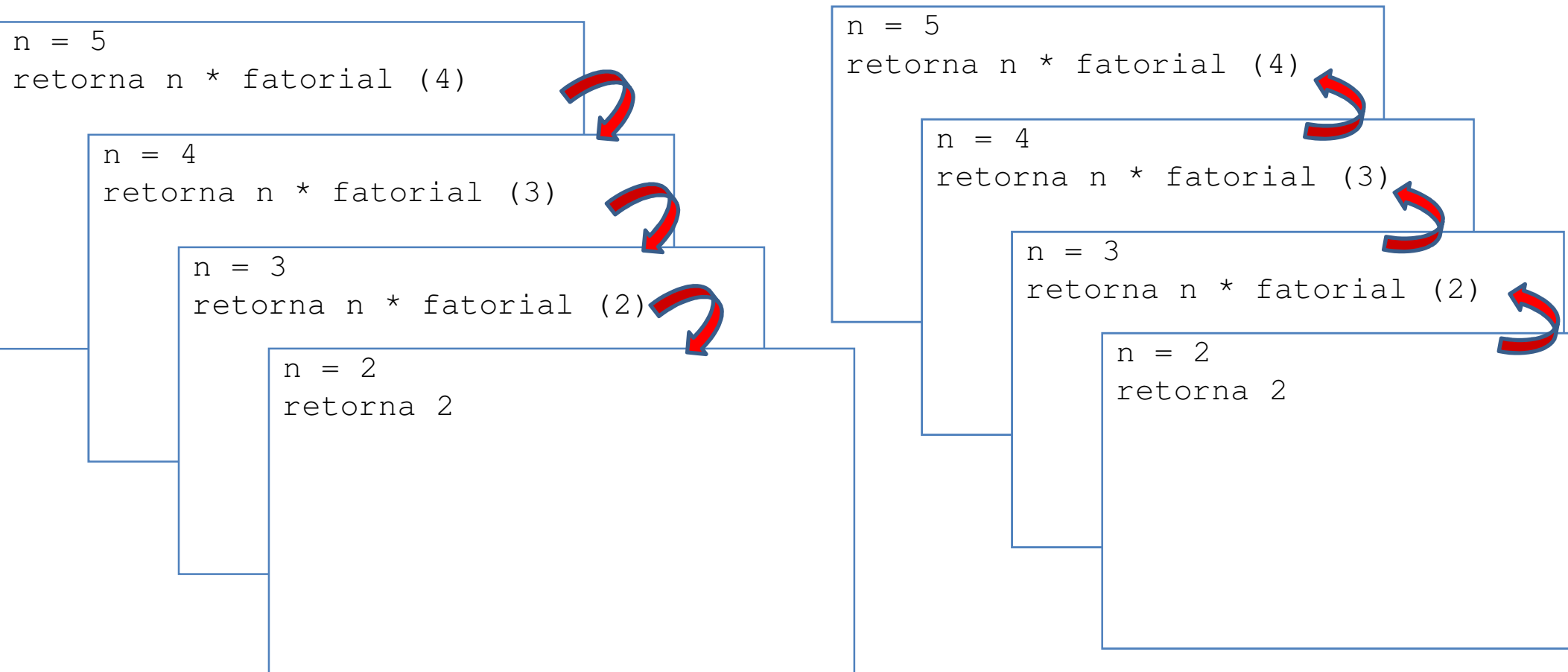
Tentativa e erro - recursividade

- O que você precisa fazer para que a recursividade registre e remova os dados?



Tentativa e erro - recursividade

- O que você precisa fazer para que a recursividade registre e remova os dados?
- **NADA!!! SIMPLEMENTE USAR RECURSIVIDADE!**



Exercício – exemplo Labirinto

Como tornar este algoritmo recursivo?

```
Enquanto não chegar ao destino e houver movimentos possíveis
  Enquanto movimento for válido
    Move direita
    Se fracasso
      Apaga último
      Volta posição anterior
    Move acima
  Fim Se
Fim Enquanto
Fim Enquanto
Se acabou movimentos possíveis
  Imprime "Impossível chegar ao destino"
Senão
  Imprime "Chegou ao destino"
Fim Se
```

Exercício – exemplo Labirinto

Como tornar este algoritmo recursivo?

```
Enquanto não chegar ao destino e houver movimentos possíveis
    Enquanto movimento for válido
        Move (direita)
    Fim Enquanto
Fim Enquanto
Se acabou movimentos possíveis
    Imprime "Impossível chegar ao destino"
Senão
    Imprime "Chegou ao destino"
Fim Se
```

```
Move (parâmetro paraOndeMover)
Se sucesso
    retorna Moveu
Senão
    Apaga último
    Volta posição anterior
    Move(próximaPosição)
Fim Se
```

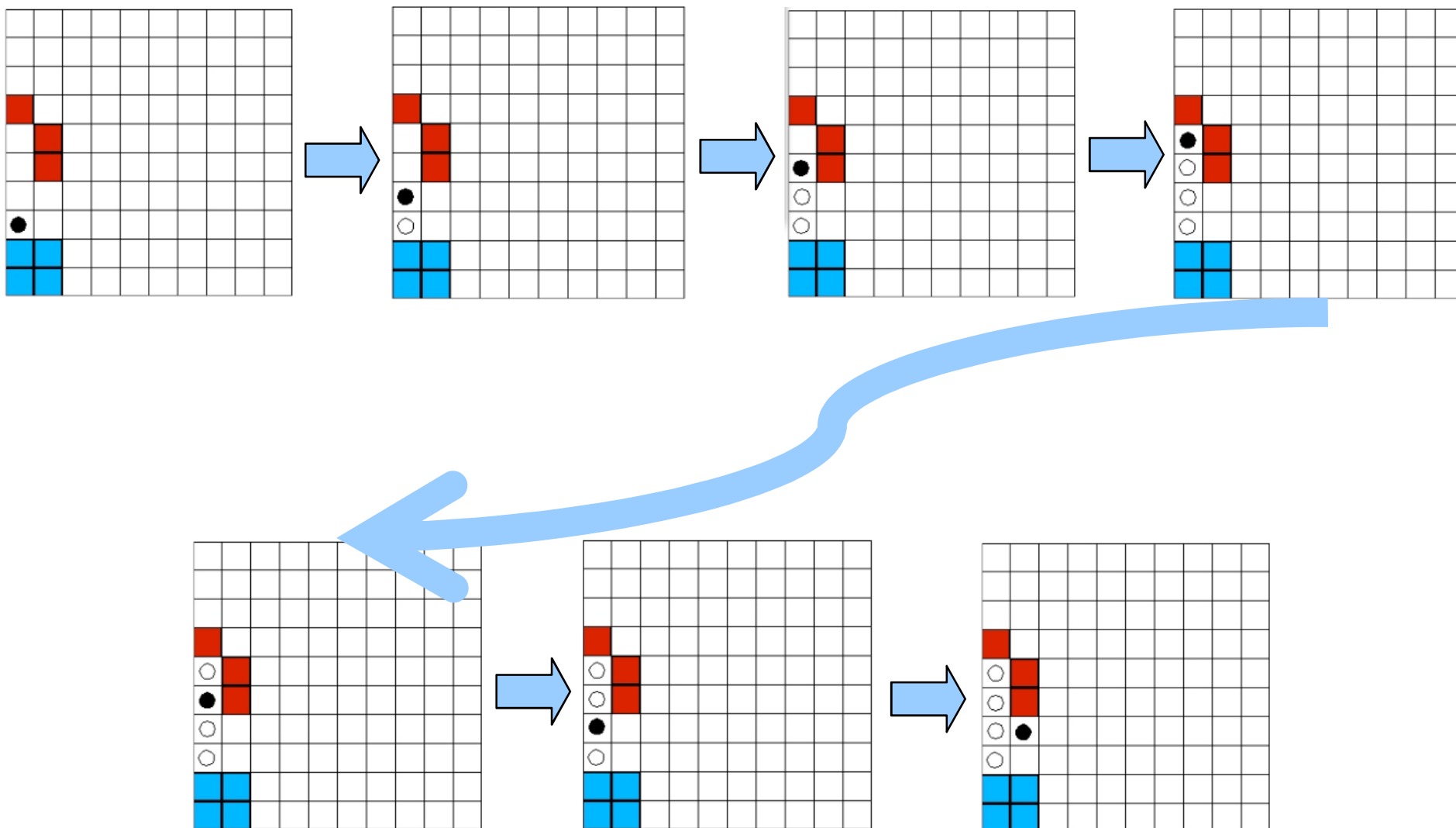
Algun lugar (estrutura de dados, por exemplo, tem que definir qual a próxima posição).

Exemplo array sonde cada posição indica o deslocamento em x e y. Cada iteração chama o método com o parâmetro (x[i+1],y[i+1])
deslocamentoX [] = {0,-1,0,1}
deslocamentoY [] = {1,0,-1,0}

Tentativa e erro - exemplos

- Você deseja buscar um elemento qualquer dentro de uma região (representada por uma matriz):
 - existem posições inacessíveis;
 - você desconhece a posição do elemento buscado;
 - em cada iteração, você deve decidir para onde ir na matriz;
 - você não tem informação suficiente para escolher a direção;
 - cada escolha leva a outro conjunto de escolhas;
 - uma ou mais sequência de passos pode ser a solução.

Tentativa e erro - exemplos



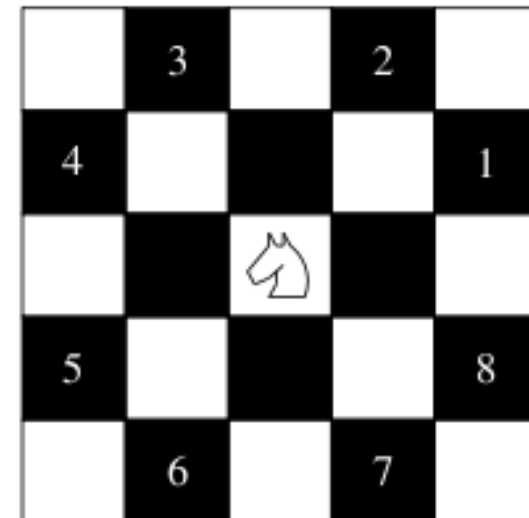
Tentativa e erro – exemplo

Passeio do cavalo no tabuleiro de xadrez (livro Ziviani)

tabuleiro com $n \times n$ posições

cavalo se movimenta segundo as regras do xadrez (próximo passo: soma 1 casa em x e 2 em y – ou 2 casas em x e 1 em y)

A partir de uma posição inicial, o problema consiste em encontrar, se existir um passeio do cavalo com n^2-1 movimentos, visitando todos os pontos do tabuleiro uma única vez



Tentativa e erro – exemplo

Passeio do cavalo no tabuleiro de xadrez (livro Ziviani)

Uma solução para um tabuleiro de tamanho 8 x 8

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

Passeio do Cavalo

```
import java.util.Calendar;  
import java.util.Date;
```

```
public class PasseioCavalo
```

```
{  
    final int[] dx = { 2, 1, -1, -2, -2, -1, 1, 2 }; // posições possíveis  
                                                    // de movimento na coordenada x  
    final int[] dy = { 1, 2, 2, 1, -1, -2, -2, -1 }; // posições possíveis  
                                                    //de movimento na coordenada y  
  
    final int _num; // n número de linhas e colunas  
    final int _numSqr; // n ao quadrado  
    int[][] table; // matriz que representa tabuleiro  
    long totalMovimentos = 0;  
    long numDeOperacoes = 0;
```

```
public PasseioCavalo(int num) //construtor
```

```
{  
    _num = num;  
    _numSqr = num * num;  
    table = new int[num][num];  
}
```

Passeio do Cavalo

```
boolean ehAceitavel(int x, int y)
    //devolve true se todas as condições forem válidas
{
    boolean result = (x >= 0 && x <= _num - 1);
        //verificar número da coordenada x
    result = result && (y >= 0 && y <= _num - 1);
        //verificar número da coordenada y
    result = result && (table[x][y] == 0);
        //posição da tabela está válida
    return result;
}
```

Passeio do Cavalo

```
boolean tentaMovimento(int i, int x, int y) { // tenta fazer movimentos
    boolean termina = (i > _numSqr);
        // termina quando i for maior que n ao quadrado

    int k = 0;
    int u, v;          // u,v: posicao de destino - x, y: posicao atual
    while (!termina && k < 8) { // 8 eh o numero de movimentos
        // teoricamente possiveis

        numDeOperacoes++;
        u = x + dx[k];
        v = y + dy[k];
        if (ehAceitavel(u, v)) {
            table[u][v] = i;
            totalMovimentos++;
            termina = tentaMovimento(i + 1, u, v); // tenta outro movimento
            if (!termina) {
                totalMovimentos++;
                table[u][v] = 0; // não sucedido. Descarta movimento
            }
        }
        k = k + 1;
    }
    return termina;
}
```



```
void resolverPasseio(int x, int y) // chama procedimento que faz movimento
{
    numDeOperacoes=0;
    totalMovimentos=0;
    table[x][y] = 1;
    Date inicio = Calendar.getInstance().getTime();
    System.out.println("Inicio = " + inicio);
    boolean problemaResolvido = tentaMovimento(2, x, y);
    Date fim = Calendar.getInstance().getTime();
    System.out.println("Fim      = " + fim);
    if (problemaResolvido) {
        System.out.println("Passeio resolvido em
                           (" + ((fim.getTime() - inicio.getTime())/1000.0) + " s)");
        for (int i = 0; i < _num; i++) {
            for (int j = 0; j < _num; j++) {
                if (table[i][j]<100) System.out.print(" ");
                // apenas para deixar a impressao mais organizada
                if (table[i][j]<10) System.out.print(" ");
                // apenas para deixar a impressao mais organizada
                System.out.print(table[i][j] + " ");
            }
            System.out.println();
        }
    } else {
        System.out.println("Passeio impossivel de ser realizado.");
    }
    System.out.println("Numero total de movimentos do cavalo: " + totalMovimentos);
    System.out.println("Numero total de operacoes do algoritmo: " + numDeOperacoes);
}
```

Passeio do Cavalo

```
public static void main(String[] args) {  
    /*  Uso as seguintes linhas (comentadas abaixo) se  
        nao quiser utilizar argumentos  
args = new String[3];  
args[0] = "8";  
args[1] = "0";  
args[2] = "0";  
*/  
if (args.length >= 3){  
    int n = Integer.parseInt(args[0]);  
    int x = Integer.parseInt(args[1]);  
    int y = Integer.parseInt(args[2]);  
    new KnightsTour(n).resolverPasseio(x, y);  
}else{  
    System.out.println("Voce precisa passar 3 numeros como  
    parâmetros de entrada: ordem_do_tabuleiro x0 y0");  
}  
}
```


Referências

- 𠄎 Nívio Ziviani. Projeto de Algoritmos com implementações em C e Pascal. Editora Thomson, 2a. Edição, 2004 (texto base)
- 𠄎 Notas de aula – Prof. Delano Beder – EACH-USP
- 𠄎 Notas de aula – Prof. Norton Roman – EACH-USP
- 𠄎 Notas de aula – Prof. José Perez – EACH-USP

Paradigmas de Projeto de Algoritmos

Tentativa e Erro (Backtracking)

Professora:
Fátima L. S. Nunes