

# Aula 04 – Técnicas de Desenvolvimento de Algoritmos: Tentativa e Erro (Backtracking)

Norton Trevisan Roman  
norton@usp.br

23 de agosto de 2018

# Tentativa e Erro

- Suponha que você tem que tomar uma série de decisões dentre várias possibilidades, onde

# Tentativa e Erro

- Suponha que você tem que tomar uma série de decisões dentre várias possibilidades, onde
  - Você não tem informação suficiente para saber o que escolher

# Tentativa e Erro

- Suponha que você tem que tomar uma série de decisões dentre várias possibilidades, onde
  - Você não tem informação suficiente para saber o que escolher
  - Cada decisão leva a um novo conjunto de escolhas

# Tentativa e Erro

- Suponha que você tem que tomar uma série de decisões dentre várias possibilidades, onde
  - Você não tem informação suficiente para saber o que escolher
  - Cada decisão leva a um novo conjunto de escolhas
  - Alguma sequência de escolhas (possivelmente mais que uma) pode ser a solução para o problema

# Tentativa e Erro

- Suponha que você tem que tomar uma série de decisões dentre várias possibilidades, onde
  - Você não tem informação suficiente para saber o que escolher
  - Cada decisão leva a um novo conjunto de escolhas
  - Alguma sequência de escolhas (possivelmente mais que uma) pode ser a solução para o problema
- Tentativa e erro é um modo metódico de tentar várias sequências de decisões, até encontrar uma que funcione

# Tentativa e Erro

- Técnica de solução de problemas
  - Usada quando se quer achar soluções para problemas para os quais não se conhece uma regra fixa de computação
  - Basicamente, tentamos todas as alternativas possíveis

# Tentativa e Erro

- Técnica de solução de problemas
  - Usada quando se quer achar soluções para problemas para os quais não se conhece uma regra fixa de computação
  - Basicamente, tentamos todas as alternativas possíveis
- Passos
  - Escolher uma operação plausível;
  - Executar a operação com os dados;
  - Se a meta não foi alcançada, repita o processo até que se atinja a meta ou se evidencie a insolubilidade do problema.



# Tentativa e Erro

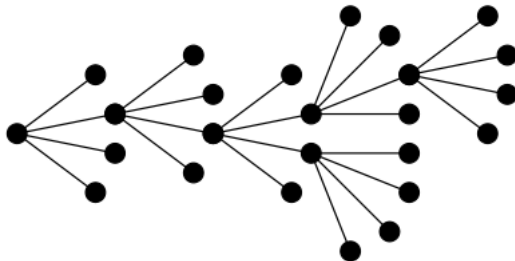
- Tentativa e erro é uma técnica que aproveita bem a recursividade
  - A recursividade pode ser usada para resolver problemas cuja solução é do tipo tentar todas as alternativas possíveis

# Tentativa e Erro

- Tentativa e erro é uma técnica que aproveita bem a recursividade
  - A recursividade pode ser usada para resolver problemas cuja solução é do tipo tentar todas as alternativas possíveis
- A ideia para algoritmos tentativa e erro é decompor o processo em um número finito de sub-tarefas parciais (expressas de forma recursiva)
  - E então explorá-las exaustivamente

# Tentativa e Erro

- A construção de uma solução é obtida através de tentativas (ou pesquisas) que gradualmente constroem e percorrem uma árvore de sub-tarefas.



# Tentativa e Erro

## Funcionamento:

- Passos em direção à solução final são tentados e registrados (em uma estrutura de dados)
  - Caso esses passos tomados não levem à solução final, eles podem ser retirados e apagados do registro.
- 
- A busca na árvore de soluções pode crescer rapidamente (até mesmo exponencialmente)
    - Pode ser necessário usar algoritmos aproximados ou heurísticas, que não garantem a solução ótima, mas são rápidas

## Exploramos cada possibilidade como segue:

- Se a possibilidade é a resposta, retorne “sucesso”
- Se a possibilidade não for a resposta, e não houver outra a ser testada a partir dela, retorne “falha”
- Para cada possibilidade, a partir da atual:
  - Explore a nova possibilidade (recursivo)
  - Se encontrou a resposta, retorne “sucesso”
- Esgotadas as possibilidades, retorne “falha”

# Tentativa e Erro

## Exemplo: Caminho em Labirinto

- Dado um labirinto, encontre um caminho da entrada à saída



# Tentativa e Erro

## Exemplo: Caminho em Labirinto

- Dado um labirinto, encontre um caminho da entrada à saída
- Em cada interseção, você tem que decidir se:
  - Segue direto
  - Vai à esquerda
  - Vai à direita



# Tentativa e Erro

## Exemplo: Caminho em Labirinto

- Só que... você não tem informação suficiente para escolher corretamente





# Tentativa e Erro

## Exemplo: Caminho em Labirinto

- Só que... você não tem informação suficiente para escolher corretamente
- Cada escolha leva a outro conjunto de escolhas



# Tentativa e Erro

## Exemplo: Caminho em Labirinto

- Só que... você não tem informação suficiente para escolher corretamente
- Cada escolha leva a outro conjunto de escolhas
- Uma ou mais sequência de escolhas pode ser a solução...



# Tentativa e Erro

## Exemplo: Caminho em Labirinto

- Só que... você não tem informação suficiente para escolher corretamente
- Cada escolha leva a outro conjunto de escolhas
- Uma ou mais sequência de escolhas pode ser a solução... Ou não!



# Tentativa e Erro

## Exemplo: Caminho em Labirinto

- E a solução?



# Tentativa e Erro

## Exemplo: Caminho em Labirinto

- E a solução?
  - Escolha uma metodologia para fazer tentativas



# Tentativa e Erro

## Exemplo: Caminho em Labirinto

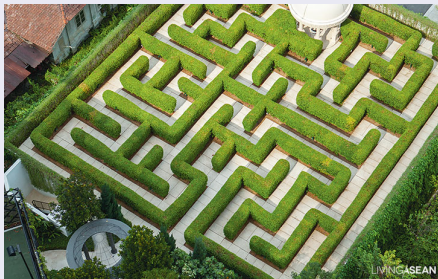
- E a solução?
  - Escolha uma metodologia para fazer tentativas
  - Ex: primeiro tentar à direita, depois à frente, e então à esquerda



# Tentativa e Erro

## Exemplo: Caminho em Labirinto

- E a solução?
  - Escolha uma metodologia para fazer tentativas
  - Ex: primeiro tentar à direita, depois à frente, e então à esquerda
  - Então, a cada decisão, siga essa ordem de escolhas



# Tentativa e Erro

## Exemplo: Caminho em Labirinto

- E a solução?
  - Escolha uma metodologia para fazer tentativas
  - Ex: primeiro tentar à direita, depois à frente, e então à esquerda
  - Então, a cada decisão, siga essa ordem de escolhas
  - Se todas as escolhas em um ponto se esgotaram, volte ao anterior, e escolha uma alternativa ainda não explorada (seguindo a metodologia de escolha) a partir desse ponto





## Exemplo: Coloração de Mapas

- Você deseja colorir um mapa com no máximo 4 cores: Vermelho, amarelo, verde e azul

## Exemplo: Coloração de Mapas

- Você deseja colorir um mapa com no máximo 4 cores: Vermelho, amarelo, verde e azul
- Países adjacentes devem ter cores diferentes

# Tentativa e Erro

## Exemplo: Coloração de Mapas

- Você deseja colorir um mapa com no máximo 4 cores: Vermelho, amarelo, verde e azul
- Países adjacentes devem ter cores diferentes



# Tentativa e Erro

## Exemplo: Coloração de Mapas

- Em cada iteração, você deve decidir de que cor pintar um país, dadas as cores já atribuídas aos vizinhos imediatos



# Tentativa e Erro

## Exemplo: Coloração de Mapas

- Em cada iteração, você deve decidir de que cor pintar um país, dadas as cores já atribuídas aos vizinhos imediatos
- Você não tem informação suficiente para escolher as cores (só conhece os vizinhos imediatos)



# Tentativa e Erro

## Exemplo: Coloração de Mapas

- Cada escolha leva a outro conjunto de escolhas



# Tentativa e Erro

## Exemplo: Coloração de Mapas

- Cada escolha leva a outro conjunto de escolhas
- Uma ou mais sequência de passos pode ser a solução



# Tentativa e Erro

## Exemplo: Coloração de Mapas

- E a solução?





# Tentativa e Erro

## Exemplo: Coloração de Mapas

- E a solução?
  1. Pinte um país inicial



# Tentativa e Erro

## Exemplo: Coloração de Mapas

- E a solução?
  1. Pinte um país inicial
  2. Escolha cores para seus vizinhos, obedecendo à regra



# Tentativa e Erro

## Exemplo: Coloração de Mapas

- E a solução?
  1. Pinte um país inicial
  2. Escolha cores para seus vizinhos, obedecendo à regra
  3. Escolha um vizinho (seguindo uma ordem pré-definida)



# Tentativa e Erro

## Exemplo: Coloração de Mapas

- E a solução?
  1. Pinte um país inicial
  2. Escolha cores para seus vizinhos, obedecendo à regra
  3. Escolha um vizinho (seguindo uma ordem pré-definida)
  4. Repita o procedimento do passo 2 a partir desse vizinho



# Tentativa e Erro

## Exemplo: Coloração de Mapas

5. Se não for possível colorir alguém, volte à escolha anterior, e troque a cor do país no centro desse passo



# Tentativa e Erro

## Exemplo: Coloração de Mapas

5. Se não for possível colorir alguém, volte à escolha anterior, e troque a cor do país no centro desse passo
- Note que, se não for possível trocar a cor deste, voltamos ao anterior a ele



# Tentativa e Erro

## Exemplo: Coloração de Mapas

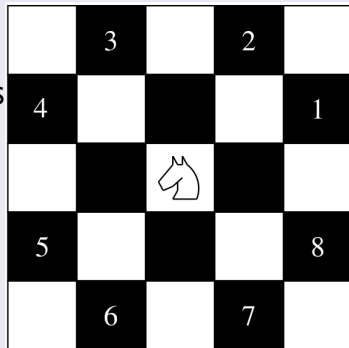
5. Se não for possível colorir alguém, volte à escolha anterior, e troque a cor do país no centro desse passo
- Note que, se não for possível trocar a cor deste, voltamos ao anterior a ele
  - Vamos voltando e revendo nossas escolhas até acertar



# Tentativa e Erro

## Exemplo: Passeio do Cavalo

- Dado um tabuleiro com  $n \times n$  posições, o cavalo se movimenta segundo as regras do xadrez:

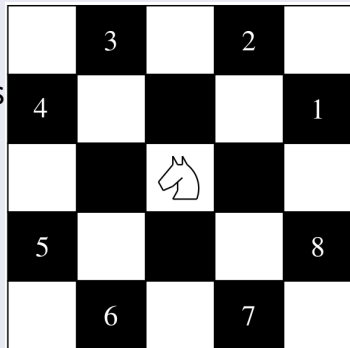




# Tentativa e Erro

## Exemplo: Passeio do Cavalo


- Dado um tabuleiro com  $n \times n$  posições, o cavalo se movimenta segundo as regras do xadrez:
- Problema: a partir de  $(x_0, y_0)$ , encontrar, se existir, um passeio do cavalo que visite todos os pontos do tabuleiro uma única vez



# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Representação


- Tabuleiro: matriz  $t$  ( $n \times n$ ) de inteiros
  - $t[x][y] = 0 \rightarrow (x, y)$  não visitado
  - $t[x][y] = i \rightarrow (x, y)$  visitado no  $i$ -ésimo movimento

|   |   |   |   |   |
|---|---|---|---|---|
|   | 3 |   | 2 |   |
| 4 |   |   |   | 1 |
|   |   |  |   |   |
| 5 |   |   |   | 8 |
|   | 6 |   | 7 |   |

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Representação

- Tabuleiro: matriz  $t$  ( $n \times n$ ) de inteiros
  - $t[x][y] = 0 \rightarrow (x, y)$  não visitado
  - $t[x][y] = i \rightarrow (x, y)$  visitado no  $i$ -ésimo movimento

|   |   |   |   |   |
|---|---|---|---|---|
|   | 3 |   | 2 |   |
| 4 |   |   |   | 1 |
|   |   |  |   |   |
| 5 |   |   |   | 8 |
|   | 6 |   | 7 |   |

- Note que o valor das células guarda o histórico dos movimentos

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Algoritmo

```
void tenta() {  
    inicializa seleção de movimentos  
    ENQUANTO (movimento não bem sucedido E existem candidatos a  
                                                    movimento) {  
        seleciona próximo candidato ao movimento  
        SE (aceitável) {  
            registra movimento  
            SE (tabuleiro não está cheio) {  
                tenta novo movimento; // Chamada recursiva a tenta()  
                SE (não sucedido) apaga registro anterior  
            }  
        }  
    }  
}
```

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Algoritmo

```
void tenta() {  
    inicializa seleção de movimentos  
    ENQUANTO (movimento não bem sucedido E existem candidatos a  
                                                    movimento) {  
        seleciona próximo candidato ao movimento  
        SE (aceitável) {  
            registra movimento ← abastece t[i][j]  
            SE (tabuleiro não está cheio) {  
                tenta novo movimento; // Chamada recursiva a tenta()  
                SE (não sucedido) apaga registro anterior  
            }  
        }  
    }  
}
```

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Algoritmo

```
void tenta() {  
    inicializa seleção de movimentos  
    ENQUANTO (movimento não bem sucedido E existem candidatos a  
                                                    movimento) {  
        seleciona próximo candidato ao movimento  
        SE (aceitável) {  
            registra movimento  
            SE (tabuleiro não está cheio) {  
                tenta novo movimento; // Chamada recursiva a tenta()  
                SE (não sucedido) ← apaga registro anterior  
            }  
        }  
    }  
}
```

O movimento anterior não leva à solução. Deve voltar (backtracking)

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Algoritmo

```
void tenta() {  
    inicializa seleção de movimentos  
    ENQUANTO (movimento não bem sucedido E existem candidatos a  
                                                    movimento) {  
        seleciona próximo candidato ao movimento  
        SE (aceitável) {  
            registra movimento  
            SE (tabuleiro não está cheio) {  
                tenta novo movimento; // Chamada recursiva a tenta()  
                SE (não sucedido) apaga registro anterior  
            }  
        }  
    }  
}
```

Ao final do método, o tabuleiro conterá a resposta

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
class Cavalo {
    int[] dx = { 2, 1,-1,-2,-2,-1, 1, 2};
    int[] dy = { 1, 2, 2, 1,-1,-2,-2,-1};
    int num, numQuad;
    int[] [] tab;

    Cavalo(int num) {
        this.num = num;
        this.numQuad = num * num;
        this.tab = new int[num][num];
    }

    boolean aceitavel(int x, int y) {
        return(x >= 0 && x <= num-1 && y >= 0 && y <= num-1 &&
            tab[x][y] == 0);
    }
}
```



# Tentativa e Erro

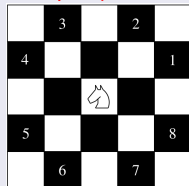
## Exemplo: Passeio do Cavalo – Código

```
class Cavalo {  
    int[] dx = { 2, 1,-1,-2,-2,-1, 1, 2};  
    int[] dy = { 1, 2, 2, 1,-1,-2,-2,-1};  
    int num, numQuad;  
    int[] [] tab;
```

```
    Cavalo(int num) {  
        this.num = num;  
        this.numQuad = num * num;  
        this.tab = new int[num][num];  
    }
```


```
    boolean aceitavel(int x, int y) {  
        return(x >= 0 && x <= num-1 && y >= 0 && y <= num-1 &&  
               tab[x][y] == 0);  
    }
```

Deslocamentos possíveis a partir da posição do cavalo



# Tentativa e Erro


## Exemplo: Passeio do Cavalo – Código

```
class Cavalo {  
    int[] dx = { 2, 1,-1,-2,-2,-1, 1, 2};  
    int[] dy = { 1, 2, 2, 1,-1,-2,-2,-1};  
    int num, numQuad;   
    int[] [] tab;  
  
    Cavalo(int num) {  
        this.num = num;  
        this.numQuad = num * num;  
        this.tab = new int[num][num];  
    }  
  
    boolean aceitavel(int x, int y) {  
        return(x >= 0 && x <= num-1 && y >= 0 && y <= num-1 &&  
            tab[x][y] == 0);  
    }  
}
```

Número de quadros em uma linha e total de quadros no tabuleiro

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
class Cavalo {  
    int[] dx = { 2, 1,-1,-2,-2,-1, 1, 2};  
    int[] dy = { 1, 2, 2, 1,-1,-2,-2,-1};  
    int num, numQuad;  
    int[] [] tab;   
  
    Cavalo(int num) {  
        this.num = num;  
        this.numQuad = num * num;  
        this.tab = new int[num][num];  
    }  
  
    boolean aceitavel(int x, int y) {  
        return(x >= 0 && x <= num-1 && y >= 0 && y <= num-1 &&  
            tab[x][y] == 0);  
    }  
}
```


O tabuleiro

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
class Cavalo {  
    int[] dx = { 2, 1,-1,-2,-2,-1, 1, 2};  
    int[] dy = { 1, 2, 2, 1,-1,-2,-2,-1};  
    int num, numQuad;  
    int[] [] tab;  
  
    Cavalo(int num) {  
        this.num = num;  
        this.numQuad = num * num;  
        this.tab = new int[num][num];  
    }  
  
    boolean aceitavel(int x, int y) {  
        return(x >= 0 && x <= num-1 && y >= 0 && y <= num-1 &&  
            tab[x][y] == 0);  
    }  
}
```

Determina se (x,y) é uma coordenada válida



# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

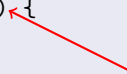
```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y){  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

Tenta mover o cavalo  
a partir de (x, y) no  
i-ésimo movimento

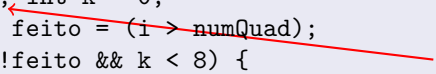


# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

Coordenadas de  
destino do movimento



# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

Quantos dos possíveis  
movimentos a partir  
de  $(x,y)$  já tentou



# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

Diz se pode parar  
(caso base: já marcou  
todo o tabuleiro)

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```


Enquanto não terminou de marcar o tabuleiro todo, e ainda há movimentos possíveis

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

Calcula a coordenada de destino



# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

Se a coordenada for válida



# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

Move o cavalo a ela




# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

E tenta novo movimento  
a partir dessa coordenada

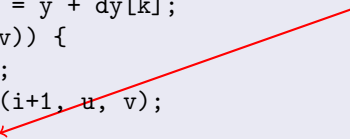


# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

Se a tentativa for infrutífera



# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

Descarta o movimento



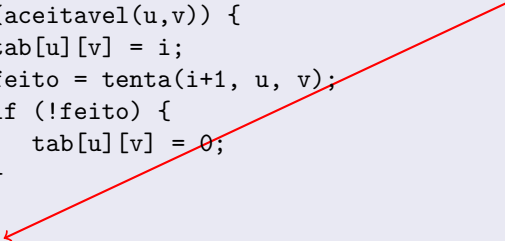


# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

passa ao próximo  
movimento possível

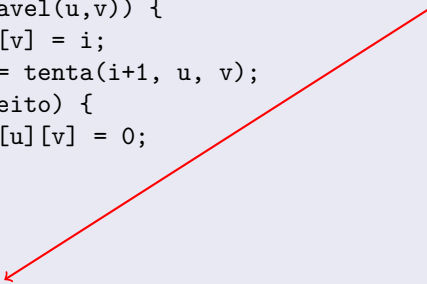


# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
boolean tenta(int i, int x, int y) {  
    int u,v; int k = 0;  
    boolean feito = (i > numQuad);  
    while (!feito && k < 8) {  
        u = x + dx[k]; v = y + dy[k];  
        if (aceitavel(u,v)) {  
            tab[u][v] = i;  
            feito = tenta(i+1, u, v);  
            if (!feito) {  
                tab[u][v] = 0;  
            }  
        }  
        k++;  
    }  
    return feito;  
}
```

define se o caminho a  
partir de (x,y) teve  
ou não teve sucesso



# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
void imprime() {  
    for (int i = 0; i < num; i++) {  
        for (int j = 0; j < num; j++)  
            System.out.print(tab[i][j] + "\t");  
        System.out.println("\n");  
    }  
}  
  
void passeia(int x, int y) {  
    tab[x][y] = 1;  
    boolean feito = tenta(2, x, y);  
    if (feito)  
        imprime();  
    else  
        System.out.println("Não há passeio possível");  
}
```

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
void imprime() {  
    for (int i = 0; i < num; i++) {  
        for (int j = 0; j < num; j++)  
            System.out.print(tab[i][j] + "\t");  
        System.out.println("\n");  
    }  
}
```

```
void passeia(int x, int y) {  
    tab[x][y] = 1;  
    boolean feito = tenta(2, x, y);  
    if (feito)  
        imprime();  
    else  
        System.out.println("Não há passeio possível");  
}
```

Inicia o passeio em (x,y)

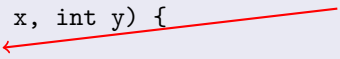
# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
void imprime() {  
    for (int i = 0; i < num; i++) {  
        for (int j = 0; j < num; j++)  
            System.out.print(tab[i][j] + "\t");  
        System.out.println("\n");  
    }  
}
```

```
void passeia(int x, int y) {  
    tab[x][y] = 1;  
    boolean feito = tenta(2, x, y);  
    if (feito)  
        imprime();  
    else  
        System.out.println("Não há passeio possível");  
}
```

Marca a posição de  
início como visitada




# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
void imprime() {  
    for (int i = 0; i < num; i++) {  
        for (int j = 0; j < num; j++)  
            System.out.print(tab[i][j] + "\t");  
        System.out.println("\n");  
    }  
}
```

```
void passeia(int x, int y) {  
    tab[x][y] = 1;  
    boolean feito = tenta(2, x, y);  
    if (feito)  
        imprime();  
    else  
        System.out.println("Não há passeio possível");  
}
```

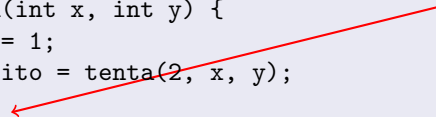
E verifica os caminhos a partir dela



# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
void imprime() {  
    for (int i = 0; i < num; i++) {  
        for (int j = 0; j < num; j++)  
            System.out.print(tab[i][j] + "\t");  
        System.out.println("\n");  
    }  
}
```

```
void passeia(int x, int y) {  
    tab[x][y] = 1;  
    boolean feito = tenta(2, x, y);  
    if (feito)   
        imprime();  
    else  
        System.out.println("Não há passeio possível");  
}
```

Se achou a resposta, mostra

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

```
void imprime() {  
    for (int i = 0; i < num; i++) {  
        for (int j = 0; j < num; j++)  
            System.out.print(tab[i][j] + "\t");  
        System.out.println("\n");  
    }  
}
```

```
void passeia(int x, int y) {  
    tab[x][y] = 1;  
    boolean feito = tenta(2, x, y);  
    if (feito)  
        imprime();  
    else  
        System.out.println("Não há passeio possível");  
}
```

Senão, avisa que não  
há resposta possível





## Exemplo: Passeio do Cavalo – Código

```
public static void main(String[] args) {  
    int n = Integer.parseInt(args[0]);  
    int x = Integer.parseInt(args[1]);  
    int y = Integer.parseInt(args[2]);  
    new Cavalo(n).passeia(x, y);  
}
```

# Tentativa e Erro

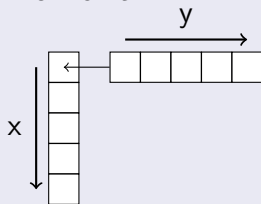
## Exemplo: Passeio do Cavalo – Representação

- Está correto  $t[x][y] \rightarrow (x, y)$ ?

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Representação

- Está correto  $t[x][y] \rightarrow (x, y)$ ?
- Memória:

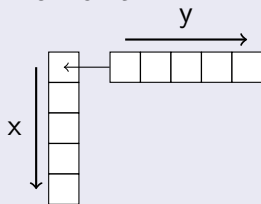


# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Representação

- Está correto  $t[x][y] \rightarrow (x, y)$ ?

- Memória:



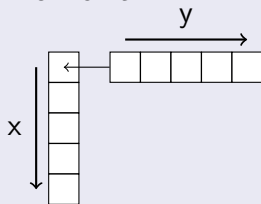
- Ao somarmos algum  $dx$  a  $t[x][y]$ , estaremos, na verdade, “decendo” na vertical

# Tentativa e Erro

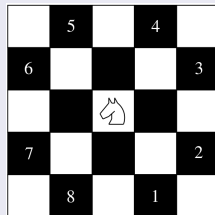
## Exemplo: Passeio do Cavalo – Representação

- Está correto  $t[x][y] \rightarrow (x, y)$ ?

- Memória:



- Então...



- Ao somarmos algum  $dx$  a  $t[x][y]$ , estaremos, na verdade, “decendo” na vertical

## Exemplo: Passeio do Cavalo – Código


- Se chamado com  
`java Cavalo 8 0 0`  
temos o seguinte  
resultado:

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 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 |

# Tentativa e Erro

## Exemplo: Passeio do Cavalo – Código

- Se chamado com  
`java Cavalo 8 0 0`  
temos o seguinte  
resultado:

|   |   |   |   |   |
|---|---|---|---|---|
|   | 5 |   | 4 |   |
| 6 |   |   |   | 3 |
|   |   |  |   |   |
| 7 |   |   |   | 2 |
|   | 8 |   | 1 |   |

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 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 |

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro



# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

n: 3

retorna: ?

Ao chamarmos fatorial(3),  
o método é empilhado

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

n: 3

retorna: ?

Não é o caso base

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

|      |            |
|------|------------|
| n: 2 | retorna: ? |
| n: 3 | retorna: ? |

Então nova chamada  
é feita e empilhada

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

|      |            |
|------|------------|
| n: 2 | retorna: ? |
| n: 3 | retorna: ? |

Não é o caso base

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

|      |            |
|------|------------|
| n: 1 | retorna: ? |
| n: 2 | retorna: ? |
| n: 3 | retorna: ? |

Então nova chamada  
é feita e empilhada

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

|      |            |
|------|------------|
| n: 1 | retorna: ? |
| n: 2 | retorna: ? |
| n: 3 | retorna: ? |

Não é o caso base

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

|      |            |
|------|------------|
| n: 0 | retorna: ? |
| n: 1 | retorna: ? |
| n: 2 | retorna: ? |
| n: 3 | retorna: ? |

Então nova chamada  
é feita e empilhada



# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

|      |            |
|------|------------|
| n: 0 | retorna: 1 |
| n: 1 | retorna: ? |
| n: 2 | retorna: ? |
| n: 3 | retorna: ? |

É o caso base!

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

|      |            |
|------|------------|
| n: 1 | retorna: ? |
| n: 2 | retorna: ? |
| n: 3 | retorna: ? |

Desempilha, retornando 1.  
E assim “volta” à porção de  
memória no passo anterior  
da recursão → backtracking

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

|      |            |
|------|------------|
| n: 1 | retorna: 1 |
| n: 2 | retorna: ? |
| n: 3 | retorna: ? |

Faz  $1 * 1$

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

|      |            |
|------|------------|
| n: 2 | retorna: ? |
| n: 3 | retorna: ? |

Desempilha, retornando 1 → backtracking

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

|      |            |
|------|------------|
| n: 2 | retorna: 2 |
| n: 3 | retorna: ? |

Faz  $2 * 1$

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

n: 3

retorna: ?

Desempilha, retornando 2 → backtracking

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

n: 3

retorna: 6

Faz  $3 * 2$

# Recursão com Tentativa e Erro

Recursão é a maneira mais natural de se implementar tentativa e erro

Vamos relembrar nosso  
fatorial(3):

```
long fatorial(long n) {  
    if (n==0) return(1);  
    return(n *  
           fatorial(n-1));  
}
```

Desempilha, retornando 6



# Recursão com Tentativa e Erro

- Se conseguirmos codificar o problema de modo a que:
  - Cada nova decisão seja uma chamada recursiva
  - Cada backtracking corresponda a voltar de uma chamada recursiva
- Então a solução do problema, se feita recursivamente, naturalmente gerenciará o mecanismo de tentativa e erro

# Recursão com Tentativa e Erro

E se não pudermos/quisermos usar recursão?

# Recursão com Tentativa e Erro

E se não pudermos/quisermos usar recursão?

- Precisaremos de um mecanismo que nos permita:

# Recursão com Tentativa e Erro

## E se não pudermos/quisermos usar recursão?

- Precisaremos de um mecanismo que nos permita:
  - Registrar cada decisão tomada

# Recursão com Tentativa e Erro

## E se não pudermos/quisermos usar recursão?

- Precisaremos de um mecanismo que nos permita:
  - Registrar cada decisão tomada
  - Voltar à decisão imediatamente anterior a cada decisão tomada, para que possamos fazer o backtracking em caso de falha
  - Ou seja, precisamos registrar também a ordem em que foram tomadas

# Recursão com Tentativa e Erro

## E se não pudermos/quisermos usar recursão?

- Precisaremos de um mecanismo que nos permita:
  - Registrar cada decisão tomada
  - Voltar à decisão imediatamente anterior a cada decisão tomada, para que possamos fazer o backtracking em caso de falha
    - Ou seja, precisamos registrar também a ordem em que foram tomadas
- Um mecanismo assim é fornecido por uma pilha:

# Recursão com Tentativa e Erro

## E se não pudermos/quisermos usar recursão?

- Precisaremos de um mecanismo que nos permita:
  - Registrar cada decisão tomada
  - Voltar à decisão imediatamente anterior a cada decisão tomada, para que possamos fazer o backtracking em caso de falha
    - Ou seja, precisamos registrar também a ordem em que foram tomadas
- Um mecanismo assim é fornecido por uma pilha:
  - Empilhamos cada decisão tomada

# Recursão com Tentativa e Erro

## E se não pudermos/quisermos usar recursão?

- Precisaremos de um mecanismo que nos permita:
  - Registrar cada decisão tomada
  - Voltar à decisão imediatamente anterior a cada decisão tomada, para que possamos fazer o backtracking em caso de falha
    - Ou seja, precisamos registrar também a ordem em que foram tomadas
- Um mecanismo assim é fornecido por uma pilha:
  - Empilhamos cada decisão tomada
  - Em caso de falha, desempilhamos → voltamos à decisão imediatamente anterior



# Recursão com Tentativa e Erro

E se não pudermos/quisermos usar recursão?

- Note que isso é exatamente o que a pilha de recursão faz

# Recursão com Tentativa e Erro

E se não pudermos/quisermos usar recursão?

- Note que isso é exatamente o que a pilha de recursão faz

decisão 1

Registramos cada decisão tomada

# Recursão com Tentativa e Erro

## E se não pudermos/quisermos usar recursão?

- Note que isso é exatamente o que a pilha de recursão faz

|           |
|-----------|
| decisão 2 |
| decisão 1 |

Registramos cada decisão tomada

# Recursão com Tentativa e Erro

## E se não pudermos/quisermos usar recursão?

- Note que isso é exatamente o que a pilha de recursão faz

|           |
|-----------|
| decisão 3 |
| decisão 2 |
| decisão 1 |

Registramos cada decisão tomada

# Recursão com Tentativa e Erro

## E se não pudermos/quisermos usar recursão?

- Note que isso é exatamente o que a pilha de recursão faz

|           |
|-----------|
| decisão 4 |
| decisão 3 |
| decisão 2 |
| decisão 1 |

Registramos cada decisão tomada

# Recursão com Tentativa e Erro

## E se não pudermos/quisermos usar recursão?

- Note que isso é exatamente o que a pilha de recursão faz

|           |
|-----------|
| decisão 4 |
| decisão 3 |
| decisão 2 |
| decisão 1 |

Em caso de falha, desempilhamos a decisão problemática

# Recursão com Tentativa e Erro

## E se não pudermos/quisermos usar recursão?

- Note que isso é exatamente o que a pilha de recursão faz

|           |
|-----------|
| decisão 3 |
| decisão 2 |
| decisão 1 |

Voltando à decisão imediatamente anterior a esta

# Recursão com Tentativa e Erro

## E se não pudermos/quisermos usar recursão?

- Note que isso é exatamente o que a pilha de recursão faz

|           |
|-----------|
| decisão 5 |
| decisão 3 |
| decisão 2 |
| decisão 1 |

E podendo então tomar outra decisão no lugar da que falhou



# Referências

- Ziviani, Nivio. Projeto de Algoritmos: com implementações em Java e C++. Cengage. 2007.
- Gersting, Judith L. Fundamentos Matemáticos para a Ciência da Computação. 3a ed. LTC. 1993.