Tentativa e Erro (ou força bruta)

- É a mais simples das estratégias de projeto

🔎 **Definição:**

> Solução prática para resolver um problema, baseada diretamente no enunciado do problema em questão e nas definições dos conceitos envolvidos

Segue o princípio do "somente faça" -> Força do PC

Uma das estratégias mais fáceis

- **Exemplo:**

Ordenação, busca, multiplicação de matrizes, casamento de cadeias

- **Características principais:**

Algoritmo simples e de fácil compreensão

Algoritmos de valor prático

Algoritmos com limitações quanto ao tamanho da entrada

Mesmo sendo pouco eficiente (ou ineficiente), essa técnica pode ser útil para resolver problemas de pequena instância

- Seleção:

Encontra o menor e troca com o da primeira posição

Complexidade $O(n^2)$

```
SELECAO (A, n)
1. para i = 1 até n - 1 faça
2.     menor = i
3.     para j = i + 1 até n faça
4.         se A[j] < A[menor] então menor = j
5.     aux = A[menor]
6.     A[menor] = A[i]
7.     A[i] = aux
```

	1	2	3	4	5	6
Chaves iniciais:	O	R	D	E	N	A
i = 1	**A**	R	D	E	N	**O**
i = 2	A	**D**	**R**	E	N	O
i = 3	A	D	**E**	**R**	N	O
i = 4	A	D	E	**N**	**R**	O
i = 5	A	D	E	N	**O**	**R**

- Busca sequencial ou linear

Busca um elemento até encontrar ele dentro de uma lista ou até a lista ser exaurida (elemento não encontrado)

Complexidade $O(n)$ → busca binária é melhor: $O(\log n)$

```
LINEAR (A, n, x)
1. i  =  1
2. enquanto (i <= n  e  x != A[i]) faça
3.     i = i + 1
4. se (i <= n)
5.     então escreve ("encontrado")
6.     senão escreve ("ausente")
```

- Casamento de cadeias:
  - Dada uma cadeia T de n caracteres chamada de texto, e uma cadeia P de m caracteres ($m \leq n$) chamada de padrão, a ideia é encontrar partes do texto que "coincidam" com o padrão.

```
STRING_MATCH (T, n, P, m)
1. para (i = 1) até (n - m + 1) faça
2.     j = 1
3.     k = i
4.     enquanto (P[j] = T[k] e j <= m) faça
5.         j = j + 1
6.         k = k + 1
7.     se (j > m)
8.         retorna (i)  % padrão encontrado
9. retorna (-1)
```

  - O pior caso acontece quando o algoritmo tem que comparar todos os m caracteres antes de deslocar-se e isso pode ocorrer para cada uma das $n - m$ tentativas.

```
t =  x x x x x x x x x x x x x x x x x x x x
-------------------------------------------
p =  x x x x x x x x y
  p =  x x x x x x x x y
    p =  x x x x x x x x y
      p =  x x x x x x x x y
        ...
```

Complexidade: $O(nm)$ → Shift-and: $O(n)$
BMH: $O(n/m)$

- 👍Prós:

Ampla aplicabilidade
Simplicidade

- 👎Contras:

Baixa criatividade
Ruim para problemas grandes

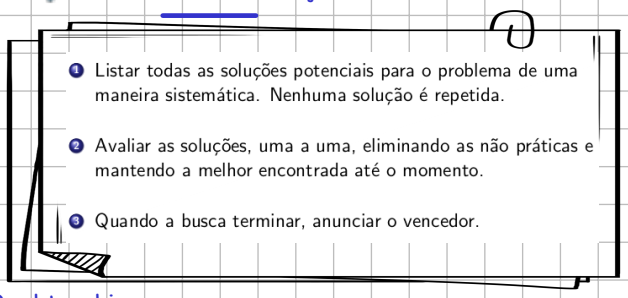
- Tipos de problemas:

Decisão:

✔️Almejam encontrar um elemento com propriedades especiais em um domínio que cresce com o tamanho da instância
✔️Sim/Não
✔️Mostra uma estrutura

Otimização:

✔️Problemas ainda mais complexos que buscam encontrar uma solução máxima (max) ou mínima (min) em relação a algum critério
✔️Melhor para cima ou melhor para baixo
✔️Mostra a melhor solução



- Backtracking

✔️Força bruta com uma certa inteligência
✔️Para PROBLEMAS DE DECISÃO
✔️Armazena coisas
✔️Não melhora a complexidade
✔️incrementalmente constrói candidatas de soluções e **abandona** uma candidata parcialmente construída tão logo quanto for possível determinar que ela não pode gerar uma solução válida
✔️Na prática mais rápido, pois backtracking só gera soluções válidas (restrição p/ evitar repetição) e o força bruta gera todas as possíveis soluções para depois verificar se são válidas
✔️Backtracking e força bruta usam busca em profundidade
  - Exemplo: ciclo hamiltoniano: percorrer todos os vértices (uma vez) e volta para a origem

- Branch-and-bound

✔️Não termina ao achar a primeira solução. Ele continua até achar a **melhor** solução ser encontrada
✔️Tem um mecanismo de pontuação que **encerra/interrompe** a tentativa tão logo se saib a que a mesma não levará a um valor menor (min) ou ultrapassará um dado limite (máx)
✔️No caso do ciclo hamiltoiniano é quando eu ionterrompo um caminho quando eu já tenho o melhor valor
✔️Serve para problemas de **otimização e minimização**
  - Exemplo: Caixeiro viajante -> encontrar o trajeto mais curto que passe por todas as cidades exatamente uma vez antes de retornar a cidade de origem
    Problema da mochila -> preencher uma mochila com objetos de diferentes pesos e valores. O objetivo é que se preencha a mochila com o maior valor possível sem ultrapassar seu peso máximo