**Quadro Comparativo de Estratégias e Paradigmas de Algoritmos**

| **Paradigma / Técnica** | **Ideia Principal** | **Exemplo Clássico** | **Vantagens** | **Desvantagens / Limitações** | **Tipo de Problema Ideal** |
| --- | --- | --- | --- | --- | --- |
| **Divisão e Conquista (Divide and Conquer)** | Divide o problema em subproblemas menores, resolve cada um e combina os resultados. | Merge Sort, Quick Sort, Busca Binária | Alta eficiência; fácil paralelização; lógica estruturada. | Pode ter sobreposição de subproblemas; requer boa estratégia de divisão. | Problemas que podem ser naturalmente divididos e combinados. |
| **Algoritmos Gulosos (Greedy)** | Faz a melhor escolha local em cada passo, esperando que leve à solução ótima global. | Problema do Troco, Mochila Fracionária, Árvore Geradora Mínima (Kruskal/Prim) | Simples, rápido e eficiente para certos problemas. | Nem sempre encontra a solução ótima global. | Problemas com subestrutura ótima e propriedade de escolha gulosa. |
| **Programação Dinâmica (PD)** | Resolve subproblemas menores e armazena resultados para evitar recomputações. | Fibonacci, Mochila 0/1, Caminho Mínimo (Bellman-Ford) | Otimiza tempo ao evitar repetição de cálculos. | Requer memória extra; difícil de implementar em problemas grandes. | Problemas com subproblemas sobrepostos e subestrutura ótima. |
| **Problemas NP (Não Polinomiais)** | Problemas cuja solução é fácil de verificar, mas difícil de encontrar (tempo exponencial). | Caixeiro Viajante, Coloração de Grafos, Mochila Inteira | Estudo fundamental da complexidade; base teórica importante. | Nenhum algoritmo eficiente conhecido; explosão combinatória. | Problemas de otimização e decisão complexos. |
| **Recursividade** | Função que chama a si mesma para resolver subproblemas menores. | Fatorial, Torre de Hanói, Fibonacci | Código limpo e elegante; aproximação natural de problemas recursivos. | Pode causar sobrecarga de pilha; ineficiente sem memoização. | Problemas naturalmente definíveis de forma recursiva. |
| **Incremental** | Constrói a solução passo a passo, adicionando elementos gradualmente. | Algoritmo de Inserção, construção de grafos MST (Prim) | Simples e intuitivo; fácil controle do progresso. | Pode não garantir solução ótima; pode exigir ajustes posteriores. | Problemas que podem ser construídos progressivamente. |
| **Interativo (Iterativo)** | Usa laços (for/while) para repetir cálculos até uma condição de parada. | Cálculo de fatorial iterativo, busca linear, ordenação por inserção | Geralmente mais eficiente em memória que recursão. | Pode ser mais verboso; menos intuitivo para problemas recursivos. | Processos repetitivos e com controle explícito de iteração. |
| **Árvore** | Estrutura hierárquica de dados com nós e ramos; usada em busca, ordenação e hierarquias. | Árvores Binárias, AVL, Heaps, Árvores de Decisão | Busca e inserção eficientes; estrutura lógica natural. | Pode desbalancear; requer manutenção. | Problemas hierárquicos e de busca ordenada. |
| **Grafos** | Estrutura que representa relações entre entidades (vértices e arestas). | Dijkstra, Bellman-Ford, DFS, BFS | Representa redes e conexões complexas; base de muitos algoritmos. | Custo computacional pode ser alto; representação complexa. | Problemas de redes, caminhos, fluxos e conexões. |