ESTRUTURAS DE DADOS: CRIAÇÃO, ADAPTAÇÃO E APLICAÇÕES MODERNAS EM SISTEMAS DE GRANDE ESCALA

Autor: *Jesiel Araújo (GitHub)*
Ano: 2025

Resumo

Estruturas de dados constituem o núcleo da ciência da computação e da engenharia de software moderna. Sua correta escolha determina eficiência, escalabilidade, segurança e até mesmo a viabilidade financeira de sistemas complexos. Este artigo apresenta uma visão acessível, porém profunda, sobre criação e adaptação de estruturas de dados, descrevendo conceitos fundamentais, boas práticas, erros comuns, aplicações em empresas de grande porte e insights provenientes de estudos científicos recentes. Busca-se despertar no leitor não apenas conhecimento técnico, mas entusiasmo pelo potencial transformador que estruturas bem projetadas possuem.

1. Introdução

Estruturas de dados são entidades abstratas que organizam, armazenam e manipulam informações de forma eficiente. Para Knuth (1997), elas constituem “as ferramentas essenciais da arte de programar”. Em sistemas modernos — desde redes neurais até bancos de dados distribuídos — o comportamento emergente depende da interação entre algoritmos e estruturas de dados adequadamente projetadas.

Em gigantes da tecnologia como Google, Amazon e Meta, a escolha de uma estrutura de dados raramente é trivial: afeta latência global, throughput, custo computacional, consumo energético e consistência. Assim, compreender **como criar, adaptar e escolher estruturas de dados** é uma das habilidades mais críticas para qualquer engenheiro.

2. Importância das Estruturas de Dados no Desenvolvimento Moderno

A relevância prática é confirmada por diversos estudos (Cormen et al., 2009; Sedgewick & Wayne, 2011):

1. **Performance** – alterações sutis em estruturas internas podem reduzir tempos de execução em ordens de magnitude.
2. **Escalabilidade** – sistemas distribuídos dependem de estruturas que minimizam contenção e comunicação.
3. **Uso eficiente de memória** – essencial em dispositivos embarcados, IoT e sistemas de tempo real.
4. **Tolerância a falhas** – certos tipos de árvores e logs são projetados para recuperação segura.
5. **Modelagem do Mundo Real** – grafos, heaps, tries e árvores customizadas permitem representar fenômenos complexos.

3. Criação e Adaptação de Estruturas de Dados

A criação de uma nova estrutura de dados ocorre quando:

- Nenhuma estrutura clássica (árvore, lista, grafo, hash) atende plenamente ao problema;
- Requisitos de performance exigem modificações internas;
- O sistema opera em ambiente distribuído, com limitações específicas.

3.1. Etapas do Processo Científico de Criação

1. **Modelagem formal do problema**
    - Definir entradas, operações, restrições temporais e espaciais.
2. **Construção de abstrações**
    - A partir de estruturas base (ex.: árvore + hash → *hash tree*).
3. **Prova formal de corretude**
    - Demonstração matemática de manutenção das invariantes.
4. **Análise de complexidade**
    - Big-O, comportamento amortizado, limites inferiores.
5. **Validação empírica**
    - Testes, benchmarks, análise de colisões, perfis de memória.

3.2. Estruturas Adaptadas Usadas no Mercado

Grandes empresas normalmente não utilizam apenas estruturas clássicas puras, mas **versões otimizadas**:

Empresa	Estrutura Adaptada	Uso
Google	**B-Tree + Log Structured Merge Trees (LSM-Tree)**	Sistemas de busca e bancos distribuídos
Amazon	**Hash distribuído consistente**	Balanceamento de carga e sharding
Meta	**TAO Graph (grafo distribuído customizado)**	Modelagem de conexões sociais
Netflix	**Armazenamento baseado em heaps e priority queues**	Prioridade de jobs e streaming adaptativo
Microsoft	**Trie compactado + compressão**	Autocompletar e indexação

Essas estruturas raramente são ensinadas em cursos tradicionais, pois são adaptações híbridas criadas a partir de pesquisas internas, benchmarks e requisitos externos.

4. Dicas Essenciais para Uso Correto

✓ 4.1. Sempre modele o problema antes da estrutura

Evite “escolher uma árvore porque gosta de árvores”. As operações e restrições definem a estrutura — nunca o contrário.

✓ 4.2. Prefira estruturas imutáveis quando possível

Em sistemas paralelos, imutabilidade reduz race conditions e é amplamente usada por Google, Amazon e Spotify.

✓ 4.3. Evite estruturas complexas desnecessariamente

Simplicidade > engenhosidade.
Muitas falhas em sistemas ocorrem por estruturas excessivamente difíceis de manter.

✓ 4.4. Estabeleça invariantes claras

Toda estrutura deve responder: “o que nunca pode acontecer?”
Sem essa definição, bugs se acumulam silenciosamente.

✓ 4.5. Métricas reais importam mais que Big-O

Uma estrutura com complexidade *teoricamente pior* pode ser superior em hardware real devido a:

- cache locality
- branch prediction
- paralelismo

Exemplo: arrays superam listas ligadas na maioria dos casos modernos (McSherry, 2015).

5. Erros Comuns e o Que Evitar

❌ 5.1. Escolher estrutura pelo nome ou moda

“Vamos usar Grafos porque é legal.”
Não.
Cada estrutura tem custo e responsabilidade.

❌ 5.2. Esquecer o comportamento no pior caso

Especialmente em sistemas críticos (financeiros, médicos, aeronáuticos).

❌ 5.3. Confiar em estruturas prontas sem entender implementação

Muitos problemas surgem porque a equipe assume que a biblioteca padrão resolve tudo.

❌ 5.4. Não considerar concorrência

Muitas árvores clássicas quebram em ambientes paralelos.
Por isso surgiram: **Skip Lists lock-free**, **Concurrent HashMaps**, **Segment Trees paralelas**.

6. Estruturas de Dados Mais Utilizadas em Grandes Companhias

Abaixo, uma visão consolidada:

6.1. Árvores balanceadas (B-Trees, AVL, Red-Black)

Bancos de dados, indexação, sistemas de busca.

6.2. Tries e Prefix Trees

Autocompletar, pesquisa, compressão, sistemas distribuídos.

6.3. Grafos customizados

Detecção de comunidades, recomendações, redes sociais.

6.4. Log Structured Merge Trees (LSM-Trees)

Sistemas de larga escala como BigTable, Cassandra, RocksDB.

6.5. Estruturas probabilísticas

- Bloom Filters
- Count-Min Sketch
- HyperLogLog

Usadas para economia de memória em Big Data.

6.6. Estruturas lock-free

Fundamentais para concorrência sem gargalos.

7. Considerações Finais

Estruturas de dados não são meros componentes internos: são **fundamentos arquiteturais** que determinam a eficiência e a elegância de sistemas modernos. A capacidade de criar e adaptar estruturas é uma habilidade que separa desenvolvedores comuns de engenheiros altamente especializados. Espera-se que este artigo desperte no leitor a mesma paixão que muitos pesquisadores sentem ao lidar com algoritmos, complexidade e estruturas elegantes que tornam o software possível.

Referências (ABNT)

- CORMEN, Thomas H. et al. *Algoritmos: Teoria e Prática*. 3. ed. Rio de Janeiro: Elsevier, 2009.
- KNUTH, Donald. *The Art of Computer Programming*. 3. ed. Boston: Addison-Wesley, 1997.
- MC SHERRY, Frank. *Scalability! But at what COST?* Proceedings of HotOS XV. USENIX, 2015.
- SEDGEWICK, Robert; WAYNE, Kevin. *Algorithms*. 4. ed. Addison-Wesley, 2011.
- DEAN, Jeffrey; GHEMAWAT, Sanjay. *MapReduce: Simplified Data Processing on Large Clusters*. OSDI, 2004.
- O'NEIL, Patrick et al. *The Log-Structured Merge Tree (LSM-Tree)*. Acta Informatica, 1996.
- SHARDANAND, Upendra; MAES, Pattie. *Social Information Filtering*. ACM SIGCHI, 1995.