Módulo 03 · Fundamentos

Algoritmos, Notação Big-O, Ponteiros & Memória

Portal de Computação — Alinhamento CS50 Harvard

Gestão da Tecnologia da Informação — Fatec Assis

Agenda do Módulo 03 — 50 minutos

Bloco	Tempo	Tópico	Alinhamento CS50
1	12 min	Algoritmos de Busca & Notação Big-O	CS50 Week 3 (Linear vs Binary)
2	13 min	Algoritmos de Ordenação (Bubble & Merge Sort)	CS50 Week 3 (Sorting & Recursion)
3	13 min	Ponteiros & Alocação (Call Stack vs Heap)	CS50 Week 4 (Pointers & `malloc`)
4	12 min	Estruturas de Dados Dinâmicas	CS50 Week 5 (Linked Lists & Hash)

Bloco 1 · Busca & Notação Big-O

Análise de Complexidade de Algoritmos

Notação Big-O e Curvas de Crescimento

Notação	Nome	Exemplo Prático	Desempenho para $n = 1.000.000$
$O(1)$	Constante	Acesso direto ao array `arr[5]`	**1 passo**
$O(\log n)$	Logarítmica	Busca Binária em array ordenado	**20 passos**
$O(n)$	Linear	Busca Sequencial / Percorrer lista	**1.000.000 passos**
$O(n^2)$	Quadrática	Bubble Sort (laços aninhados)	**1.000.000.000.000 passos**

Busca Linear ($O(n)$) vs Busca Binária ($O(\log n)$)

Busca Linear (CS50 Week 3)

- Percorre o array célula por célula a partir do índice 0.
- Funciona com dados não ordenados.
- **Pior Caso:** n comparações.

Busca Binária (CS50 Week 3)

- **Exige que o array esteja previamente ordenado.**
- Divide o espaço de busca ao meio repetidamente.

> 📊 **Demonstração:** Teste no **Simulador Visual de Algoritmos** do portal!

Bloco 2 · Algoritmos de Ordenação

Bubble Sort & Merge Sort

Bubble Sort ($O(n^2)$) vs Merge Sort ($O(n \log n)$)

- **Bubble Sort:** Compara elementos vizinhos `array[j] > array[j+1]` e faz a troca (*swap*). Repete em laço duplo. Simples, porém lento para grandes volumes de dados.
- **Merge Sort:** Algoritmo de **Divisão e Conquista**. Divide a lista recursivamente em metades e as intercala ordenadamente.

```
Entrada:  [45, 12, 89, 34]
Dividir:  [45, 12]  e  [89, 34]
Ordenar:  [12, 45]  e  [34, 89]
Intercalar: [12, 34, 45, 89]
```

Bloco 3 · Ponteiros & Memória

Call Stack vs Heap (CS50 Week 4)

Mapeamento de Memória & Ponteiros em Hexadecimal

> 🎯 **Demonstração:** Teste a alocação dinâmica no **Simulador de Ponteiros e Memória** do portal!

Bloco 4 · Estruturas de Dados

Listas Encadeadas & Hash Tables (CS50 Week 5)

Listas Encadeadas vs Arrays

- **Arrays (Vetores):** Bloco contíguo de memória. Acesso rápido por índice ($O(1)$), mas inserção e remoção no meio são lentas ($O(n)$).
- **Listas Encadeadas (Linked Lists):** Nós espalhados pela memória conectados por ponteiros (`node->next`). Inserção dinâmica $O(1)$, mas acesso requer percorrer a lista.

```
[ Dado: 10 | Next: 0x006 ] ──▶ [ Dado: 20 | Next: 0x00A ] ──▶ NULL
```

Tabelas Hash (Hash Tables)

Combinam arrays com **Funções Hash** para obter acesso e busca em tempo médio de $O(1)$!

Resumo do Módulo 03

1. **Big-O:** Avalia o desempenho dos algoritmos ($O(1)$, $O(\log n)$, $O(n)$, $O(n^2)$).
2. **Busca Binária:** Reduz drasticamente o tempo em dados ordenados ($O(\log n)$).
3. **Ponteiros & Heap:** Permitem alocação dinâmica de memória via `malloc()` e `free()`.
4. **Estruturas:** Listas encadeadas usam ponteiros para nós flexíveis.

> 💡 **Próximo Módulo:** Módulo 04 — Da Linguagem C ao Assembly, Registradores CPU & Criptografia.