Estruturas de Dados

Aula 1 – Introdução às Estruturas de Dados

Prof. Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal da Bahia – Campus Feira de Santana

2026

Agenda

Introdução às Estruturas de Dados Simples

Por que precisamos delas?

O que são Estruturas de Dados?

Definição

Servem para **organizar os dados** de modo que possam ser processados mais **eficientemente**. Eficiência está diretamente relacionada à **escalabilidade** — como o tempo de processamento cresce conforme o tamanho do problema aumenta.

A questão central

Como o tempo de processamento cresce quando a **quantidade de dados aumenta**?

- De maneira **moderada**?
- De maneira **explosiva**?

Exemplo motivador

Qual a diferença entre encontrar uma palavra em uma lista de **10** palavras e encontrar em uma lista de **10.000** palavras?

O tempo é $10\times$, $1.000\times$ ou $1.000.000\times$ maior?

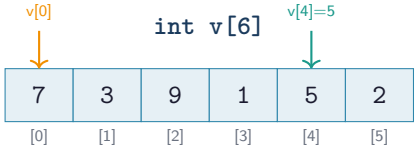
Vetores

Estrutura de dados sequencial

O que é um Vetor?

Definição

Um **vetor** (ou *array*) é uma estrutura que armazena uma sequência de objetos do **mesmo tipo** em posições **consecutivas** da memória RAM. Permite **acesso aleatório**: qualquer elemento pode ser acessado diretamente pelo índice, sem percorrer os anteriores. [Cormen et al. 2009]



Três problemas fundamentais

1. **Busca** — encontrar um elemento
2. **Inserção** — adicionar um novo elemento
3. **Remoção** — eliminar um elemento

Esses problemas reaparecerão em todas as estruturas!

Índices

Posições de **0** a **n**$-$**1**.
$n = 0$: vetor **vazio**
$n = N$: vetor **cheio**

Declaração e Tamanho

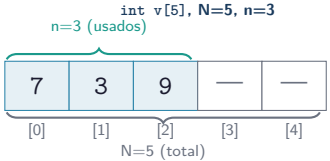
Declaração em C

```
int v[N]; // N deve ser constante
```

Convenção de tamanho

Considerando n elementos armazenados ($0 \leq n \leq N$):

- $v[0\,..\,(n-1)]$ é o vetor de inteiros
- Se $n = 0$: vetor vazio
- Se $n = N$: vetor cheio



Busca em Vetores

Encontrando elementos

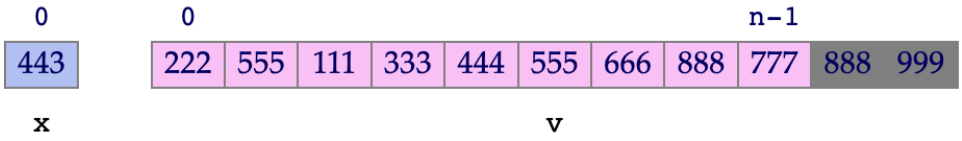
Problema

Dado um inteiro x e um vetor $v[0\,..\,(n-1)]$, encontrar um índice k tal que $v[k] = x$.

Decisão de projeto

O que fazer quando x **não está** no vetor?

Convenção: devolver `-1` — valor que não é índice válido ($-1 \notin [0, n-1]$).



Busca: Algoritmo com `while`

```
int busca (int x, int n, int v[]){
    int i;
    i = n-1;
    while (i >= 0 && v[i] != x)
        i -= 1;
    return i;
}
```

Busca: Versão com `for` — mais elegante

```c
int busca (int x, int n, int v[]){
   int i;
   /**i = n-1;
   while (i >= 0 && v[i] != x)
      i -= 1;**/
   for(i = (n - 1); ((i >= 0) && (v[i] != x)); --i);
   return i;
}
```

Exercícios

1. Escreva uma função que receba x, v e n e devolva **1** se $x \in v[0\,..\,(n-1)]$ e **0** caso contrário.
2. Suponha que $v[0\,..\,(n-1)]$ tem dois ou mais elementos iguais a x. Qual deles será apontado pela função de busca?
3. Quantas **comparações** entre x e elementos do vetor a função de busca realiza?

Exercícios

Manipulação de Vetores

Exercícios

1. Diga (sem usar o computador) qual o conteúdo do vetor v após as instruções abaixo:

```
int v[99];
for (i=0; i<99; ++i) v[i] = 98-i;
for (i=0; i<99; ++i) v[i] = v[v[i]];
```

2. **Inversão.** Dado $v[1 \,..\, n]$ contendo uma permutação de $[1 \,..\, n]$, escreva uma função que **inverta** a permutação: se $v[i] = j$ originalmente, queremos $v[j] = i$ ao final.

Resposta — Exercício 1

▪ Raciocínio

O primeiro laço cria $v[i] = 98 - i$ (decrescente de 98 a 0).

O segundo laço faz $v[i] = v[v[i]] = v[98 - i] = 98 - (98 - i) = i$.

Para $i \leq 49$: $v[i] = i$ (crescente).
Para $i > 49$: $v[i] = i$ também (simétrico).

Resultado: $v[i] = i$ para todo i.

▪ Padrão resultante

`v[0]=0, v[1]=1, ..., v[98]=98`

```c
#include <stdio.h>
#include <stdlib.h>
int main (void){
int i;
int v[99];
for (i = 0; i < 99; ++i)
  v[i] = 98 - i;
for(i=0; i< 99; ++i)
  printf("v[%d] = %d\n", i,v[i]);
for (i = 0; i < 99; ++i)
  v[i] = v[v[i]];
for(i=0; i< 99; ++i)
  printf("v[%d] = %d\n", i,v[i]);
}

```

Resposta — Exercício 2: Inversão de Permutação

```c
#include <stdio.h>
#include <stdlib.h>
int main (void){
int i,j,aux;
int v1[10],v[]={8,3,4,7,6,5,1,2,9,0};


for (i = 0; i < 10; ++i){
  printf("v[%d] = %d\n", i, v[i]);
  v1[v[i]] = i;
}

  for (j = 0; j < 10; ++j)
    printf("Permutado v[%d] = %d\n", j, v1[j]);

  for (i = 0; i < 10; ++i)
    printf("Original v[%d] = %d\n", i, v[i]);
}
```

Referências I

CORMEN, T. H. et al. *Introduction to Algorithms*. 2nd. ed. [S.l.]: The MIT Press, 2009. ISBN 0262032937.