UTFPR

Ciência da Computação

Algoritmos e Estrutura de Dados 1

Estruturas de Dados

Rafael Liberato
liberato@utfpr.edu.br

Agenda

-

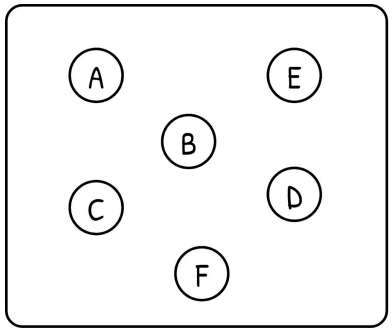
O que são Estruturas de Dados

Estruturas de Dados

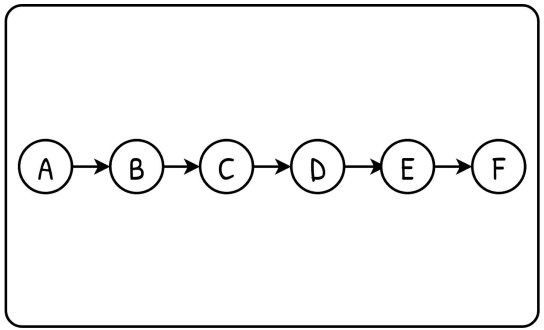
- São Tipos Abstratos de Dados
  - Composto por DADOS e OPERAÇÕES
  - Separa o conceito da implementação

- Representam uma coleção de elementos
  - Os elementos possuem um relacionamento lógico entre si

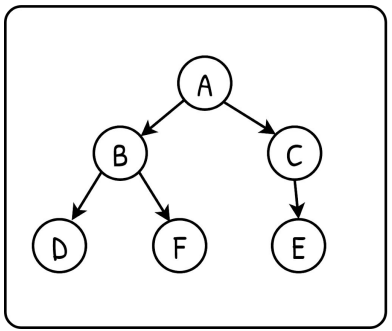
Conjunto



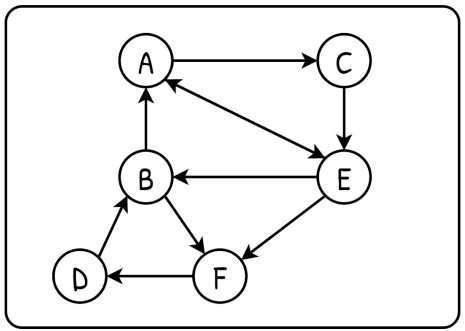
Lista



Árvore



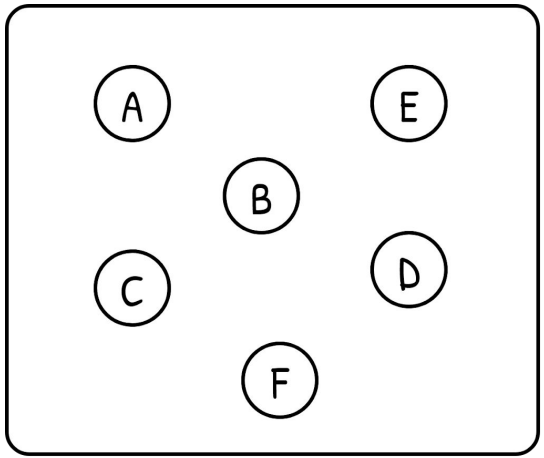
Grafo



Estrutura de Dados

Conjunto

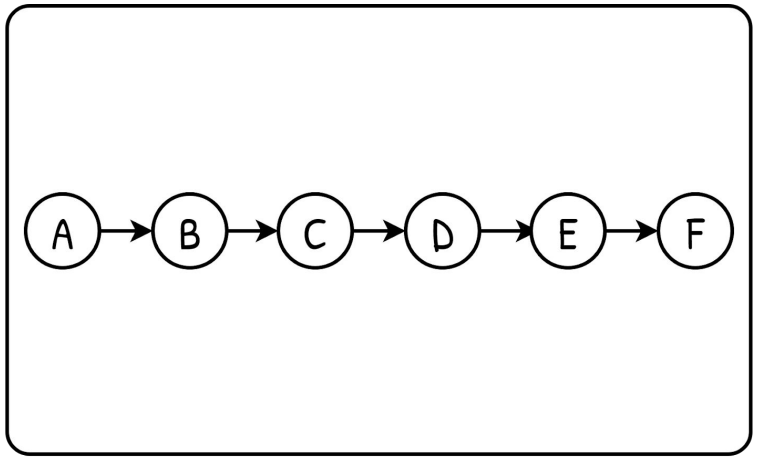
Uma coleção de elementos em que não há ordem nem repetição



Estrutura de Dados

Lista

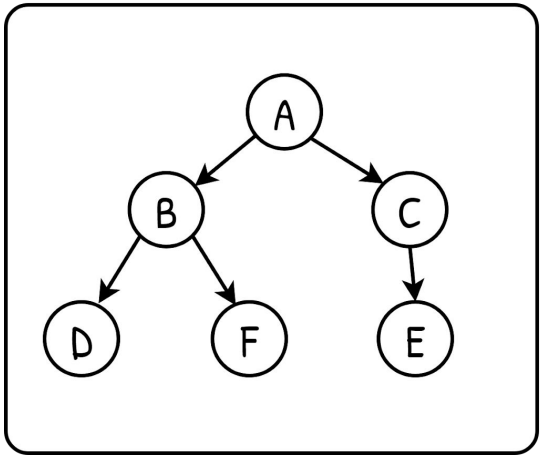
Uma coleção de elementos organizados linearmente



Estrutura de Dados

Árvore

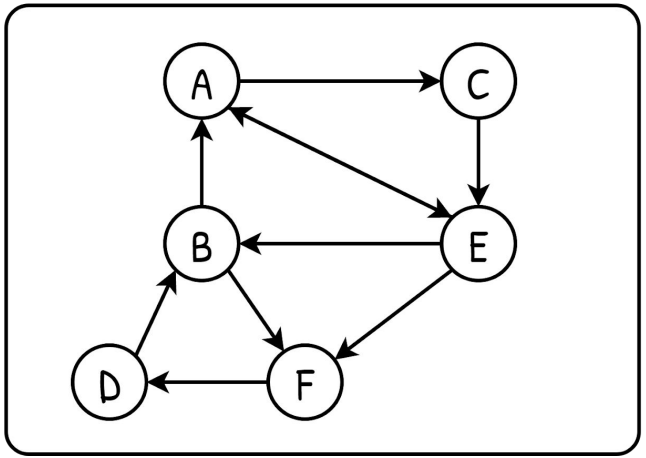
Uma coleção de elementos organizados hierarquicamente



Estruturas de Dados

Grafo

Uma coleção de elementos organizados em rede



Diferentes formas de representar o relacionamento dos dados

Alternativas de representação

Existem duas alternativas:

- Organização **contígua** dos dados na memória
- Organização **encadeada** dos dados na memória

organização
contígua

Organização contígua

Características

- Os dados são organizados em posições contíguas (sequenciais) na memória.
- Precisamos de um bloco de memória suficiente para armazenar todos os elementos.
  - Estratégia já utilizada em vetores e matrizes.
- Estratégia natural para coleções cujo relacionamento é:
  - Linear (lista), ou
  - Não há ordem (conjunto)
- Para coleções com relacionamento hierárquico ou rede, essa estratégia possui algumas restrições.

Organização contígua

Alocação de memória

- Alocação estática
  - O bloco de memória é definido em tempo de compilação (stack)
  - Não é necessário gerenciar o espaço alocado

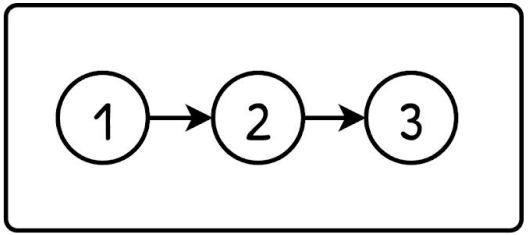
- Alocação dinâmica
  - O bloco de memória é definido em tempo de execução (heap)
  - Precisamos gerenciar o espaço alocado

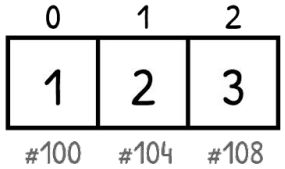
Organização contígua

Exemplos

Lista

Estruturas lineares



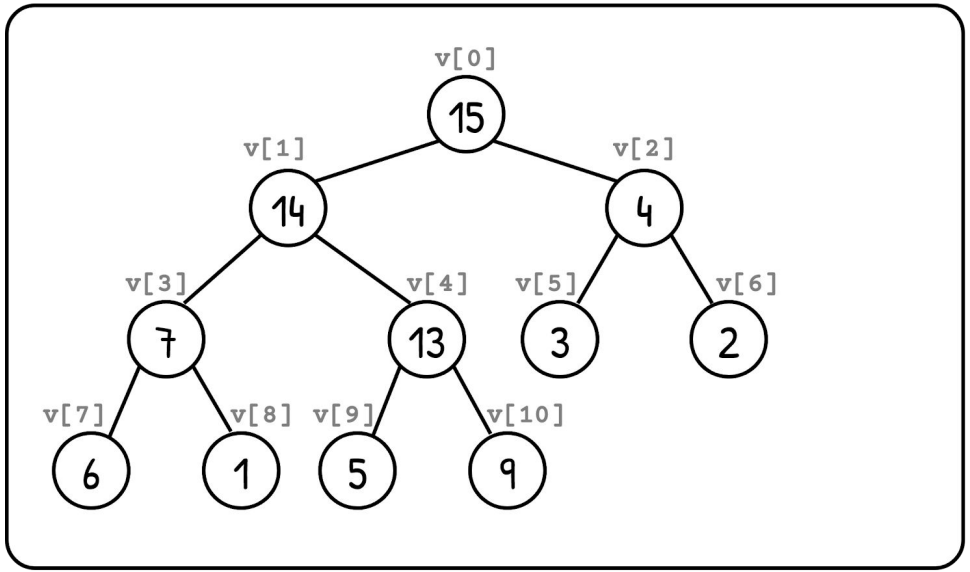


Organização contígua

Exemplos

Árvore

Estruturas hierárquicas



Pai(i) = (i-2)/2

FilhoEsquerda(i) = 2 * i + 1
FilhoDireita(i) = 2 * i + 2

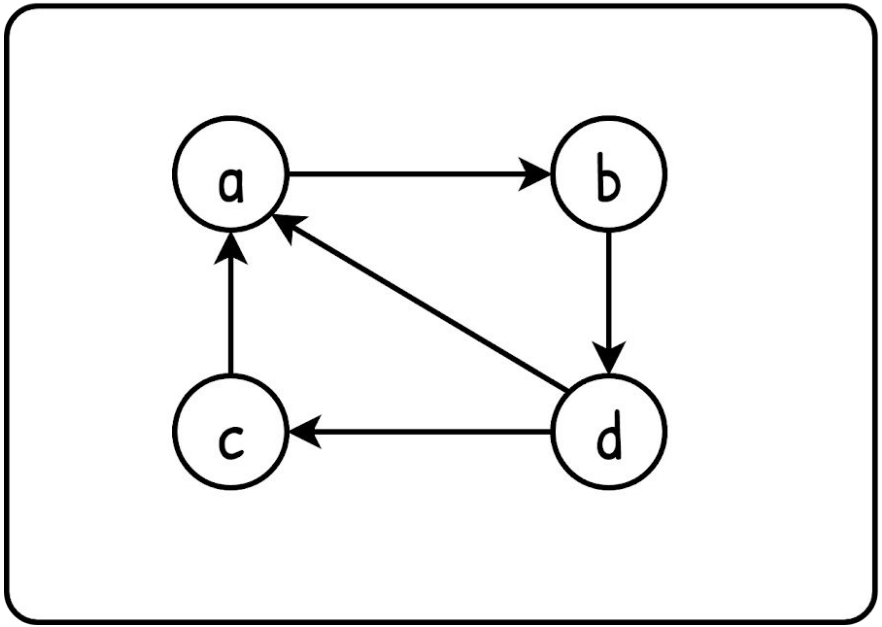
0	1	2	3	4	5	6	7	8	9	10
15	14	4	7	13	3	2	6	1	5	9
#100	#104	#108	#112	#116	#120	#124	#128	#132	#136	#140

Organização contígua

Exemplos

Grafo

Estruturas em rede



		a	b	c	d
		0	1	2	3
a	0	0	1	0	0
b	1	0	0	0	1
c	2	1	0	0	0
d	3	1	0	1	0

organização encadeada

Organização encadeada

Características

- Os dados são organizados de forma **fragmentada** na memória.
- A alocação do espaço é realizado **sob demanda**
- Os elementos ficam **espalhados** na memória.
- O relacionamento entre os elementos é realizado por meio de **ponteiros**.
    - Além do espaço utilizado para armazenar o elemento, precisamos de um espaço adicional com o endereço (ponteiro) do elemento com quem ele tem relacionamento
- Para isso criamos uma *struct*, tradicionalmente, chamada de “Nó”

```
typedef struct no{
    int dado;
    struct no* prox;
}No;
```

Organização encadeada

Alocação de memória

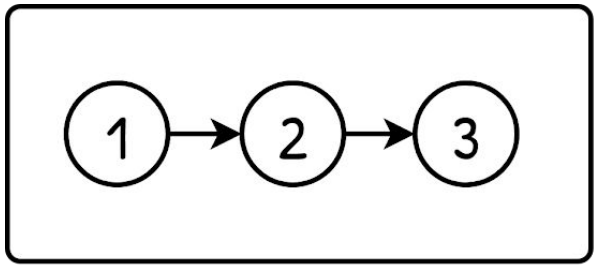
- Alocação dinâmica
  - O bloco de memória é definido em tempo de execução (heap)
  - Precisamos gerenciar o espaço alocado

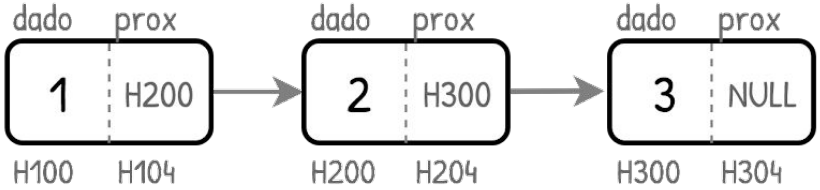
Organização encadeada

Exemplos

Lista

Estruturas lineares



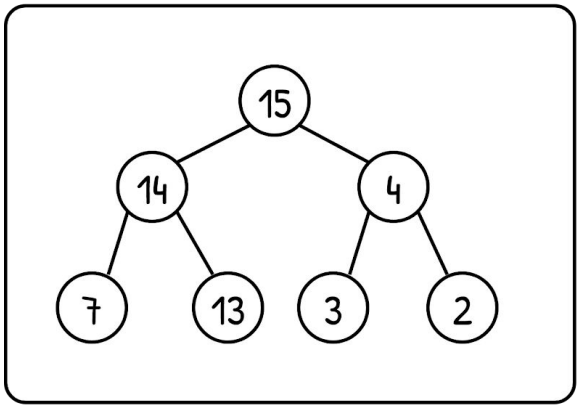


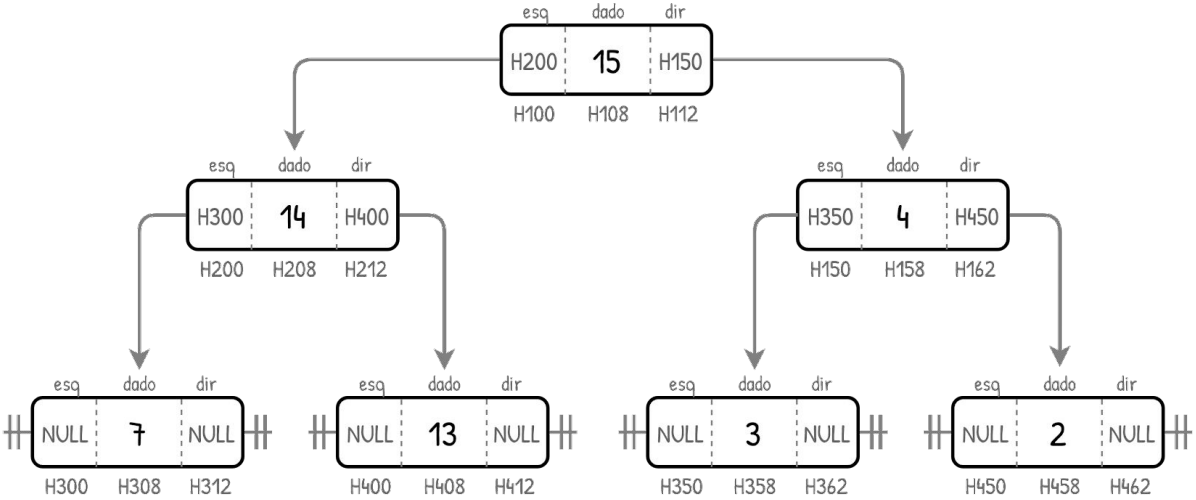
Organização encadeada

Exemplos

Árvore

Estruturas Hierárquicas

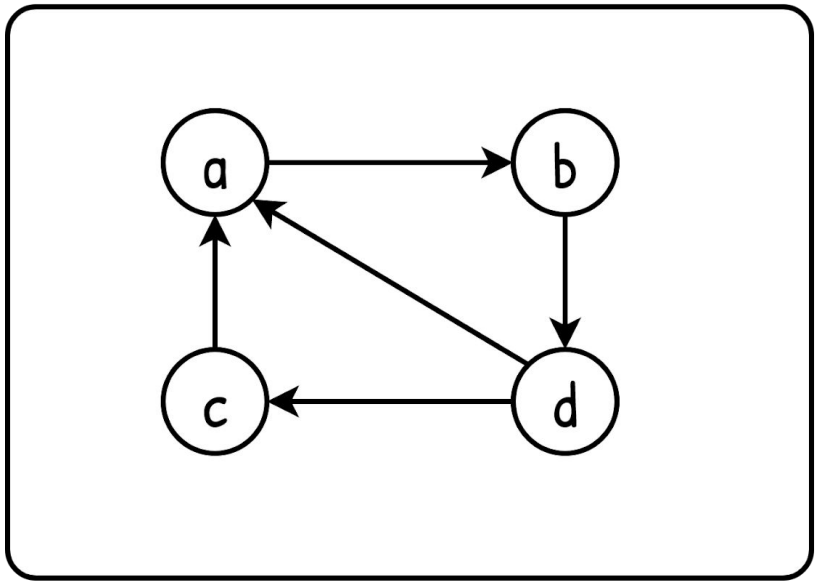




organização
mista

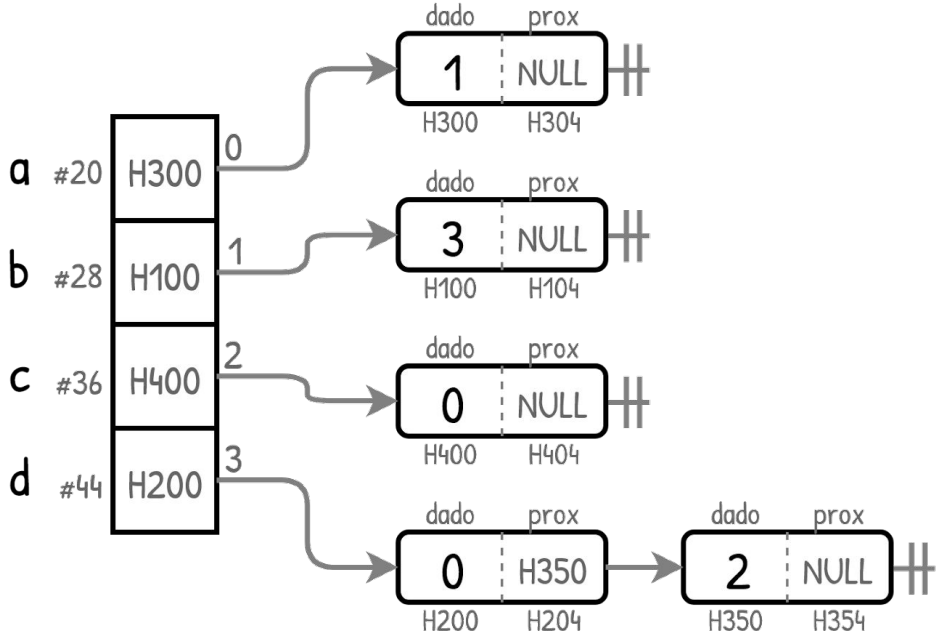
Organização mista

Exemplos



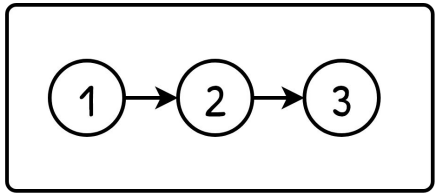
Grafo

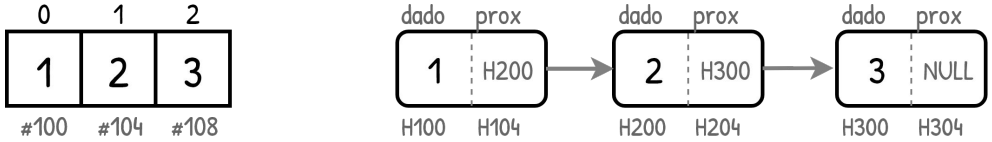
Estruturas em rede



implementação

Implementação

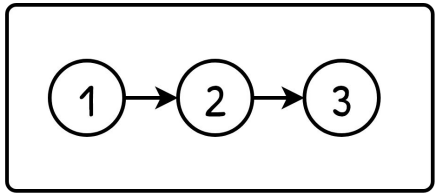


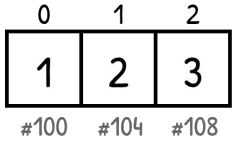


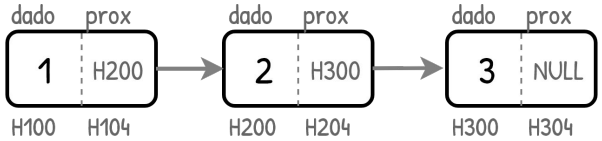

```c
// Alocando os espaços
    No* e1 = (No*) malloc(sizeof(No));
    No* e2 = (No*) malloc(sizeof(No));
    No* e3 = (No*) malloc(sizeof(No));

    // Preenchendo os elementos
    e1->dado = 1;
    e2->dado = 2;
    e3->dado = 3;

    // Estabelecimento os relacionamentos
    e1->prox = e2;
    e2->prox = e3;
    e3->prox = NULL;
```

```c
int* c = (int*) calloc(3, sizeof(int));
c[0] = 1;
c[1] = 2;
c[2] = 3;
```

Implementação



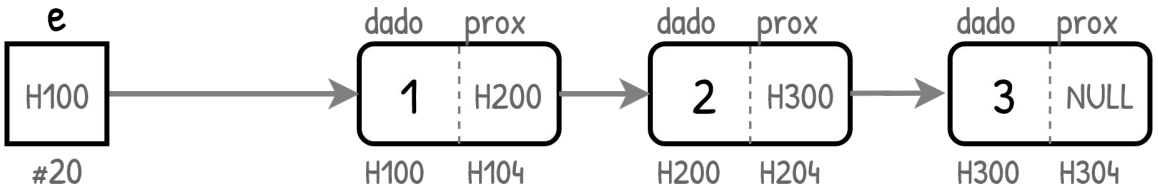





```c
int* c = (int*) calloc(3, sizeof(int));
c[0] = 1;
c[1] = 2;
c[2] = 3;
```

Implementação

Desenvolva uma função para imprimir todos os elementos do encadeamento abaixo.




Fim