"Mantenha a mente aberta, mas não aberta a ponto do seu cérebro cair para fora!"

Outras Representações

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR

Representando árvores n-Árias?

Como representar uma árvore 3-ária?

Cada nodo possui até 3 filhos.

Ideia

Como representar uma árvore 3-ária?

Cada nodo possui até 3 filhos.

```
struct nodo{
   tipo chave;
   struct nodo* fe;//filho esquerdo
   struct nodo* fc;//filho central
   struct nodo* fd;//filho direito
};
```

Ideia

Como representar uma árvore 3-ária?

Cada nodo possui até 3 filhos.

A ideia do exemplo funciona e está correta (dependendo do seu problema), mas não escala bem!

Por exemplo, como representar uma árvore 5-área? E uma com *n* filhos?

```
struct nodo{
   tipo chave;
   struct nodo* fe;//filho esquerdo
   struct nodo* fc;//filho central
   struct nodo* fd;//filho direito
};
```

Árvore n-Ária

Uma possibilidade é usar a estrutura a seguir:

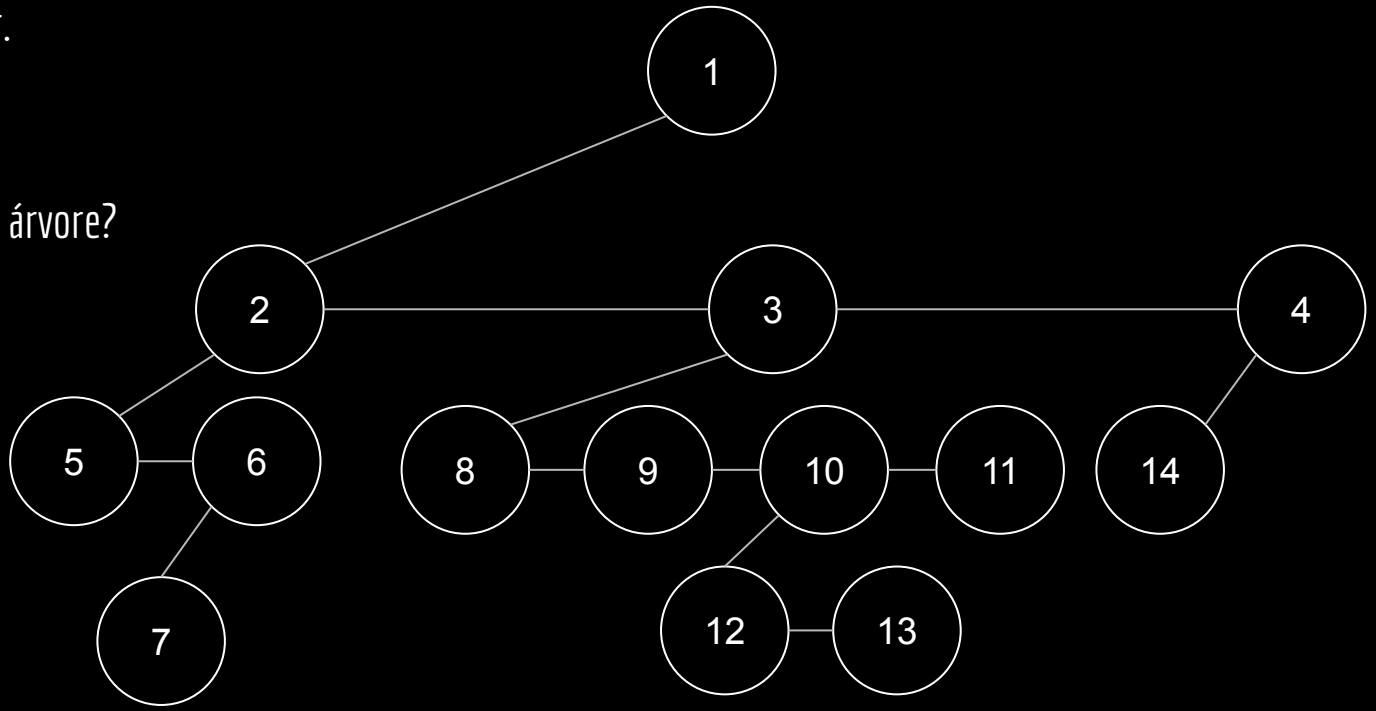
```
struct nodo{
   tipo chave;
   lista encadeada ou vetor de nodos;
};
```

Como representar?

Considere o exemplo a seguir.

Quem é filho de quem?

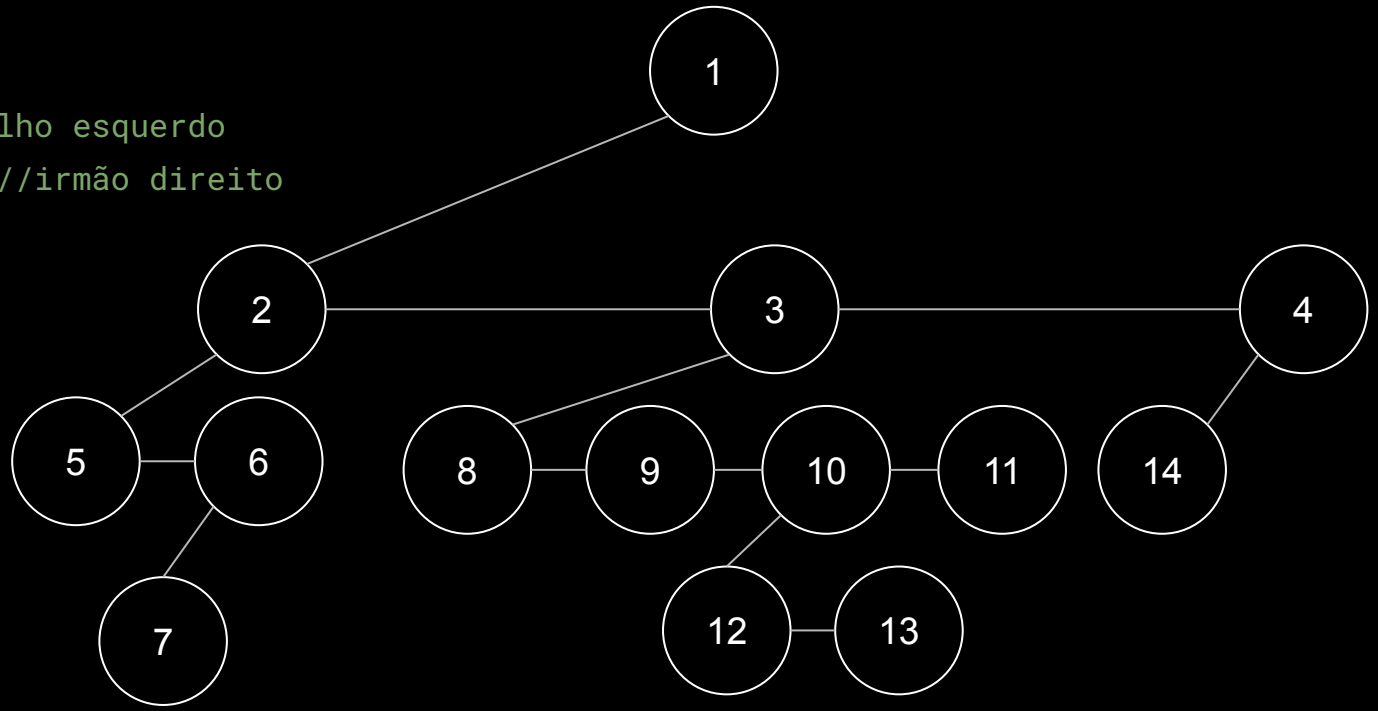
Como pode ser a struct dessa árvore?



O exemplo foi adaptado de Cormen Et. Al. 2022.

Filho esquerdo, irmão direito

```
struct nodo{
   int valor;
   struct nodo* fe;//filho esquerdo
   struct nodo* irmaod;//irmão direito
};
```



O exemplo foi adaptado de Cormen Et. Al. 2022.

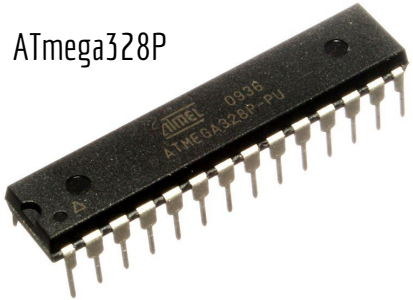
E se não tivermos memória dinâmica?

Existem muitos sistemas onde não é possível ou é muito custoso alocar memória dinamicamente.

Limitações de hardware e software.

Por exemplo, em um microcontrolador (que controla o freio ABS de um carro, um microondas, ...) a memória e conjunto de instruções são limitados.

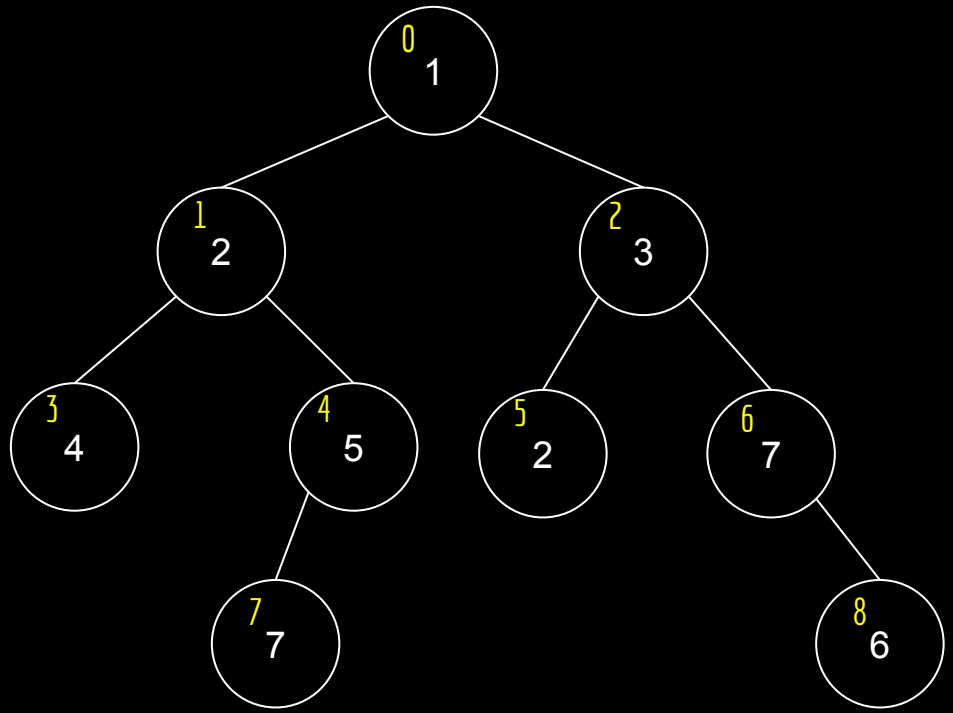
ATmega328P



- 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
- 256/512/512/1KBytes EEPROM
- 512/1K/1K/2KBytes Internal SRAM

Representando com matrizes

idx	Chave	Fe	Fd
0	1	1	2
1	2	3	4
2	3	2	7
3	4	-1	-1
4	5	7	-1
5	2	-1	-1
6	7	-1	8
7	7	-1	-1
8	6	-1	-1



Representando com matrizes

idx	Chave	Fe	Fd
0	1	1	2
1	2	3	4
2	3	2	7
3	4	-1	-1
4	5	7	-1
5	2	-1	-1
6	7	-1	8
7	7	-1	-1
8	6	-1	-1

Vantagens? Desvantagens?

Representando com matrizes

idx	Chave	Fe	Fd
0	1	1	2
1	2	3	4
2	3	2	7
3	4	-1	-1
4	5	7	-1
5	2	-1	-1
6	7	-1	8
7	7	-1	-1
8	6	-1	-1

- Não precisamos de alocação dinâmica;
- A árvore pode ficar mais compacta na memória;
  - Depende da estrutura da árvore.
- Algumas operações são mais rápidas.
  - Exemplo: listar todas as chaves.

Representando com matrizes

idx	Chave	Fe	Fd
0	1	1	2
1	2	3	4
2	3	2	7
3	4	-1	-1
4	5	7	-1
5	2	-1	-1
6	7	-1	8
7	7	-1	-1
8	6	-1	-1

+ Não precisamos de alocação dinâmica;
+ A árvore pode ficar mais compacta na memória;
    Depende da estrutura da árvore.
+ Algumas operações são mais rápidas.
    Exemplo: listar todas as chaves.
- A quantidade de nodos é fixa;
- O tamanho da matriz depende do grau máximo.
    - Problema sério se desejamos criar uma árvore em que cada nodo pode ter, por exemplo, 10 filhos;
    - Pior ainda se são raros os nodos com muitos filhos. Memória desperdiçada.

Outras Representações

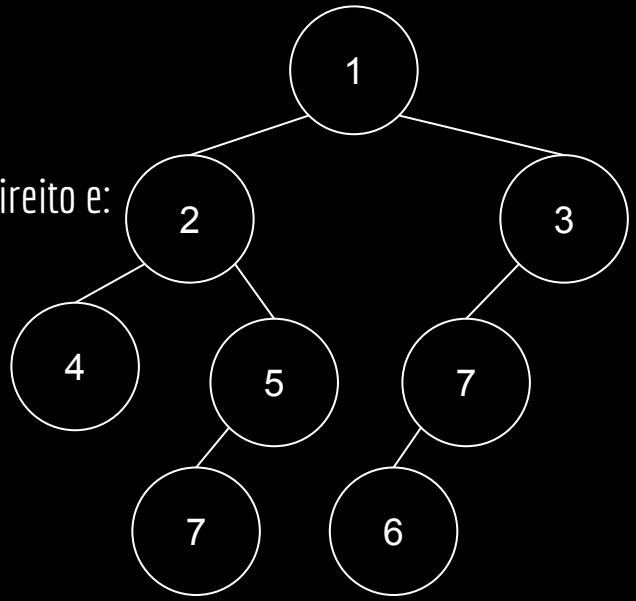
Existem ainda diversas outras representações possíveis.

Por exemplo, uma árvore completa ou quase completa pode ser representada com um array.

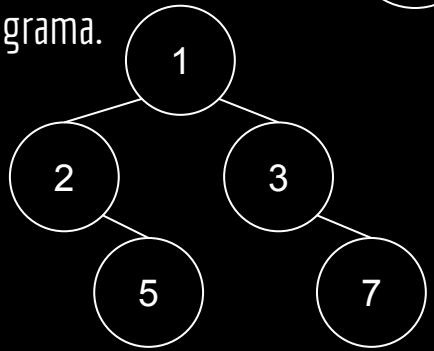
Veja a estrutura de uma heap (Algoritmos II).

Exercícios

1. Implemente uma árvore em C usando a estrutura filho esquerdo, irmão direito e:
    a. Insira a árvore dos slides nessa estrutura;
    b. Insira a árvore ao lado nessa estrutura;
    c. Faça uma função para DFS e uma para BFS.



2. É possível representar a árvore binária a seguir usando a estrutura filho esquerdo, irmão direito? Se sim, represente-a em seu programa.



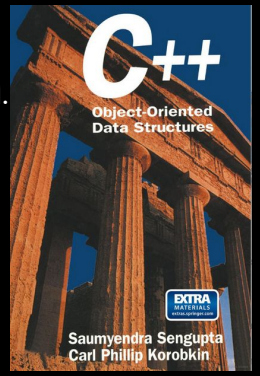
Exercícios

3. Implemente uma árvore binária em C usando a estrutura de matriz:

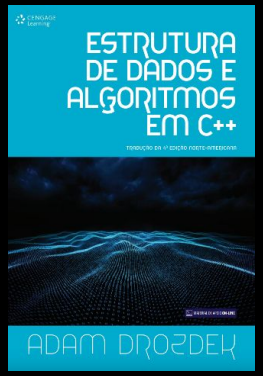
a. Insira as árvore do slide anterior nessa estrutura;
b. Percorra a árvore em pré-ordem, em ordem, e pós-ordem;
c. Faça uma função para DFS e uma para BFS.

Referências

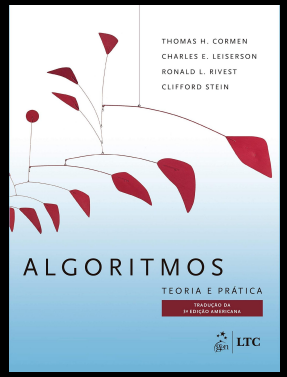
C. P Korobkin, S. Sengupta. C++: Object-Oriented Data Structures. 2012.



Estrutura de Dados e Algoritmos em C++. A. Drozdek. 4a ed. 2016.

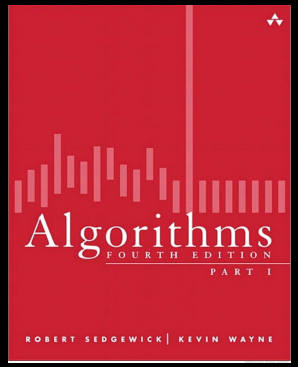


T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos: Teoria e Prática. 3a ed. 2012.

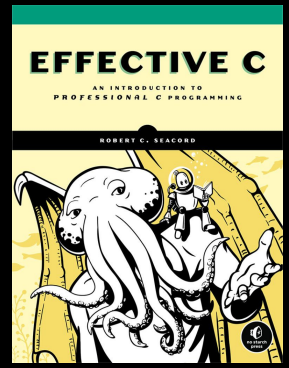


implementações em vetor.

R. Sedgewick, K. Wayne. Algorithms Part I. 4a ed. 2011



Seacord, R. C. Effective C: An ntroduction to Professional C Programming. 2020.



Licença