EDA0001 – Estruturas de Dados

Árvores, Árvores Binárias

Prof. Rui Jorge Tramontin Junior
Departamento de Ciência da Computação
UDESC / Joinville

Tópicos a serem apresentados

- Árvores;
  - Conceitos, aplicações, implementação;

Tópicos a serem apresentados

- Árvores;
  - Conceitos, aplicações, implementação;
- Árvores Binárias;
  - Conceitos, aplicações, implementação;

Tópicos a serem apresentados

- Árvores;
  - Conceitos, aplicações, implementação;
- Árvores Binárias;
  - Conceitos, aplicações, implementação;
- Árvores Binárias de Busca;
  - Aplicações e implementação;

Tópicos a serem apresentados

- Árvores;
  - Conceitos, aplicações, implementação;
- Árvores Binárias;
  - Conceitos, aplicações, implementação;
- Árvores Binárias de Busca;
  - Aplicações e implementação;
- Árvores AVL;
  - Conceitos e implementação.

Tópicos a serem apresentados

- **Árvores;**
  - **Conceitos, aplicações, implementação;**
- **Árvores Binárias;**
  - **Conceitos, aplicações, implementação;**
- Árvores Binárias de Busca;
  - Aplicações e implementação;
- Árvores AVL;
  - Conceitos e implementação.

Introdução

- *Vetores*, *listas*, *pilhas* e *filas* são estruturas lineares;

Introdução

- *Vetores*, *listas*, *pilhas* e *filas* são estruturas lineares;

- Por outro lado, **árvores** são estruturas hierárquicas;

Introdução

- *Vetores*, *listas*, *pilhas* e *filas* são estruturas lineares;
- Por outro lado, **árvores** são estruturas hierárquicas;
- Árvores são compostas por *nós* e *ligações* entre eles;

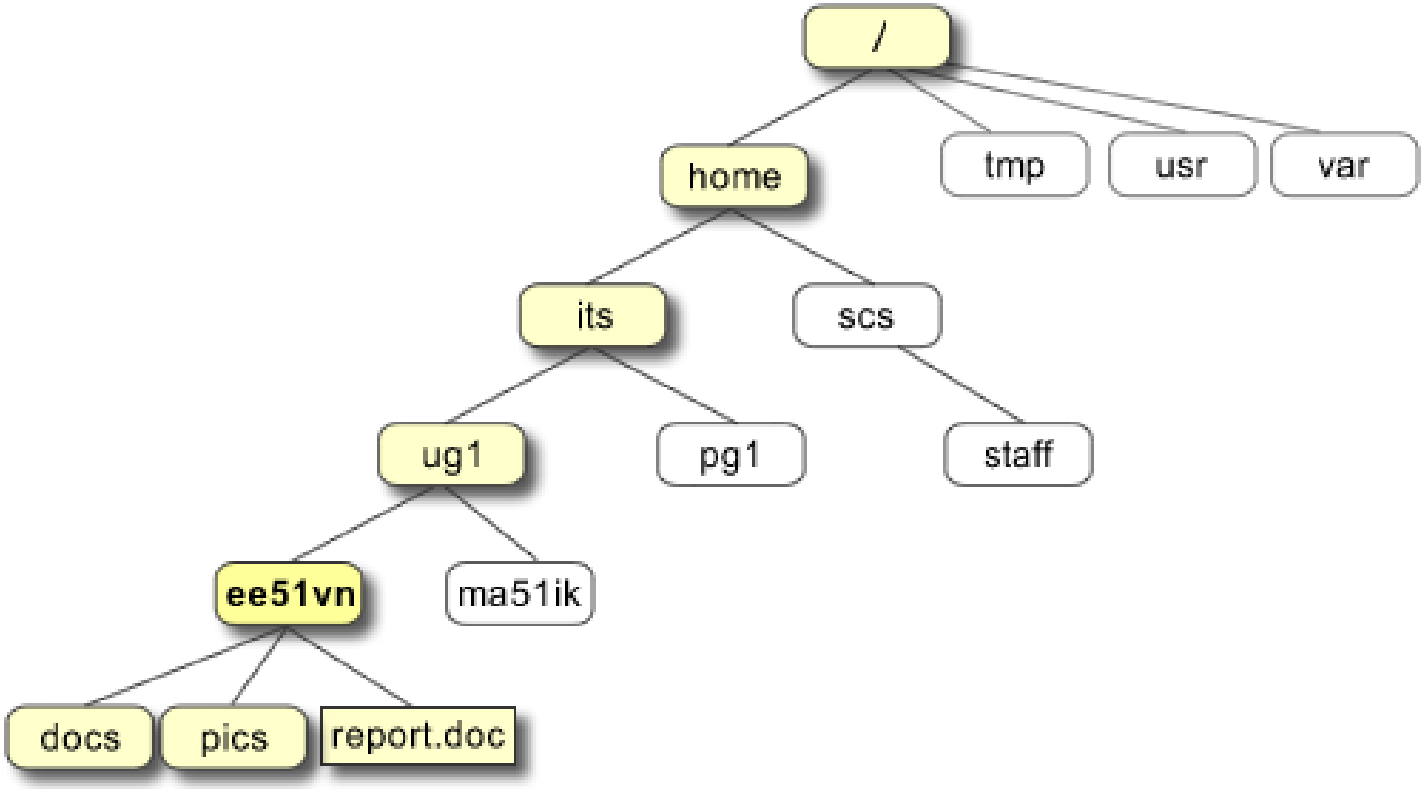
Introdução

- *Vetores*, *listas*, *pilhas* e *filas* são estruturas lineares;
- Por outro lado, **árvores** são estruturas hierárquicas;
- Árvores são compostas por *nós* e *ligações* entre eles;
  - Cada nó possui somente um pai e zero ou mais filhos;

Introdução

- *Vetores*, *listas*, *pilhas* e *filas* são estruturas lineares;
- Por outro lado, **árvores** são estruturas hierárquicas;
- Árvores são compostas por *nós* e *ligações* entre eles;
  - Cada nó possui somente um pai e zero ou mais filhos;
- Uma árvore pode ser vista como um grafo acíclico;

Exemplo: estrutura de diretórios



Aplicações

- **Armazenamento de dados hierárquicos**: sistemas de arquivos/diretórios, arquivos XML/HTML;

Aplicações

- **Armazenamento de dados hierárquicos**: sistemas de arquivos/diretórios, arquivos XML/HTML;

- **Compiladores**: análise sintática de linguagens de programação;

Aplicações

- **Armazenamento de dados hierárquicos**: sistemas de arquivos/diretórios, arquivos XML/HTML;
- **Compiladores**: análise sintática de linguagens de programação;
- **Indexadores de banco de dados**: por exemplo, Árvores B;

Aplicações

- **Inteligência Artificial**: por exemplo, árvores de decisão (mineração de dados);

Aplicações

- **Inteligência Artificial**: por exemplo, árvores de decisão (mineração de dados);

- **Árvores Binárias**:

Aplicações

- **Inteligência Artificial**: por exemplo, árvores de decisão (mineração de dados);

- **Árvores Binárias**:
  - Busca;

Aplicações

- **Inteligência Artificial**: por exemplo, árvores de decisão (mineração de dados);

- **Árvores Binárias**:
  - Busca;
  - Ordenação;

Aplicações

- **Inteligência Artificial**: por exemplo, árvores de decisão (mineração de dados);

- **Árvores Binárias**:
  - Busca;
  - Ordenação;
  - Filas de prioridade;

Aplicações

- **Inteligência Artificial**: por exemplo, árvores de decisão (mineração de dados);

- **Árvores Binárias**:
  - Busca;
  - Ordenação;
  - Filas de prioridade;
  - Algoritmos em grafos;

Aplicações

- **Inteligência Artificial**: por exemplo, árvores de decisão (mineração de dados);

- **Árvores Binárias**:
  - Busca;
  - Ordenação;
  - Filas de prioridade;
  - Algoritmos em grafos;
  - Entre outros... (mais detalhes em breve).

Definição de Árvore

- Definição recursiva:

Definição de Árvore

- Definição recursiva:
  - Conjunto de **nós**;

Definição de Árvore

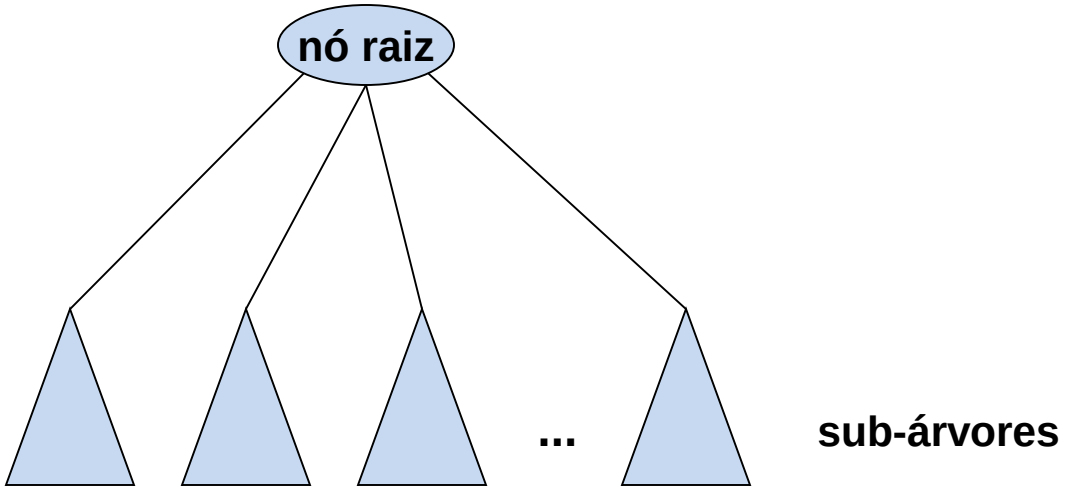
- Definição recursiva:
  - Conjunto de **nós**;
  - Existe um nó *r* denominado ***raiz***, com zero ou mais **sub-árvores**, cujas raízes são ligadas diretamente a *r*;

Definição de Árvore

- Definição recursiva:
  - Conjunto de **nós**;
  - Existe um nó *r* denominado ***raiz***, com zero ou mais **sub-árvores**, cujas raízes são ligadas diretamente a *r*;
  - Os nós raízes das sub árvores são ditos **filhos** de *r*.

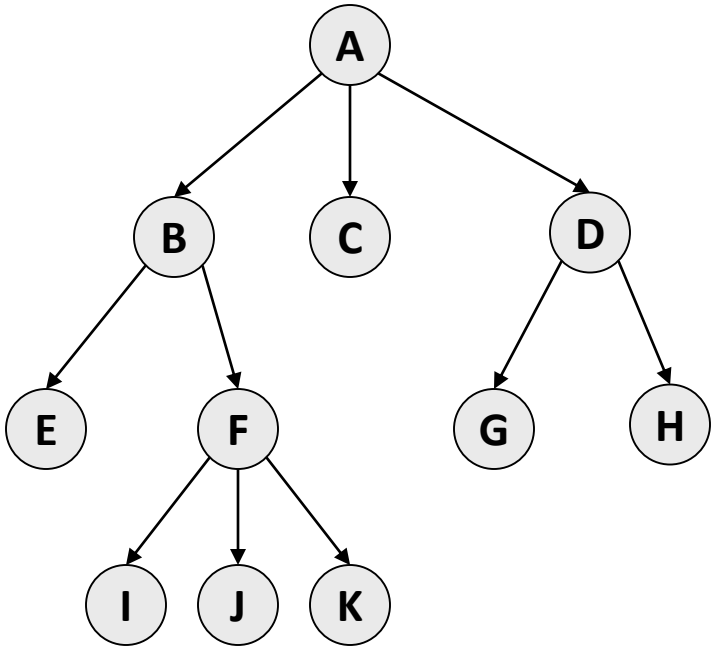
Definição de Árvore

- Definição recursiva:
  - Conjunto de **nós**;
  - Existe um nó *r* denominado ***raiz***, com zero ou mais **sub-árvores**, cujas raízes são ligadas diretamente a *r*;
  - Os nós raízes das sub árvores são ditos **filhos** de *r*.



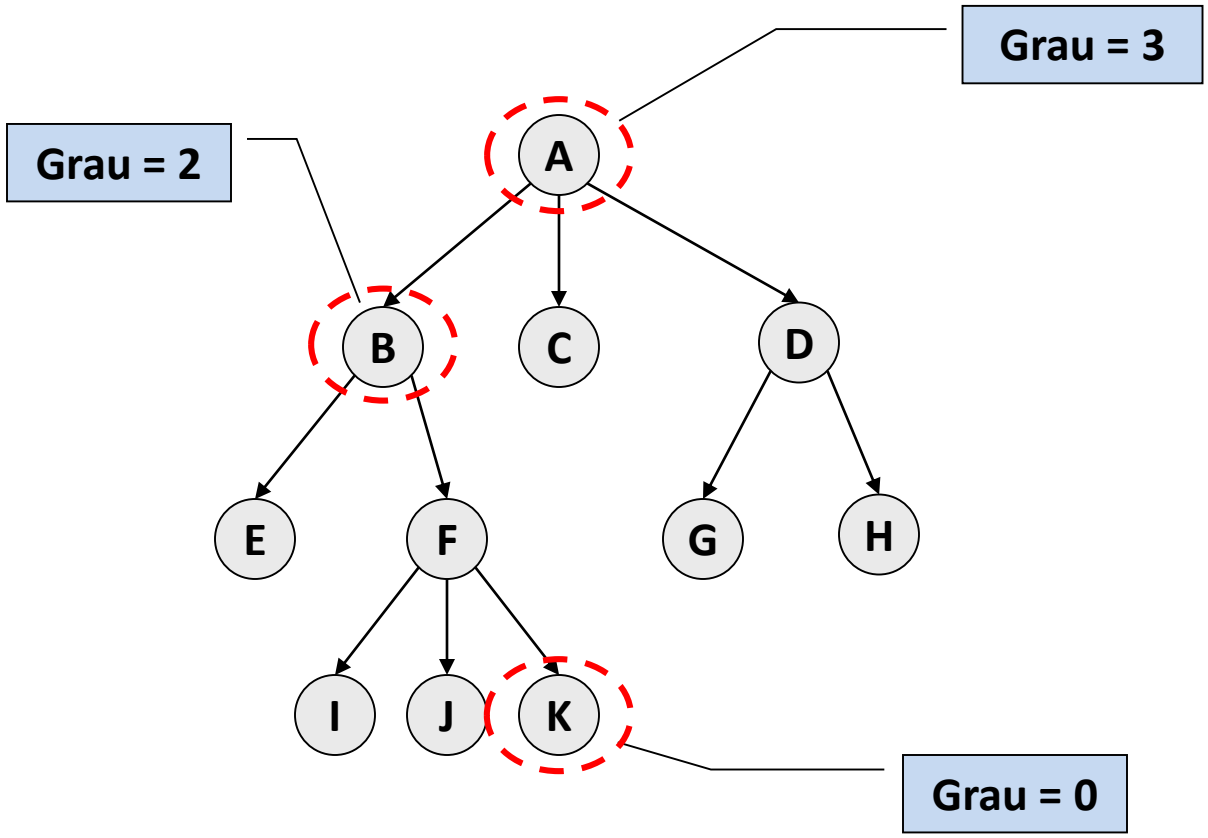
Conceitos Básicos

- **Grau de saída**: número de filhos (sub-árvores) de um nó;



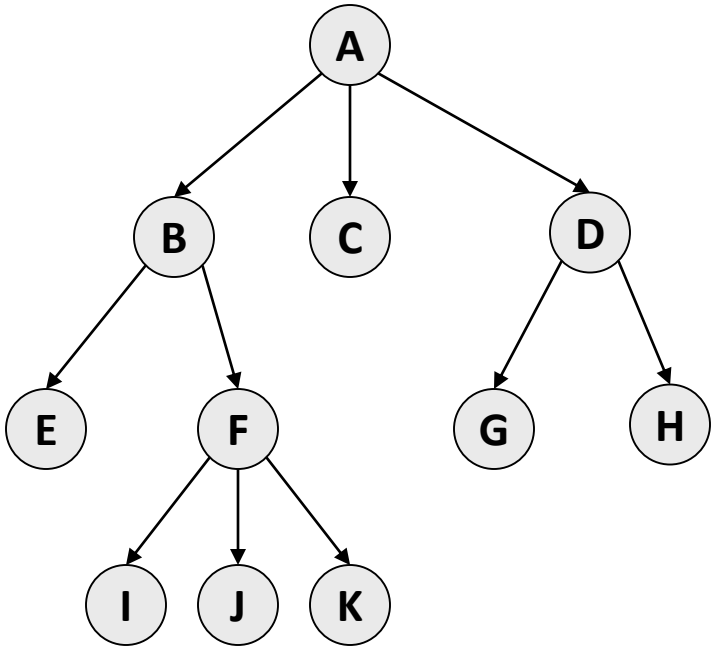
Conceitos Básicos

- **Grau de saída**: número de filhos (sub-árvores) de um nó;



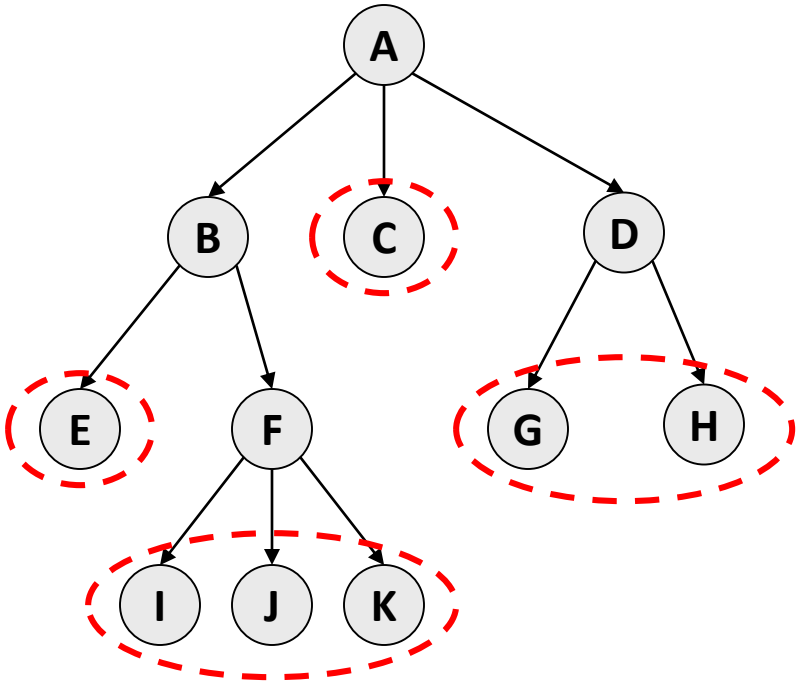
Conceitos Básicos

- **Nó folha**: grau de saída = 0;



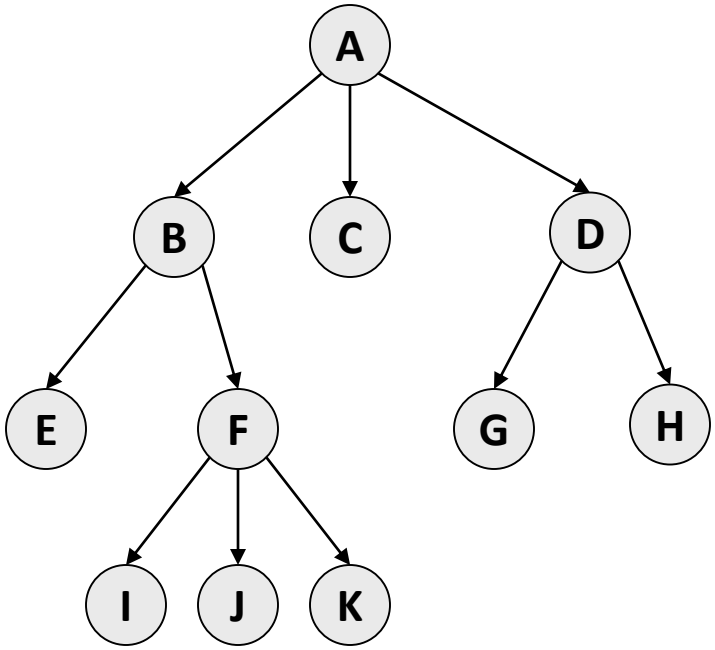
Conceitos Básicos

- **Nó folha**: grau de saída = 0;



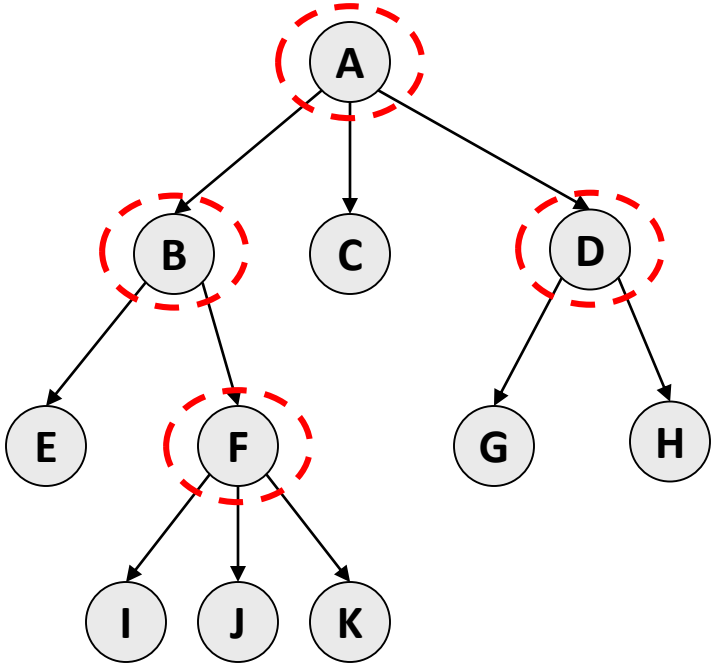
Conceitos Básicos

- **Nó interno**: grau de saída > 0;



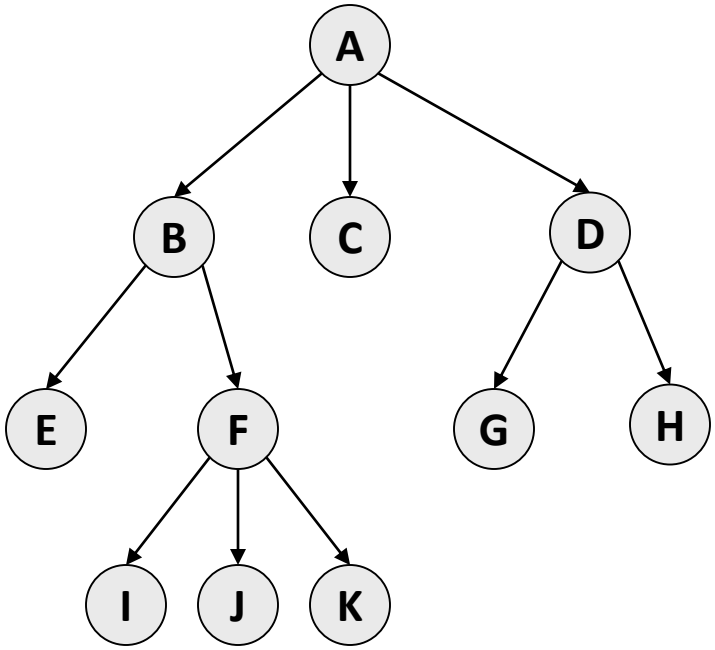
Conceitos Básicos

- **Nó interno**: grau de saída > 0;



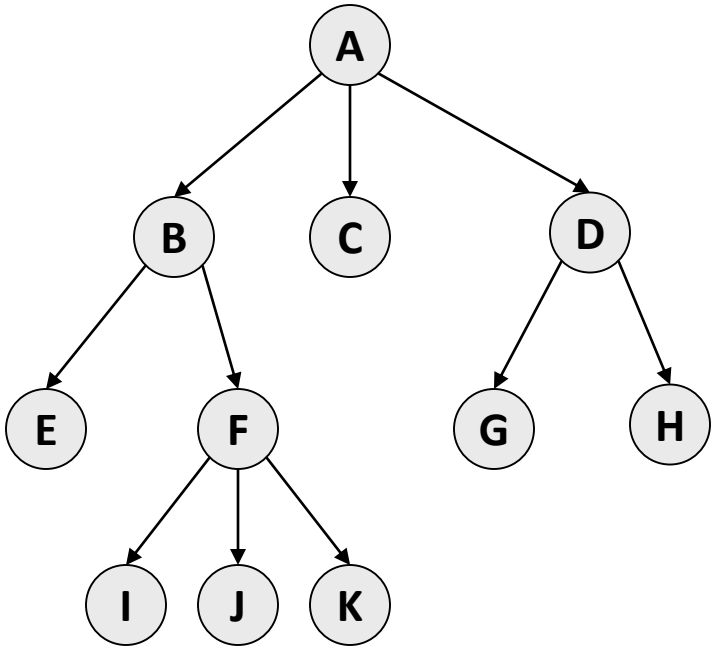
Conceitos Básicos

- **Caminho**: sequência de nós que tenham relação pai-filho;



Conceitos Básicos

- **Caminho**: sequência de nós que tenham relação pai-filho;



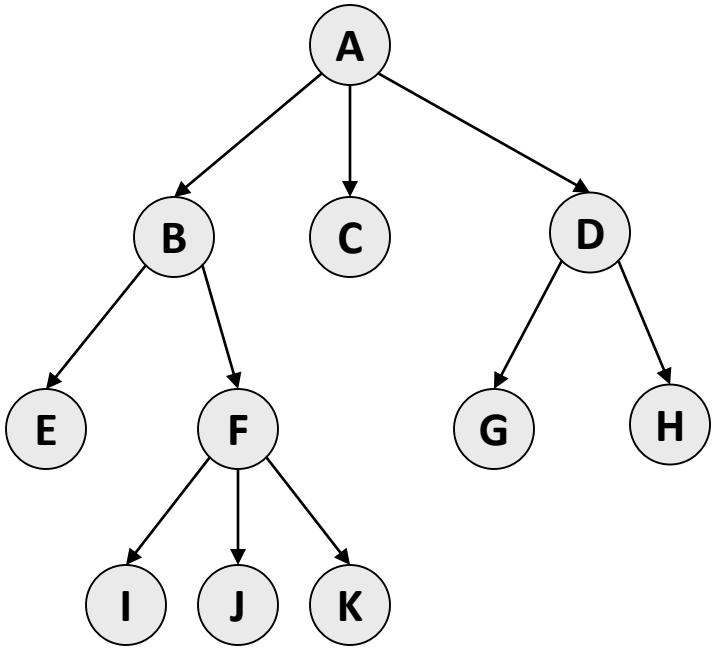
Exemplos:

A→B→F→K

A→D→G

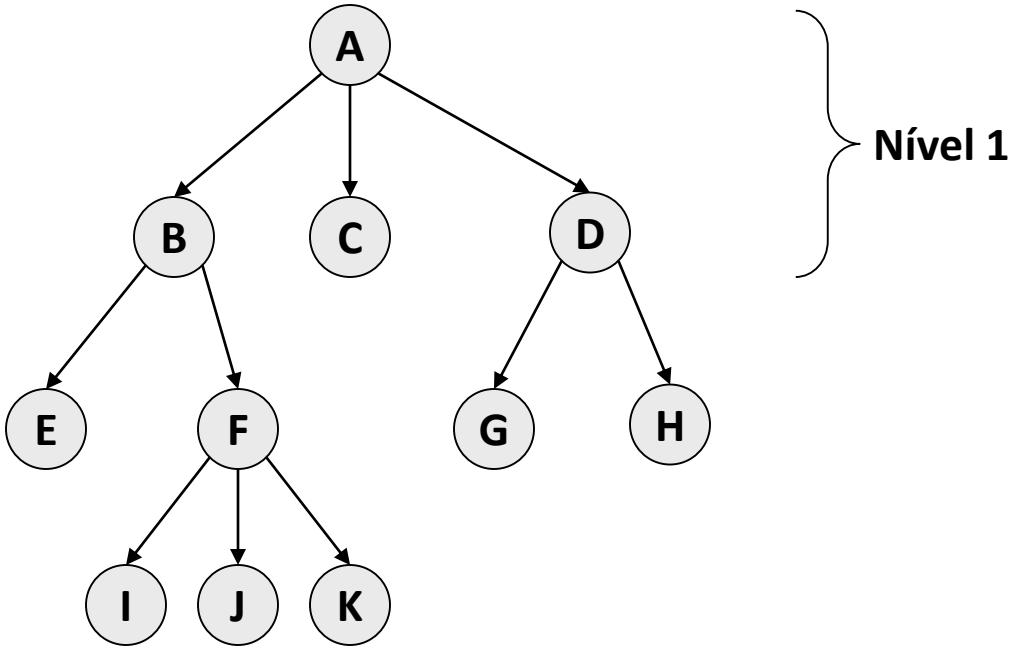
Conceitos Básicos

- **Nível de um nó**: número de nós no caminho até *r*;



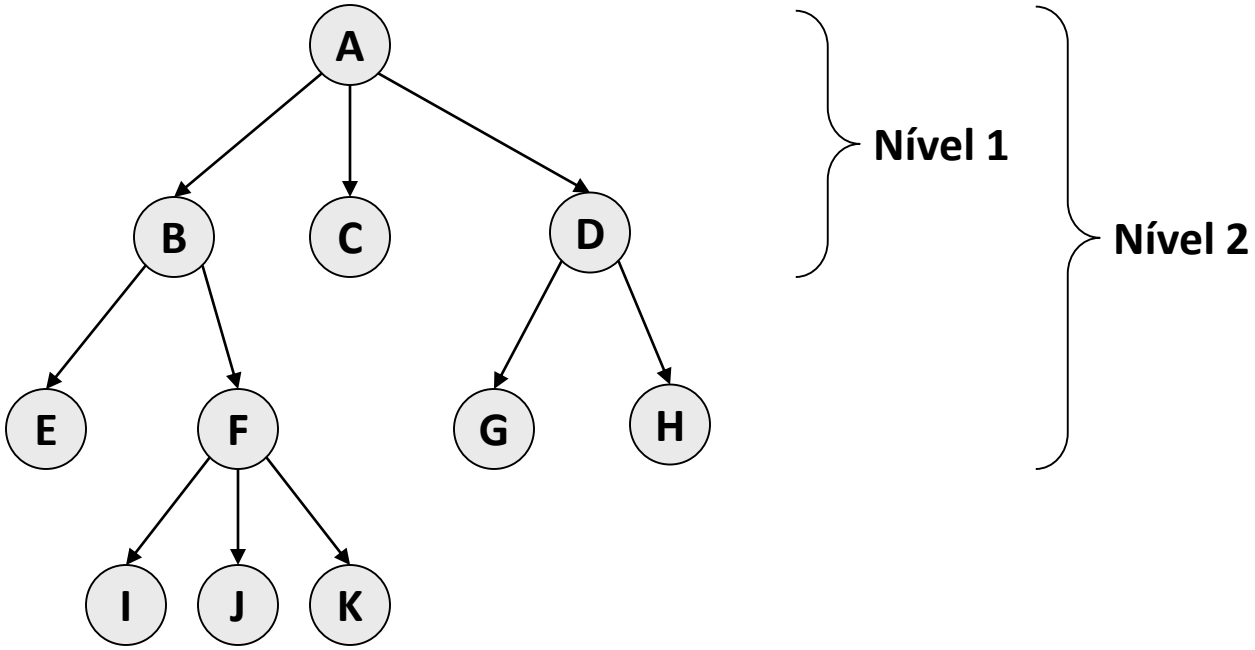
Conceitos Básicos

- **Nível de um nó**: número de nós no caminho até *r*;



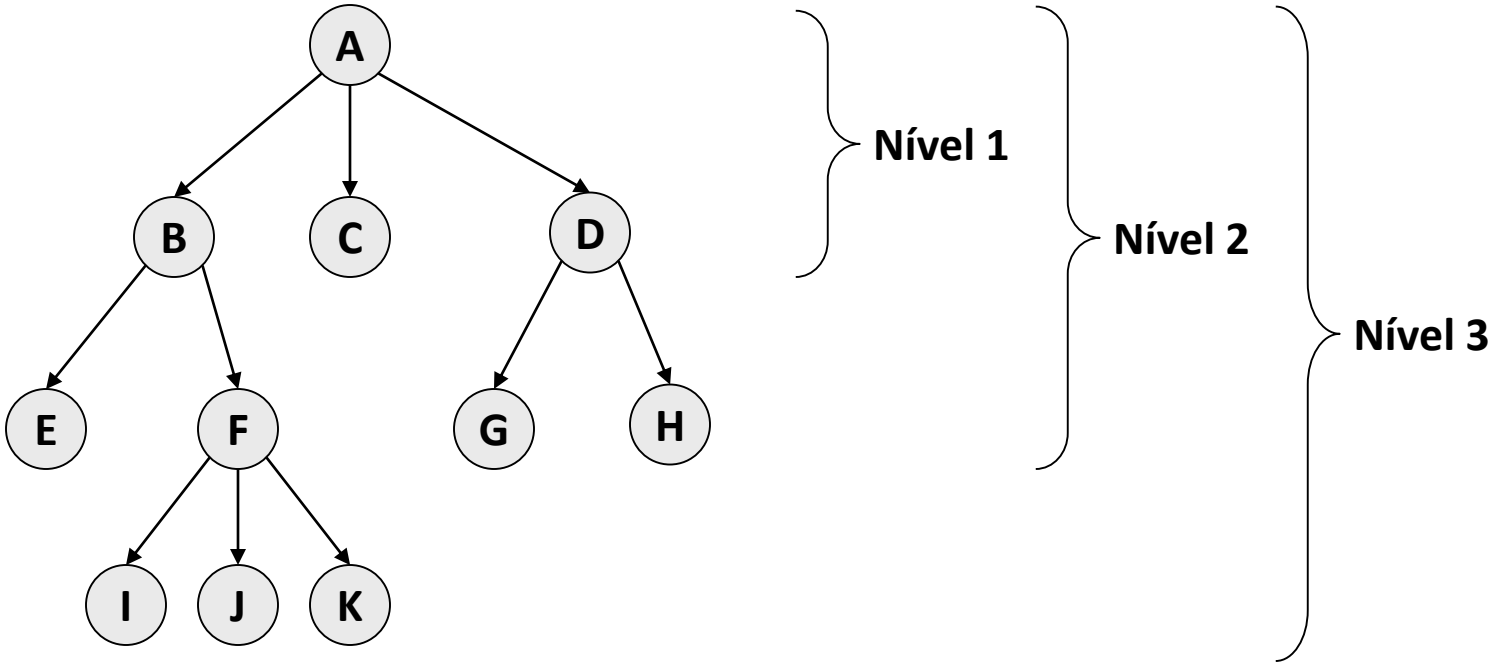
Conceitos Básicos

- **Nível de um nó**: número de nós no caminho até *r*;



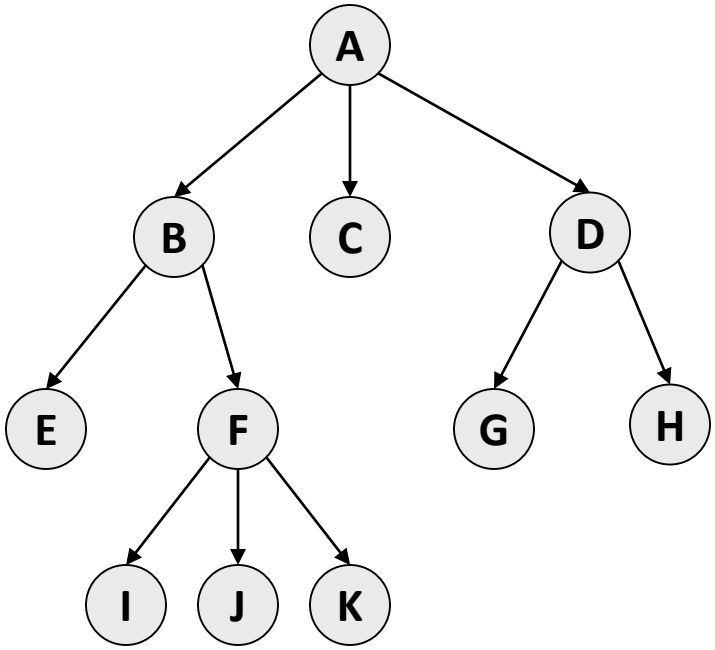
Conceitos Básicos

- **Nível de um nó**: número de nós no caminho até *r*;



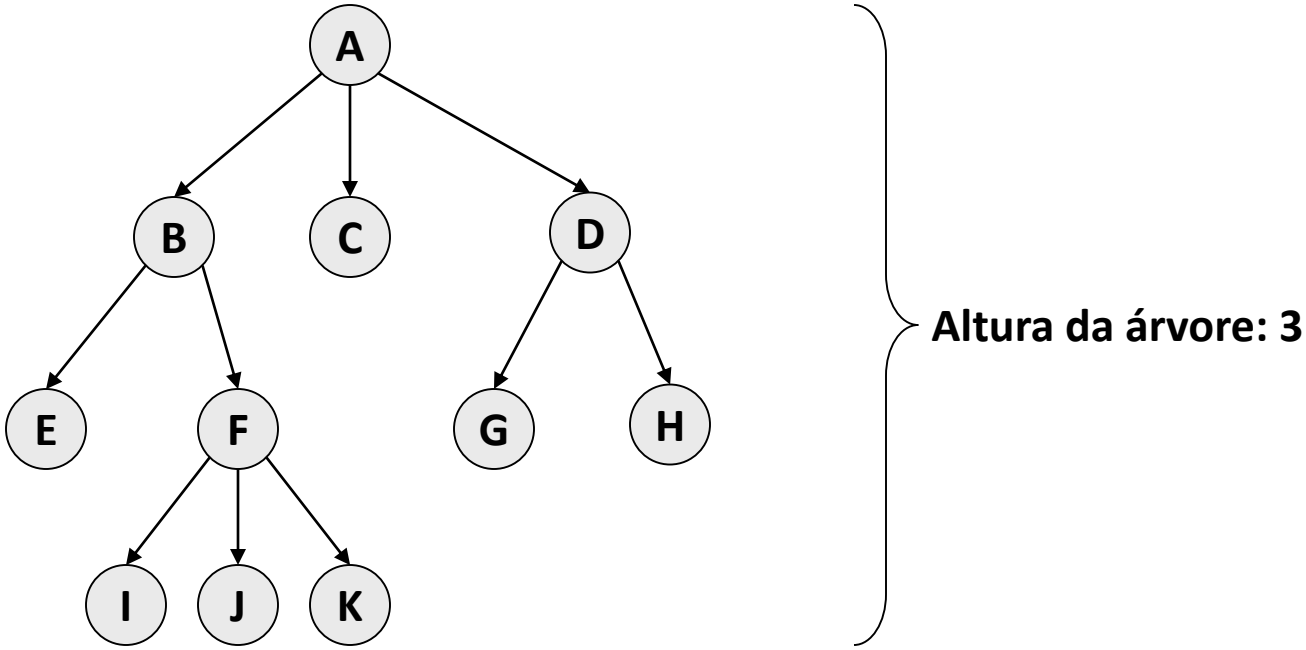
Conceitos Básicos

- **Altura de árvore**: maior nível dentre seus nós;



Conceitos Básicos

- **Altura de árvore**: maior nível dentre seus nós;



Conceitos Básicos

- **Grau de saída**: número de filhos (sub-árvores) de um nó;
- **Nó folha**: grau de saída = 0;
- **Nó interno**: grau de saída > 0;
- **Caminho**: sequência de nós que tenham relação pai-filho;
- **Nível de um nó**: número de nós no caminho até *r*;
- **Altura de árvore**: maior nível dentre seus nós.

Representação Computacional de uma Árvore

- A implementação de uma árvore consiste em representar os ***nós*** da seguinte forma:

Representação Computacional de uma Árvore

- A implementação de uma árvore consiste em representar os ***nós*** da seguinte forma:
  - Campo para a informação;

Representação Computacional de uma Árvore

- A implementação de uma árvore consiste em representar os ***nós*** da seguinte forma:
  - Campo para a informação;
  - Ponteiro para o primeiro filho (cabeça da lista de filhos);

Representação Computacional de uma Árvore

- A implementação de uma árvore consiste em representar os ***nós*** da seguinte forma:
  - Campo para a informação;
  - Ponteiro para o primeiro filho (cabeça da lista de filhos);
  - Ponteiro para o próximo irmão (da lista);

Representação Computacional de uma Árvore

- A implementação de uma árvore consiste em representar os ***nós*** da seguinte forma:
  - Campo para a informação;
  - Ponteiro para o primeiro filho (cabeça da lista de filhos);
  - Ponteiro para o próximo irmão (da lista);

- O nó raiz não tem irmãos;

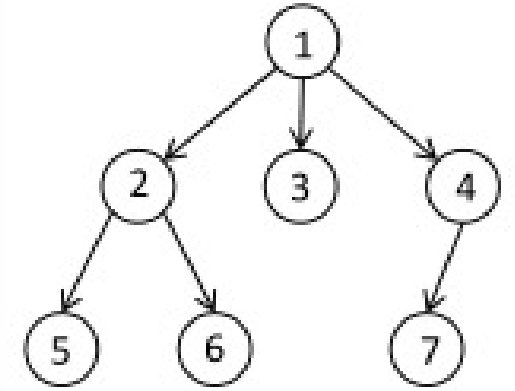
Representação Computacional de uma Árvore

- A implementação de uma árvore consiste em representar os ***nós*** da seguinte forma:
  - Campo para a informação;
  - Ponteiro para o primeiro filho (cabeça da lista de filhos);
  - Ponteiro para o próximo irmão (da lista);

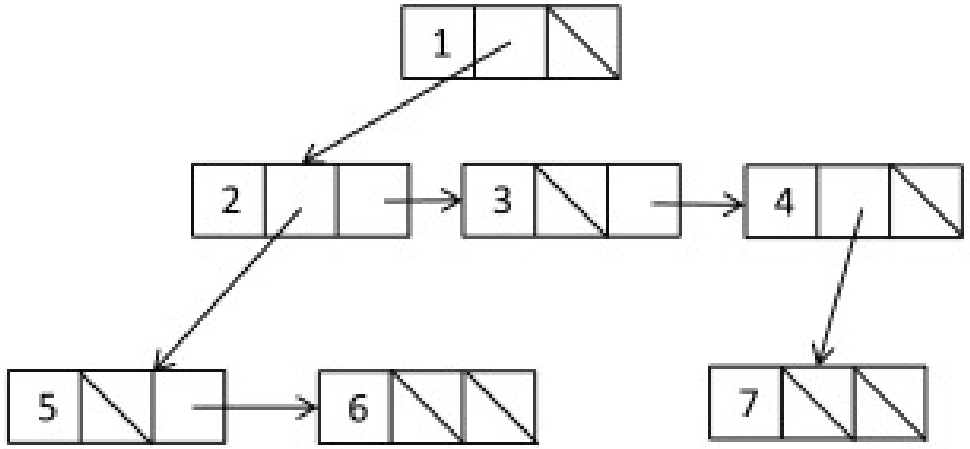
- O nó raiz não tem irmãos;

- Os nós folhas não têm filhos.

Representação Computacional de uma Árvore

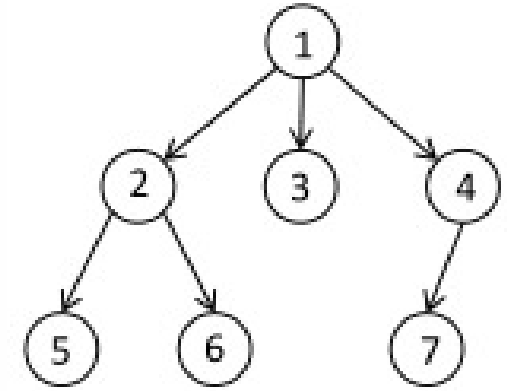


Tree Structure

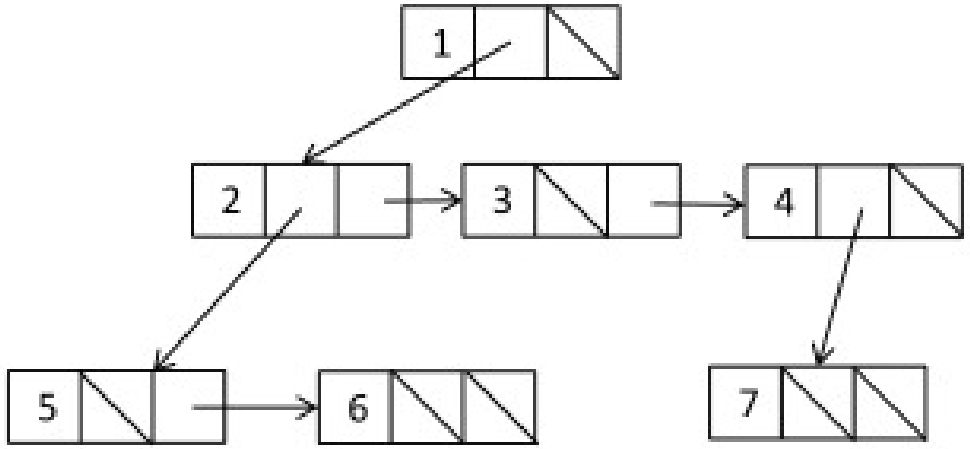


Linked List representation of A Tree

Representação Computacional de uma Árvore



Tree Structure



Linked List representation of A Tree

```
typedef struct noArvore{
   int info;
   struct noArvore *primeiro_filho;
   struct noArvore *proximo_irmao;
}NoArvore;
```

ÁRVORES BINÁRIAS

Árvore Binária

- **Definição**: é a árvore onde todos os seus nós têm no máximo 2 filhos (grau ≤ 2);

Árvore Binária

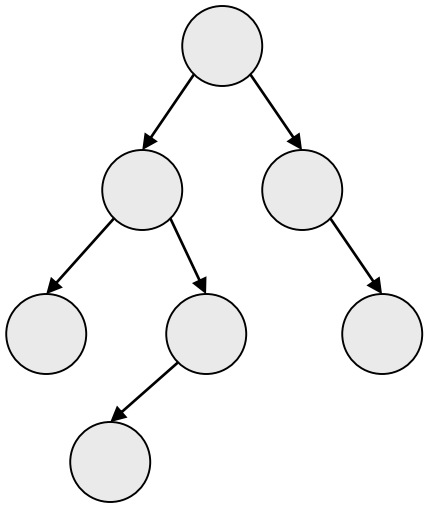
- **Definição**: é a árvore onde todos os seus nós têm no máximo 2 filhos (grau ≤ 2);

- Os filhos de um nó são tipicamente referenciados com ponteiros (*esquerda* e *direita*).

Árvore Binária

- **Definição**: é a árvore onde todos os seus nós têm no máximo 2 filhos (grau ≤ 2);

- Os filhos de um nó são tipicamente referenciados com ponteiros (*esquerda* e *direita*).



Aplicações de Árvore Binárias

- Árvores Binárias de Busca (ABB);
- *Binary Space Partition* (BSP);
- Árvore de Prefixos (*Radix Tree*);
- *Heap* binário;
- Codificação de *Huffman*;
- Árvores T (T-tree).

Aplicações de Árvore Binárias

- **Árvores Binárias de Busca (ABB)**: usadas em problemas que necessitem de busca, onde dados são inseridos e removidos constantemente;

Aplicações de Árvore Binárias

- **Árvores Binárias de Busca (ABB)**: usadas em problemas que necessitem de busca, onde dados são inseridos e removidos constantemente;
  - São usadas para implementar classes do tipo *Map* e *Set* em bibliotecas de linguagens de programação;

Aplicações de Árvore Binárias

- **Árvores Binárias de Busca (ABB)**: usadas em problemas que necessitem de busca, onde dados são inseridos e removidos constantemente;
  - São usadas para implementar classes do tipo *Map* e *Set* em bibliotecas de linguagens de programação;

- **Binary Space Partition (BSP)**: o *particionamento binário de espaço* é um método recursivo que consiste em divisões sucessivas de um espaço, dando origem a uma árvore BSP;

Aplicações de Árvore Binárias

- **Árvores Binárias de Busca (ABB)**: usadas em problemas que necessitem de busca, onde dados são inseridos e removidos constantemente;
  - São usadas para implementar classes do tipo *Map* e *Set* em bibliotecas de linguagens de programação;

- **Binary Space Partition (BSP)**: o *particionamento binário de espaço* é um método recursivo que consiste em divisões sucessivas de um espaço, dando origem a uma árvore BSP;
  - Quando aplicado em computação gráfica em 3D, torna-se útil no processo de renderização dos objetos de uma cena.

Aplicações de Árvore Binárias

- **Árvore de Prefixos (*Radix Tree*)**: organizam os dados em partes e são normalmente usadas em:

Aplicações de Árvore Binárias

- **Árvore de Prefixos (*Radix Tree*)**: organizam os dados em partes e são normalmente usadas em:
  - Roteadores IP (tabelas de roteamento);

Aplicações de Árvore Binárias

- **Árvore de Prefixos (*Radix Tree*)**: organizam os dados em partes e são normalmente usadas em:
  - Roteadores IP (tabelas de roteamento);
  - Motores de busca (listas invertidas - *inverted indexes*);

Aplicações de Árvore Binárias

- **Árvore de Prefixos (*Radix Tree*)**: organizam os dados em partes e são normalmente usadas em:
  - Roteadores IP (tabelas de roteamento);
  - Motores de busca (listas invertidas - *inverted indexes*);

- ***Heap* binário**: usado para a implementação eficiente de filas de prioridades; é usado também:

Aplicações de Árvore Binárias

- **Árvore de Prefixos (*Radix Tree*)**: organizam os dados em partes e são normalmente usadas em:
  - Roteadores IP (tabelas de roteamento);
  - Motores de busca (listas invertidas - *inverted indexes*);

- ***Heap* binário**: usado para a implementação eficiente de filas de prioridades; é usado também:
  - no algoritmo de ordenação *HeapSort*;

Aplicações de Árvore Binárias

- **Árvore de Prefixos (*Radix Tree*)**: organizam os dados em partes e são normalmente usadas em:
  - Roteadores IP (tabelas de roteamento);
  - Motores de busca (listas invertidas - *inverted indexes*);

- ***Heap* binário**: usado para a implementação eficiente de filas de prioridades; é usado também:
  - no algoritmo de ordenação *HeapSort*;
  - em alguns algortimos para grafos, tal como do algortimo de *Dijkstra* (que determina o caminho mais curto em um grafo);

Aplicações de Árvore Binárias

- **Codificação de *Huffman***: algoritmo de compressão que constrói recursivamente uma árvore binária;

Aplicações de Árvore Binárias

- **Codificação de *Huffman***: algoritmo de compressão que constrói recursivamente uma árvore binária;
  - Aplicado em diversos formatos de compressão sem perda, tais como: GZIP, PKZIP, BZIP2, JPEG e PNG.

Aplicações de Árvore Binárias

- **Codificação de *Huffman***: algoritmo de compressão que constrói recursivamente uma árvore binária;
  - Aplicado em diversos formatos de compressão sem perda, tais como: GZIP, PKZIP, BZIP2, JPEG e PNG.

- **Árvores T (*T-tree*)**: usadas para indexar bancos de dados armazenados em memória;

Aplicações de Árvore Binárias

- **Codificação de *Huffman***: algoritmo de compressão que constrói recursivamente uma árvore binária;
  - Aplicado em diversos formatos de compressão sem perda, tais como: GZIP, PKZIP, BZIP2, JPEG e PNG.

- **Árvores T (*T-tree*)**: usadas para indexar bancos de dados armazenados em memória;
  - De maneira similar a *Árvores B* que indexam dados em memória secundária (disco);

Implementação de Árvores Binárias

- Estrutura do nó contém:
  - Informação;
  - Ponteiros para os filhos à esquerda e à direita.

Implementação de Árvores Binárias

- Estrutura do nó contém:
  - Informação;
  - Ponteiros para os filhos à esquerda e à direita.

```
typedef struct noAB {
    int info;
    struct noAB *esq;
    struct noAB *dir;
}NoAB;
```

Implementação com vetores (1/2)

- Em algumas aplicações, tal como o ***Heap binário***, a árvore é implementada de maneira eficiente utilizando um vetor;

Implementação com vetores (1/2)

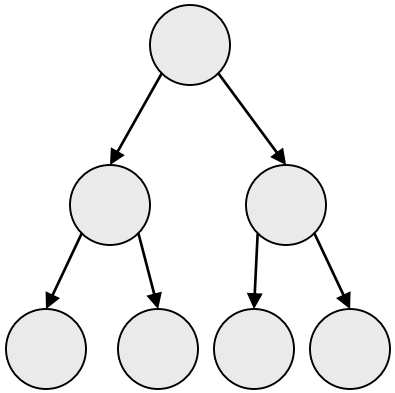
- Em algumas aplicações, tal como o ***Heap binário***, a árvore é implementada de maneira eficiente utilizando um vetor;

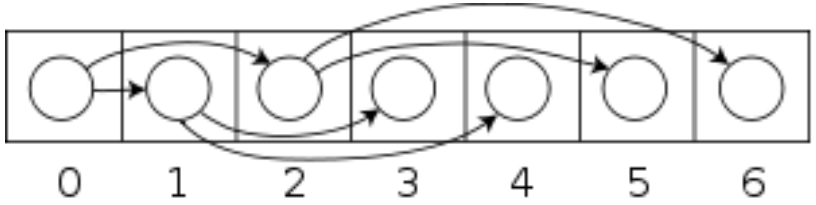
- Nesse tipo de aplicação, a árvore normalmente é dita ***cheia***, ou seja, todos os *nós folha* estão *no mesmo nível*;

Implementação com vetores (1/2)

- Em algumas aplicações, tal como o ***Heap binário***, a árvore é implementada de maneira eficiente utilizando um vetor;

- Nesse tipo de aplicação, a árvore normalmente é dita ***cheia***, ou seja, todos os *nós folha* estão *no mesmo nível*;





Implementação com vetores (2/2)

- Não é preciso utilizar ponteiros para localizas os nós filhos;

Implementação com vetores (2/2)

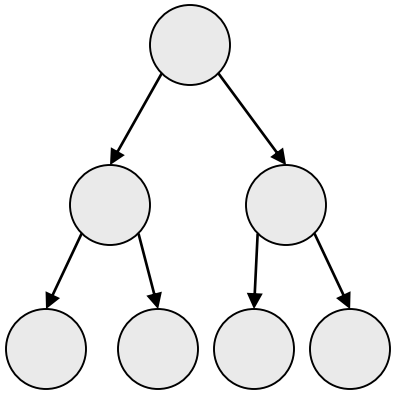
- Não é preciso utilizar ponteiros para localizas os nós filhos;

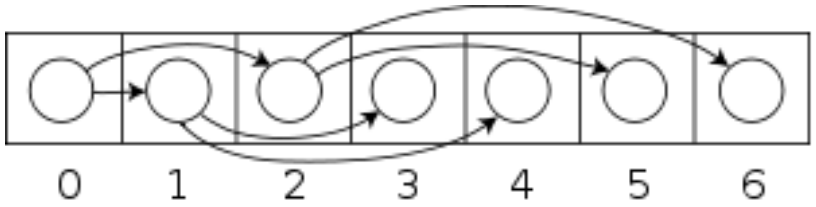
- Para cada nó *i*, seu filho à esquerda está em 2*i* + 1 e seu filho à direita está em 2*i* + 2.

Implementação com vetores (2/2)

- Não é preciso utilizar ponteiros para localizas os nós filhos;

- Para cada nó *i*, seu filho à esquerda está em 2*i* + 1 e seu filho à direita está em 2*i* + 2.





Próximo tópico de estudo

- **Árvores Binárias de Busca**!