INSTITUTO FEDERAL
Sudeste de Minas Gerais
Campus Manhuaçu

Estruturas de Dados I
Introdução a árvores

Prof. Leonardo C. R. Soares - *leonardo.soares@ifsudestemg.edu.br*
Instituto Federal do Sudeste de Minas Gerais

21 de fevereiro de 2025

Árvores

Descrição

- Árvores são estruturas de dados não lineares. Os elementos são armazenados de forma hierárquica. Com exceção do elemento do topo, cada elemento da árvore tem um elemento **pai** e zero ou mais elementos **filhos**;

Árvores

Descrição

- Árvores são estruturas de dados não lineares. Os elementos são armazenados de forma hierárquica. Com exceção do elemento do topo, cada elemento da árvore tem um elemento **pai** e zero ou mais elementos **filhos**;
- Cada elemento da árvore é chamado de **nodo**;

Árvores

Descrição

- Árvores são estruturas de dados não lineares. Os elementos são armazenados de forma hierárquica. Com exceção do elemento do topo, cada elemento da árvore tem um elemento **pai** e zero ou mais elementos **filhos**;
- Cada elemento da árvore é chamado de **nodo**;
- O **nodo** do qual todos os outros descendem é **a raiz da árvore**.

Árvores

Descrição

- Se um nodo u é pai de um novo v, então dizemos que v é **filho** de u;

Árvores

Descrição

- Se um nodo u é pai de um novo v, então dizemos que v é **filho** de u;
- Dois nodos que são filhos de um mesmo pai são irmãos;

Árvores

Descrição

- Se um nodo u é pai de um novo v, então dizemos que v é **filho** de u;
- Dois nodos que são filhos de um mesmo pai são irmãos;
- Um nodo é **externo** (ou folha) quando *não* possui filhos;

Árvores

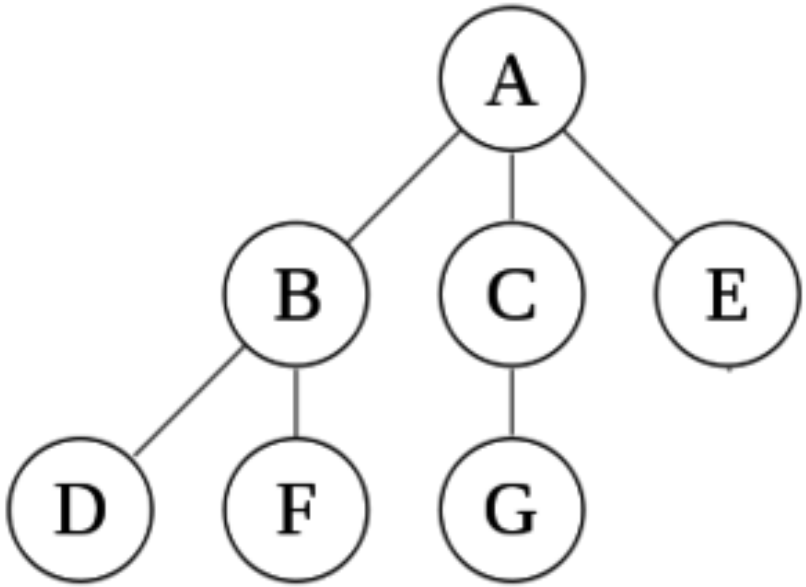
Descrição

- Se um nodo u é pai de um novo v, então dizemos que v é **filho** de u;
- Dois nodos que são filhos de um mesmo pai são irmãos;
- Um nodo é **externo** (ou folha) quando *não* possui filhos;
- Um nodo é *interno* se tem um ou mais filhos;

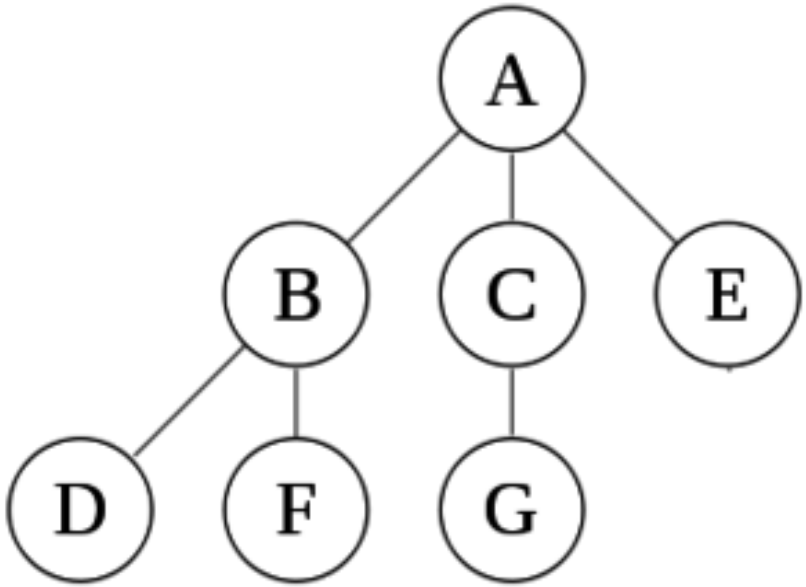
Árvores

Descrição

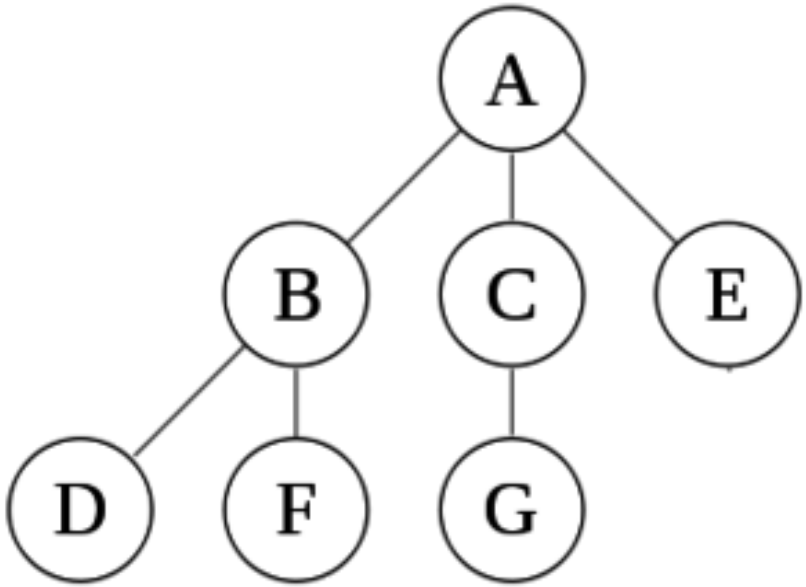
- Se um nodo u é pai de um novo v, então dizemos que v é **filho** de u;
- Dois nodos que são filhos de um mesmo pai são irmãos;
- Um nodo é **externo** (ou folha) quando *não* possui filhos;
- Um nodo é *interno* se tem um ou mais filhos;
- Um **ancestral** de um nodo é tanto um ancestral direto como um ancestral do pai do nodo. De forma inversa, v é **descendente** de u se u é ancestral de v.



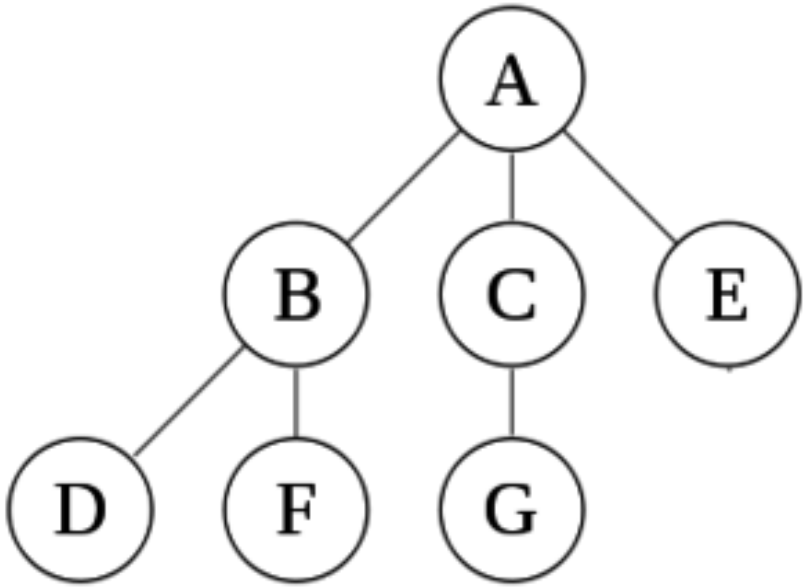
- O nodo A é a raiz da árvore;



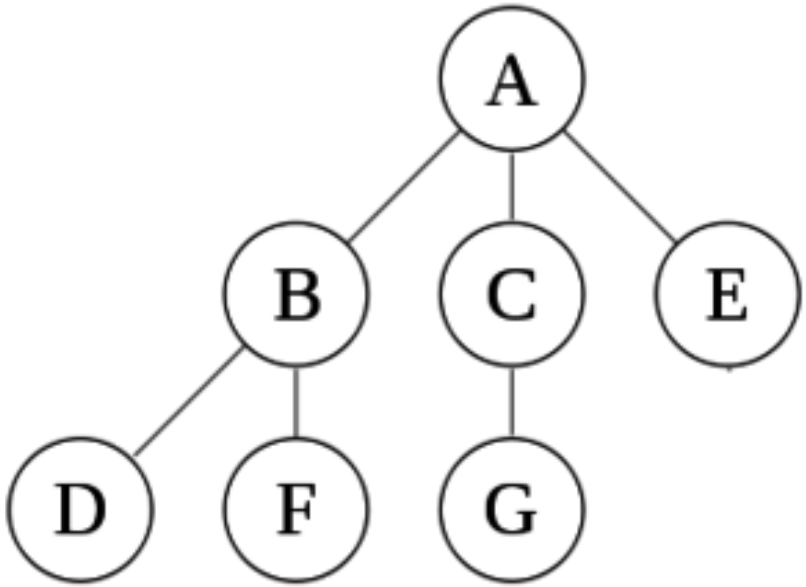
- O nodo A é a raiz da árvore;
- A é pai de B, C e E;
- B, C e E são filhos diretos de A;



- O nodo A é a raiz da árvore;
- A é pai de B, C e E;
- B, C e E são filhos diretos de A;
- B é pai de D e F
- D e F são filhos de B;



- O nodo A é a raiz da árvore;
- A é pai de B, C e E;
- B, C e E são filhos diretos de A;
- B é pai de D e F
- D e F são filhos de B;
- C é pai de G;
- G é filho de C;



- O nodo A é a raiz da árvore;
- A é pai de B, C e E;
- B, C e E são filhos diretos de A;
- B é pai de D e F
- D e F são filhos de B;
- C é pai de G;
- G é filho de C;
- D, E, F e G são nodos externos (folhas);

Descrição

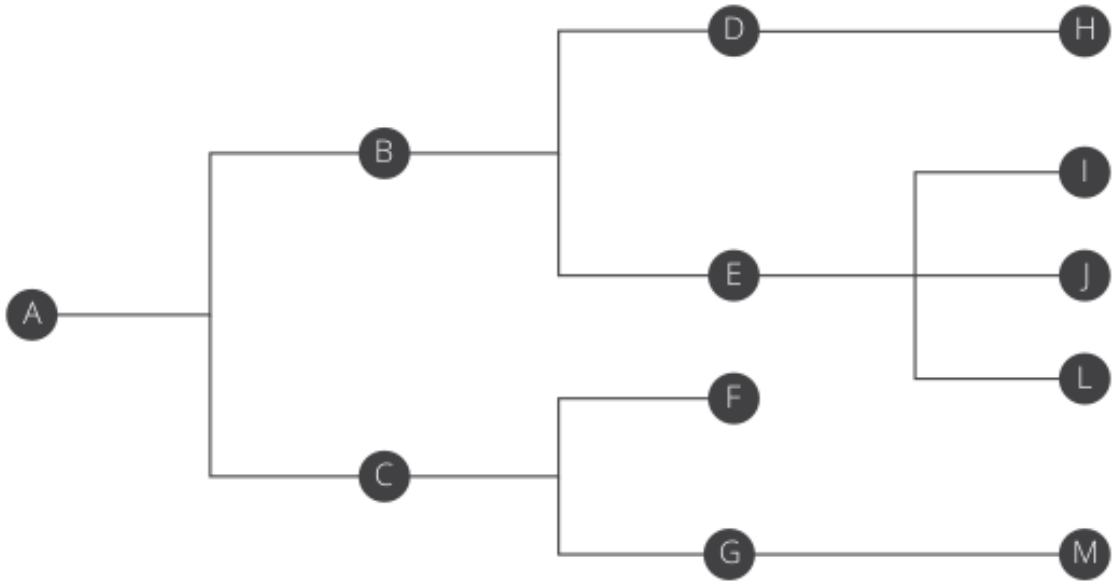
- A **altura** de uma árvore é o número de níveis da raiz até a folha mais distante;

Descrição

- A **altura** de uma árvore é o número de níveis da raiz até a folha mais distante;
- Uma árvore composta apenas pela raiz tem altura igual a 0;

Descrição

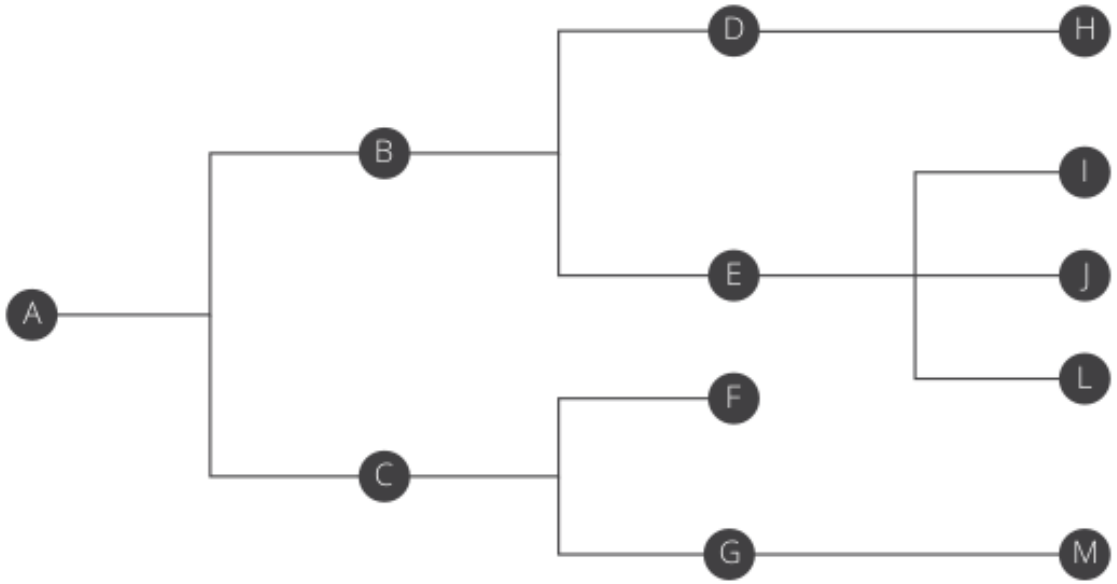
- A **altura** de uma árvore é o número de níveis da raiz até a folha mais distante;
- Uma árvore composta apenas pela raiz tem altura igual a 0;
- A altura de um nodo é a maior distância entre esse nó e uma de suas folhas.



Qual a altura dessa árvore?

Descrição

- A **altura** de uma árvore é o número de níveis da raiz até a folha mais distante;
- Uma árvore composta apenas pela raiz tem altura igual a 0;
- A altura de um nodo é a maior distância entre esse nó e uma de suas folhas.



Qual a altura dessa árvore? **3**

Descrição

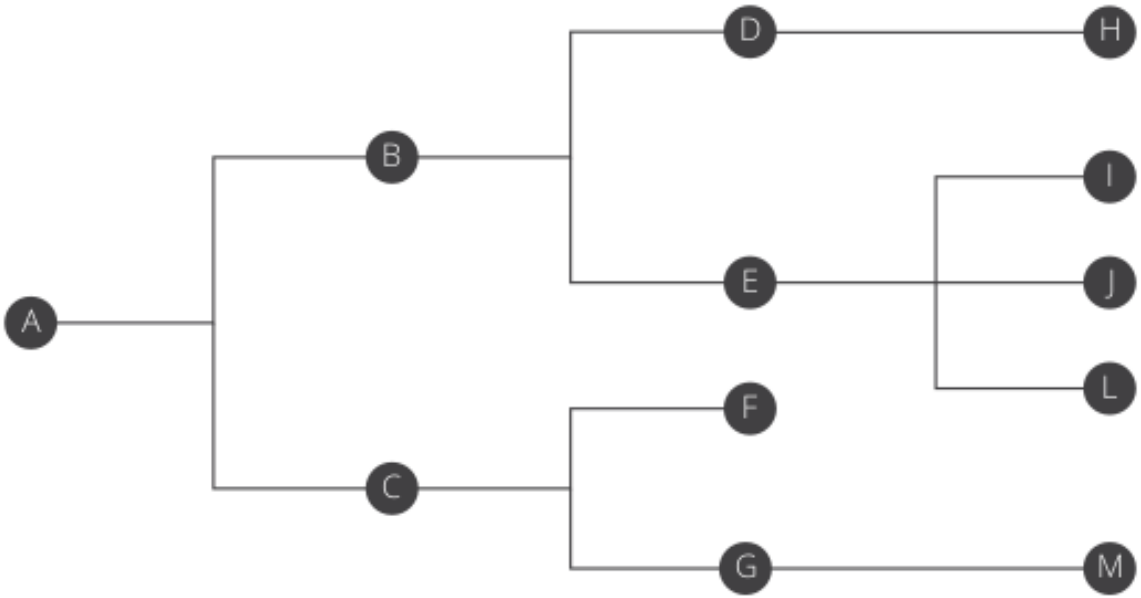
- O **grau de uma árvore** (ou ordem) é o número máximo de filhos que um nodo possui;

Descrição

- O **grau de uma árvore** (ou ordem) é o número máximo de filhos que um nodo possui;
- O **grau de um nodo** é o número de filhos que ele possui.

Descrição

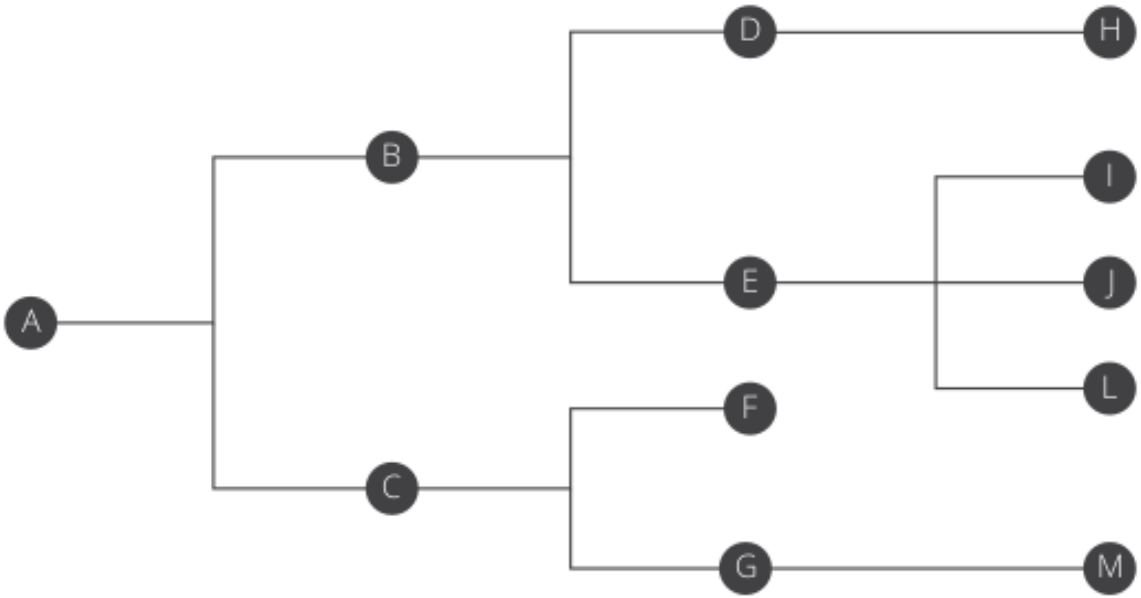
- O **grau de uma árvore** (ou ordem) é o número máximo de filhos que um nodo possui;
- O **grau de um nodo** é o número de filhos que ele possui.



Qual o grau dessa árvore?

Descrição

- O **grau de uma árvore** (ou ordem) é o número máximo de filhos que um nodo possui;
- O **grau de um nodo** é o número de filhos que ele possui.



Qual o grau dessa árvore? **3**

Descrição

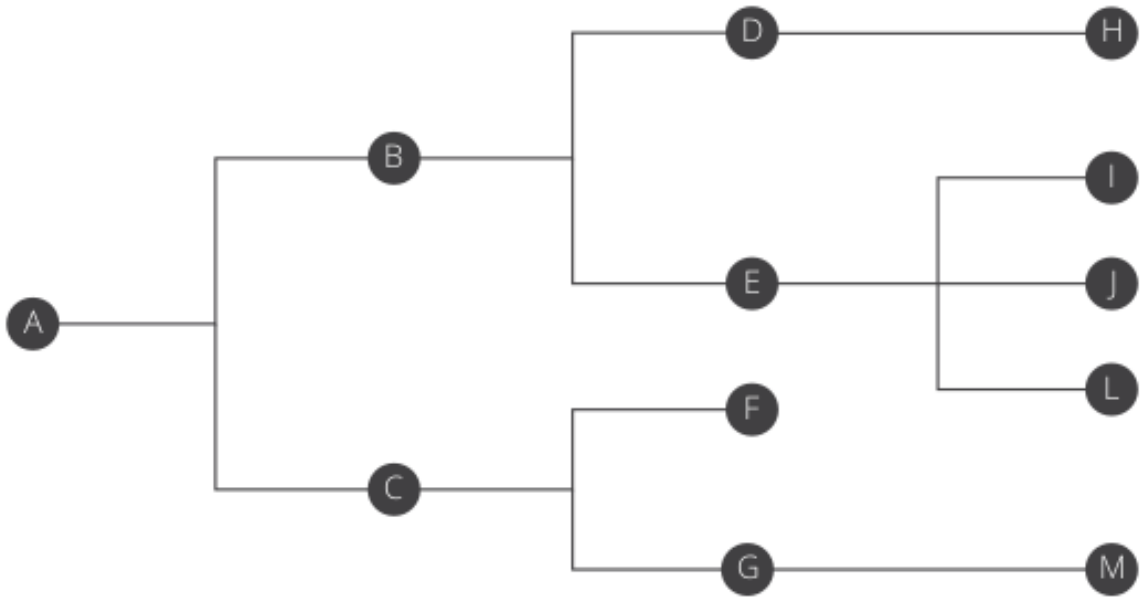
- A **profundidade** (ou nível) de um nodo é a sua distância em relação à raiz;

Descrição

- A **profundidade** (ou nível) de um nodo é a sua distância em relação à raiz;
- A profundidade da raiz é igual a zero.

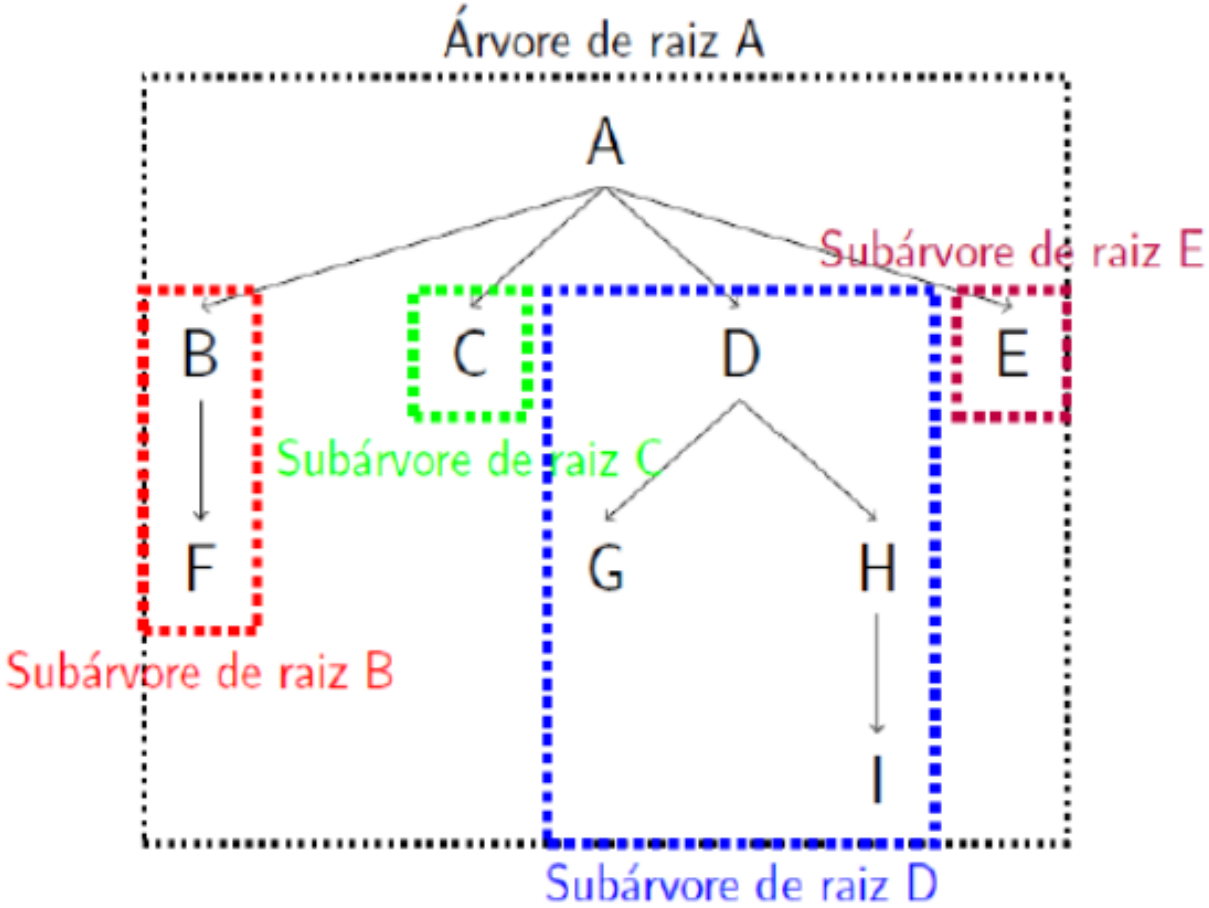
Descrição

- A **profundidade** (ou nível) de um nodo é a sua distância em relação à raiz;
- A profundidade da raiz é igual a zero.
- A profundidade de um nodo é igual a 1 + a profundidade do seu pai.

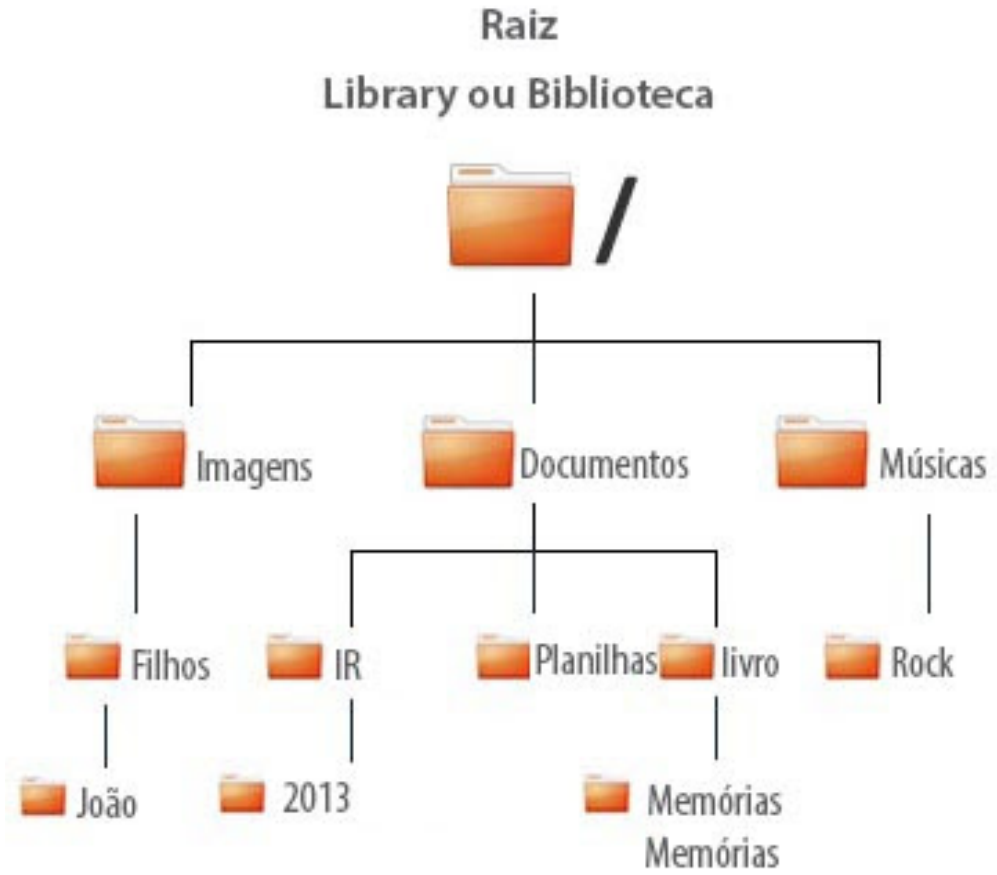


Subárvores

- Todos os nodos que compartilham um ancestral em comum.

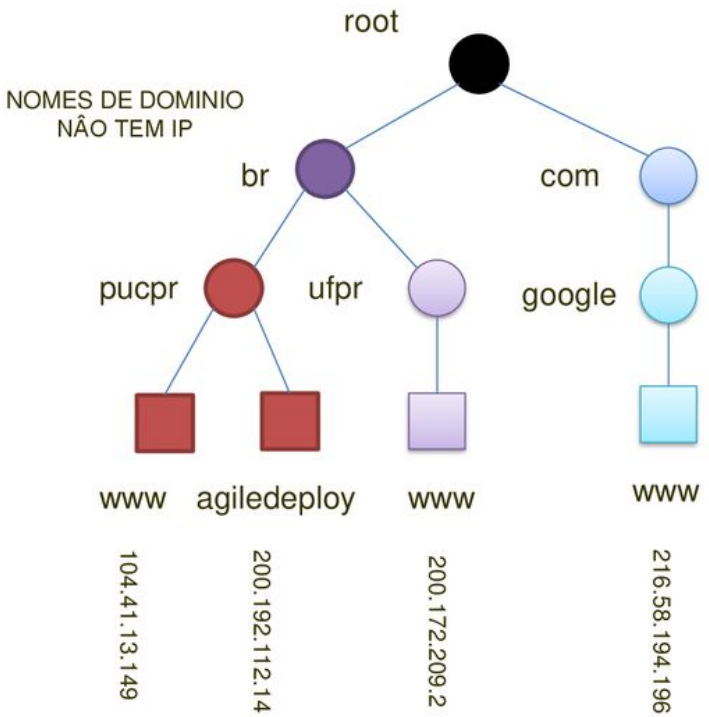


Aplicações: Diretórios



Aplicações: DNS

Árvore de nomes



APENAS AS FOLHAS TEM IP

FQDN: FULLY QUALIFIED DOMAIN NAME

NOME COMPLETO DA FOLHA ATÉ A RAIZ

www.pucpr.br

E muito mais...

- Problemas de busca de dados armazenados na memória principal do computador: árvore binária de busca, árvores (quase) balanceadas como AVL, rubro-negra, etc.
- Problemas de busca de dados armazenados na memória secundárias principal do computador (disco rígico): e.g. B-árvores.
- Aplicações em Inteligência Artificial: árvores que representam o espaço de soluções, e.g. jogo de xadrez, resolução de problemas, etc.
- No processamento de cadeias de caracteres: árvore de sufixos.
- Na gramática formal: árvore de análise sintática.
- Etc

Árvores binárias

Definição

- **Árvore binária** é uma árvore onde cada nodo tem, no máximo, dois filhos;

Árvores binárias

Definição

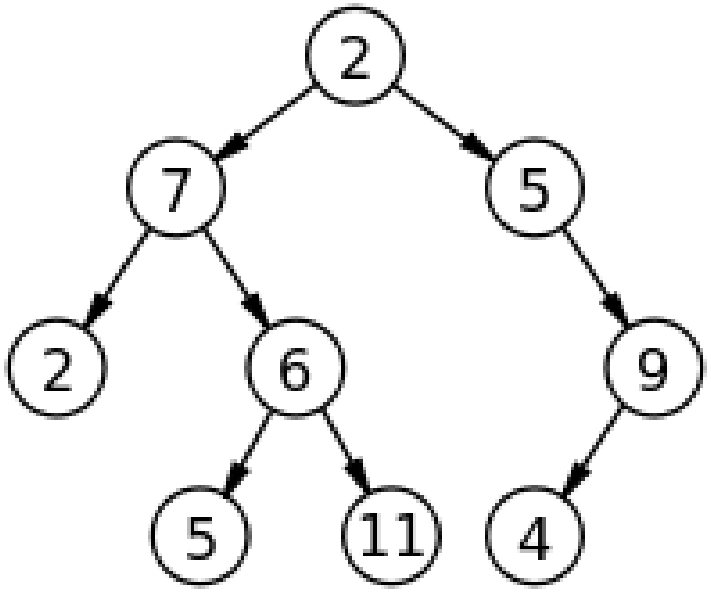
- **Árvore binária** é uma árvore onde cada nodo tem, no máximo, dois filhos;
- Uma árvore binária é **própria** se cada um de seus nodos tiver zero ou dois filhos. Logo, em uma árvore binária própria, todo nodo interno tem exatamente dois filhos;

Árvores binárias

Definição

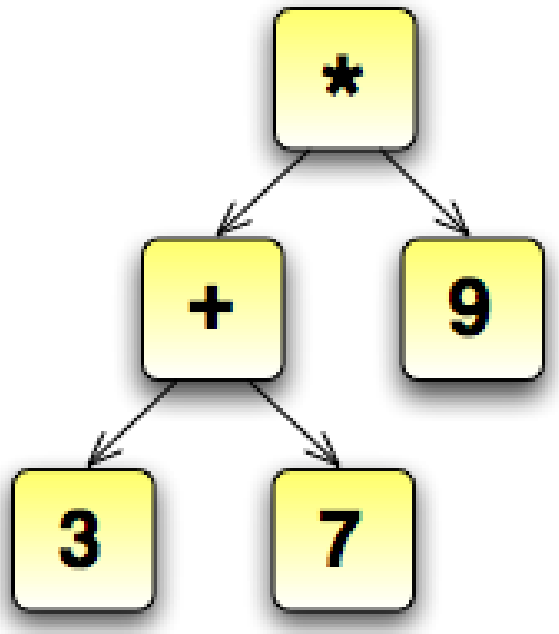
- **Árvore binária** é uma árvore onde cada nodo tem, no máximo, dois filhos;
- Uma árvore binária é **própria** se cada um de seus nodos tiver zero ou dois filhos. Logo, em uma árvore binária própria, todo nodo interno tem exatamente dois filhos;
- Para cada nodo interno de uma árvore binária, nomeamos os filhos como **filho da esquerda** e **filho da direita**.

Árvores binárias - Exemplo



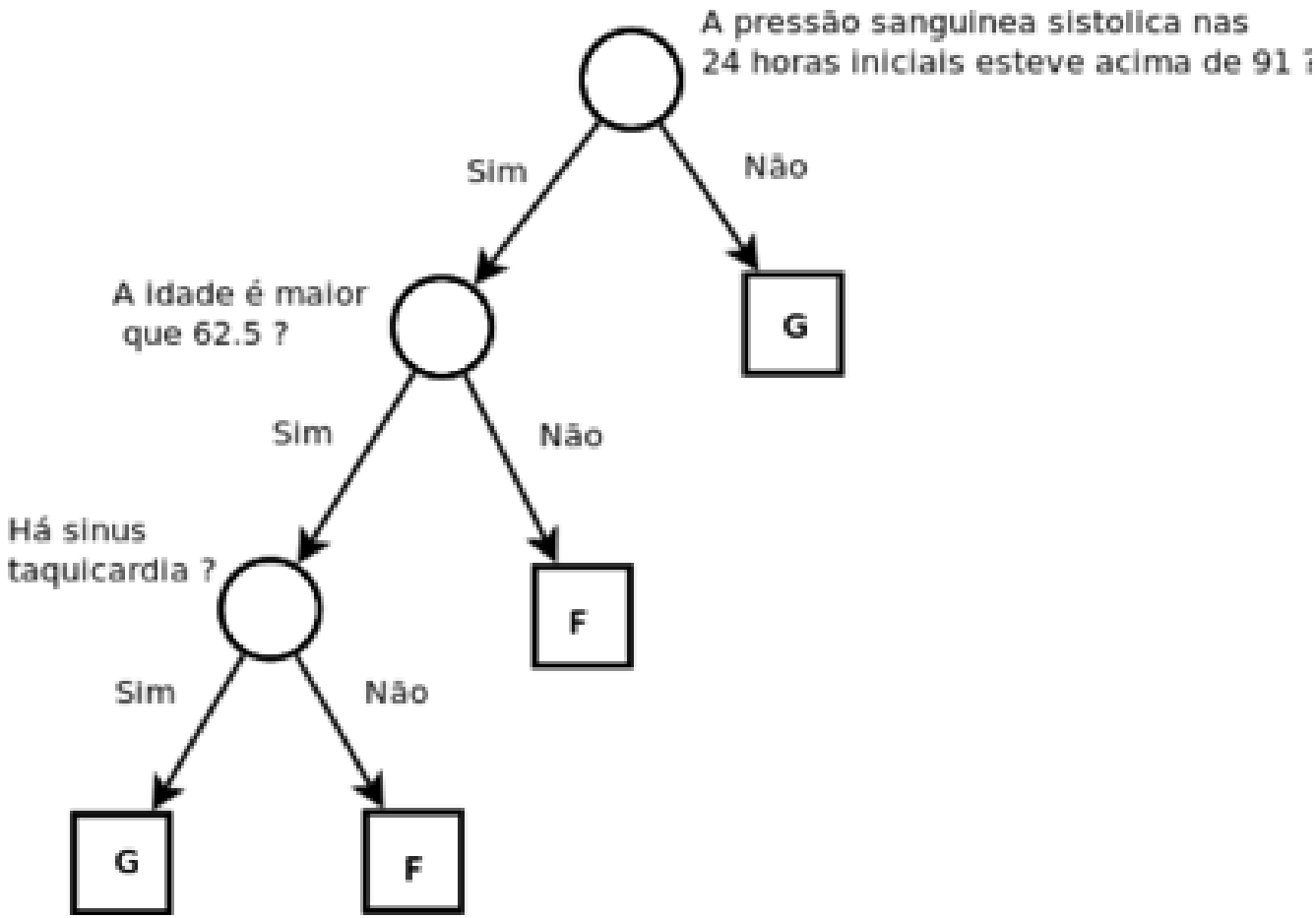
Árvores binárias - Exemplo

Uma expressão aritmética pode ser representada por uma árvore cujos nodos externos são associados com variáveis ou constantes e cujos nodos internos são associados com um operador.



Árvores binárias - Exemplo

Uma árvore de decisão (de grau 2, obviamente)



Árvores binárias de busca (BST)

Definição

- **Árvore binária de busca** é uma árvore binária onde cada nodo p possui um campo chave que admite comparação;

Árvores binárias de busca (BST)

Definição

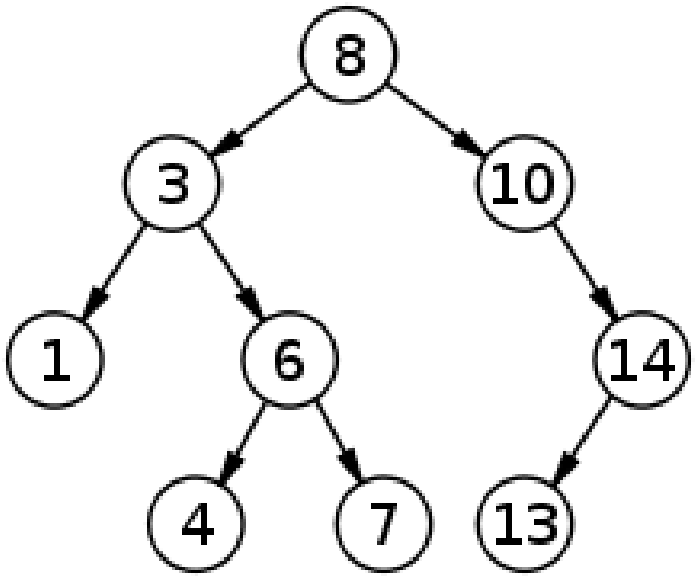
- **Árvore binária de busca** é uma árvore binária onde cada nodo p possui um campo chave que admite comparação;
- A chave de um nodo p é maior ou igual à chave de cada nodo da subárvore esquerda;

Árvores binárias de busca (BST)

Definição

- **Árvore binária de busca** é uma árvore binária onde cada nodo p possui um campo chave que admite comparação;
- A chave de um nodo p é maior ou igual à chave de cada nodo da subárvore esquerda;
- A chave de um nodo p é menor ou igual à chave de cada nodo da subárvore direita;

Árvores binárias de busca - Exemplo



Como implementar

- Uma árvore binária, onde o grau máximo dos nodos é dois, pode ser representada por uma estrutura similar a uma lista duplamente encadeada;
- Cada nodo possui, além das informações pertinentes a ele, um ponteiro para o filho da esquerda e outro para o filho da direita.

```
1  public class Node {
2          int valor;
3          Node esquerda;
4          Node direita;
5          public Node(int valor){
6                  this.valor = valor;
7                  this.esquerda = null;
8                  this.direita = null;
9          }
10 }
```

Adicionando nodos

```
1  public class BSTree {
2  Node raiz = null;
3  public void inserir (Node no, int valor) {
4  if (no != null )
5    if (valor < no.valor)
6          if (no.esquerda != null)
7                  inserir(no.esquerda, valor);
8          else
9                  no.esquerda = new Node(valor);
10   else
11         if (no.direita != null)
12                 inserir(no.direita, valor);
13         else
14                 no.direita = new Node(valor);
15 else
16   raiz = new Node(valor);
17 } }
```

Percorrendo a árvore

Em algumas aplicações, é necessário percorrer uma árvore de forma sistemática, visitando cada nó da árvore uma única vez, em determinada ordem.
Por exemplo, se cada nó da árvore possui um campo que armazena o salário, então podemos querer visitar cada nó para fazer um reajuste salarial. A visita seria atualizar o campo salário. Não podemos esquecer nenhum nó, nem queremos visitar um nó mais do que uma vez. Neste caso, a ordem de visita não é importante. Mas em algumas outras aplicações, queremos visitar os nós em certa ordem desejada. Veremos três formas para percorrer uma árvore binária.

- In-ordem ou ordem simétrica;
- Pré-ordem;
- Pós-ordem.

In-Ordem (ou Simétrica)

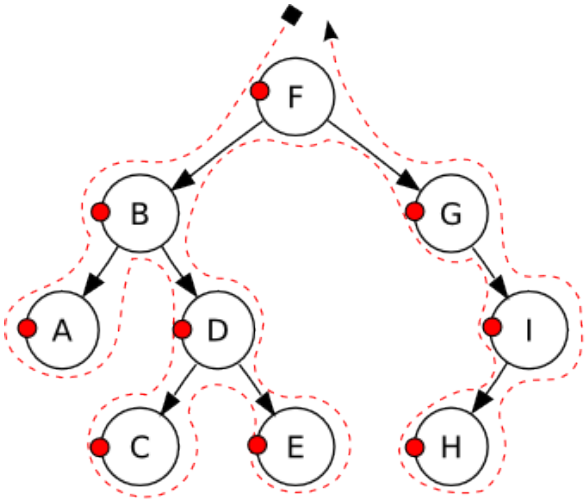
- Percorrer a sua subárvore esquerda em in-ordem;
- Vistar a raiz;
- Percorrer a sua subárvore direita em in-ordem.

Pré-Ordem

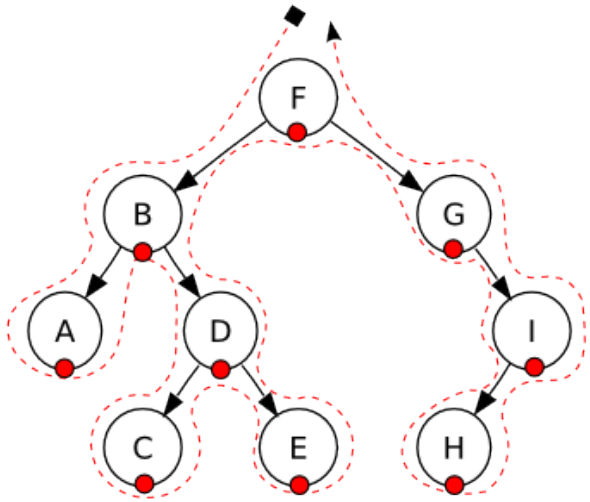
- Vistar a raiz;
- Percorrer a sua subárvore esquerda em pré-ordem;
- Percorrer a sua subárvore direita em pré-ordem.

Pós-Ordem

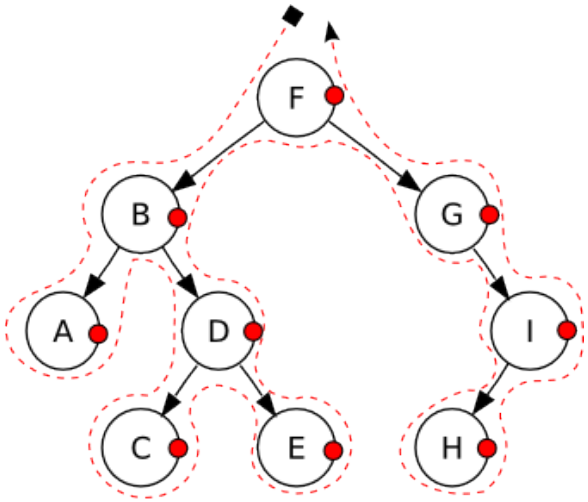
- Percorrer a sua subárvore esquerda em pós-ordem;
- Percorrer a sua subárvore direita em pós-ordem;
- Vistar a raiz.



Pré-ordem: F, B, A, D, C, E, G, I, H



Ordem simétrica: A, B, C, D, E, F, G, H, I



Pós-ordem: A, C, E, D, B, H, I, G, F

In-Ordem

```
1  public void inordem(Node no){
2    if (no!=null){
3      inordem(no.esquerda);
4      System.out.print(no.valor + " ");
5      inordem(no.direita);
6    }
7  }
```

Pré-Ordem

```
1  public void preordem(Node no){
2    if (no!=null){
3      System.out.print(no.valor + " ");
4      preordem(no.esquerda);
5      preordem(no.direita);
6    }
7  }
```

Pós-Ordem

```
1  public void posordem(Node no){
2    if (no!=null){
3          posordem(no.esquerda);
4          posordem(no.direita);
5          System.out.print(no.valor + "  ");
6    }
7  }
```

Encontrando o menor elemento de uma árvore BST

Menor elemento

Dada as características de uma árvore binária de pesquisa, onde se encontra o menor elemento de uma subárvore?

Encontrando o menor elemento de uma árvore BST

Menor elemento

Dada as características de uma árvore binária de pesquisa, onde se encontra o menor elemento de uma subárvore?

O mais a esquerda possível.

```
1  private Node minimo(Node no) throws Exception{
2      if (no==null)
3          throw new Exception ("Raiz nula");
4      if (no.esquerda!=null)
5          return minimo(no.esquerda);
6      else
7          return no;
8  }
```

Removendo o menor elemento de uma árvore BST

Removendo o menor elemento

A remoção do menor elemento de uma subárvore é simples. Temos duas opções:

- O elemento é uma folha: Nesse caso o filho esquerdo de seu pai passa a ser nulo;

Removendo o menor elemento de uma árvore BST

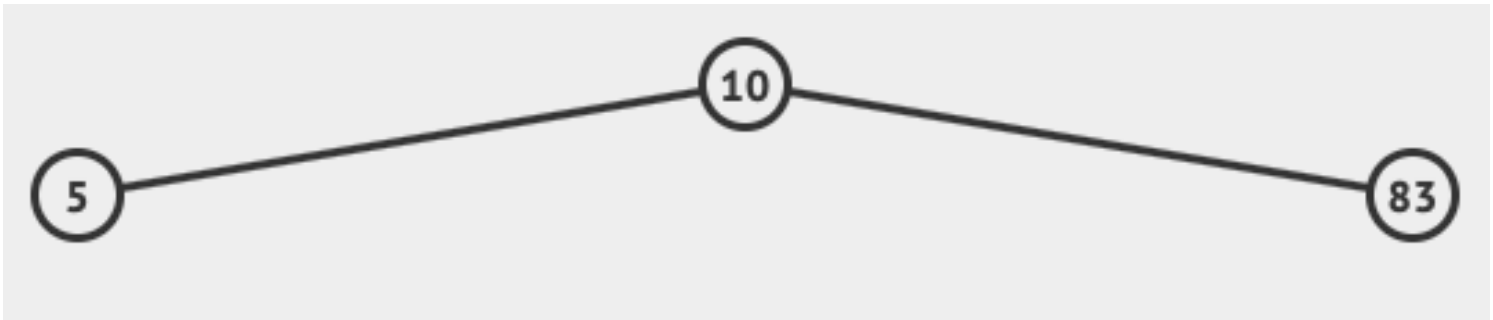
Removendo o menor elemento

A remoção do menor elemento de uma subárvore é simples. Temos duas opções:

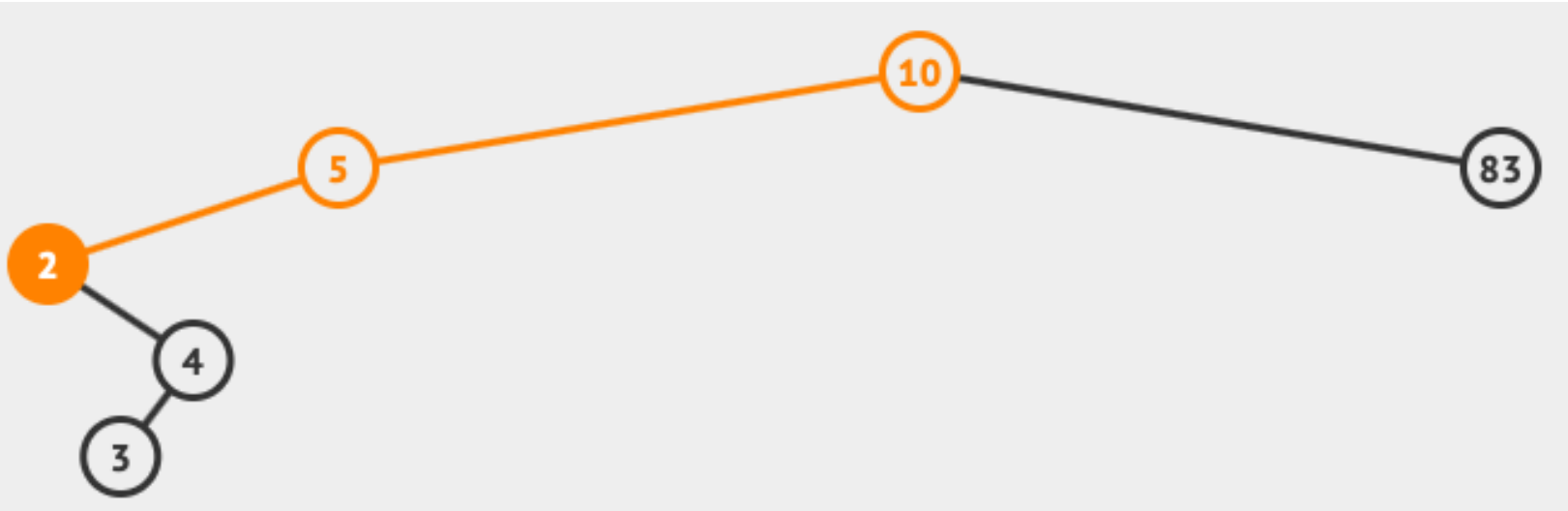
- O elemento é uma folha: Nesse caso o filho esquerdo de seu pai passa a ser nulo;
- O elemento possui um filho à direita: Nesse caso, o filho esquerdo de seu pai passa a ser o *neto* à direita.

Menor é uma folha

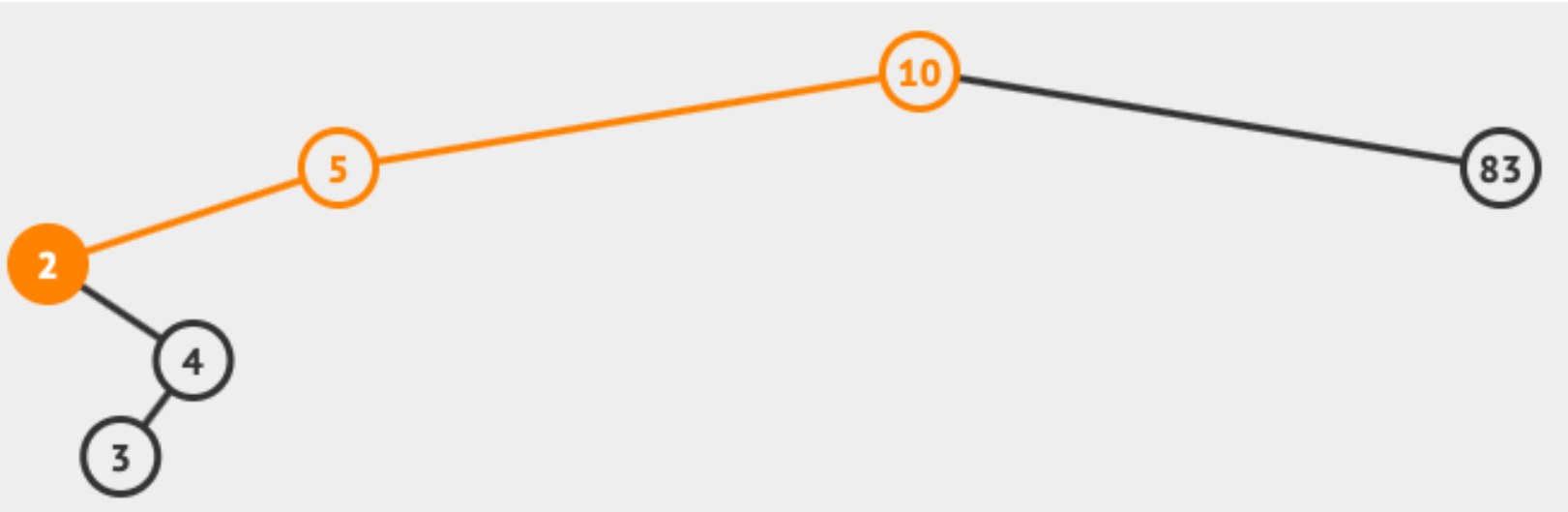
Menor é uma folha

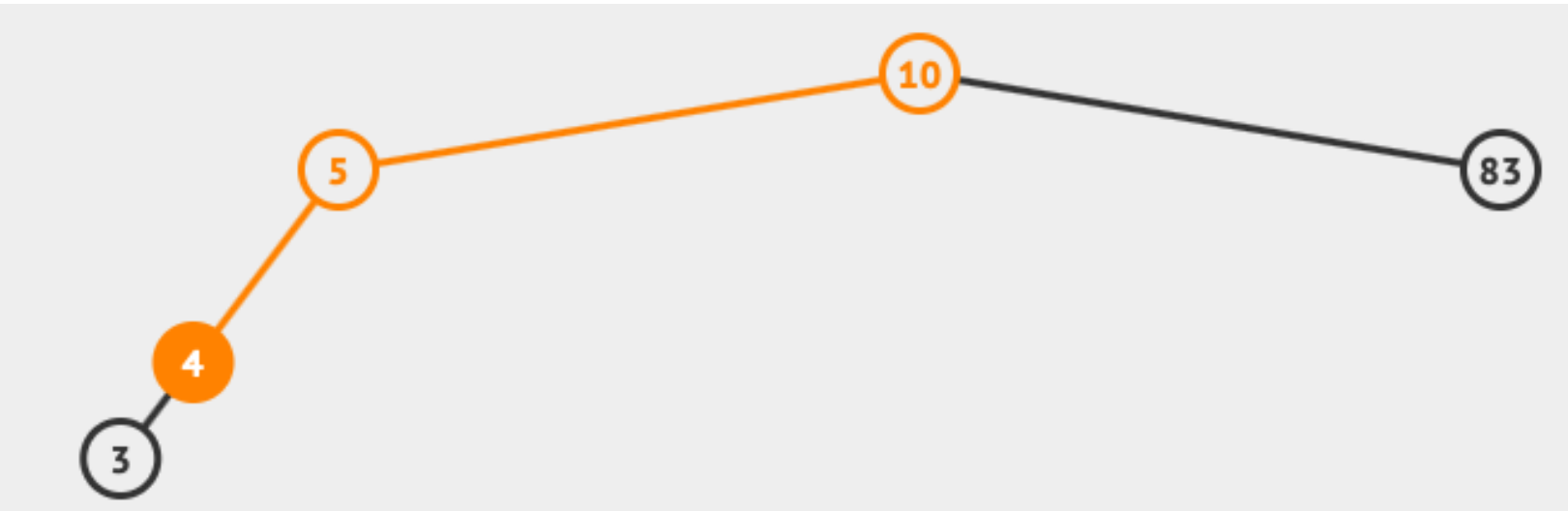


Menor tem filho



Menor tem filho





Removendo o menor elemento

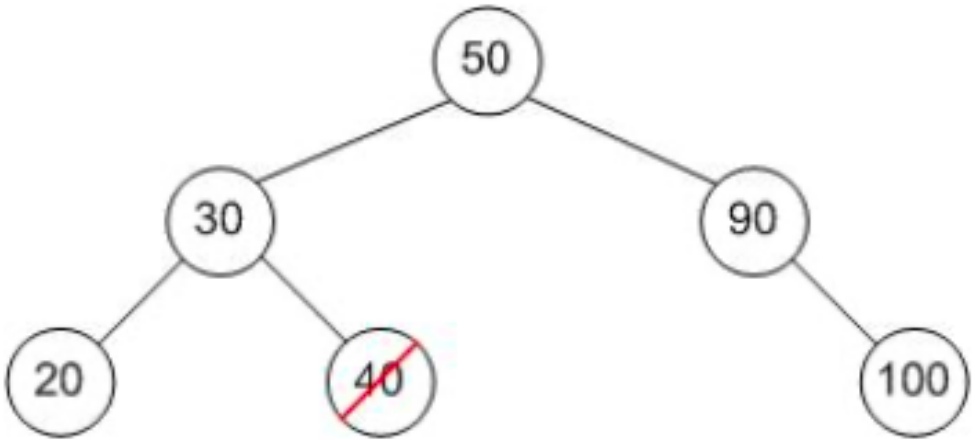
```
1  private Node removeMinimo(Node no)throws
       Exception{
2       if (no == null)
3           throw new Exception("Raiz nula");
4       else
5           if (no.esquerda!=null){
6               no.esquerda = (removeMinimo(no.
                    esquerda));
7               return no;
8           }else{
9               return no.direita;
10          }
11 }
```

Removendo um elemento qualquer da árvore BST

Removendo um elemento qualquer

A remoção de um elemento qualquer possui três casos distintos:

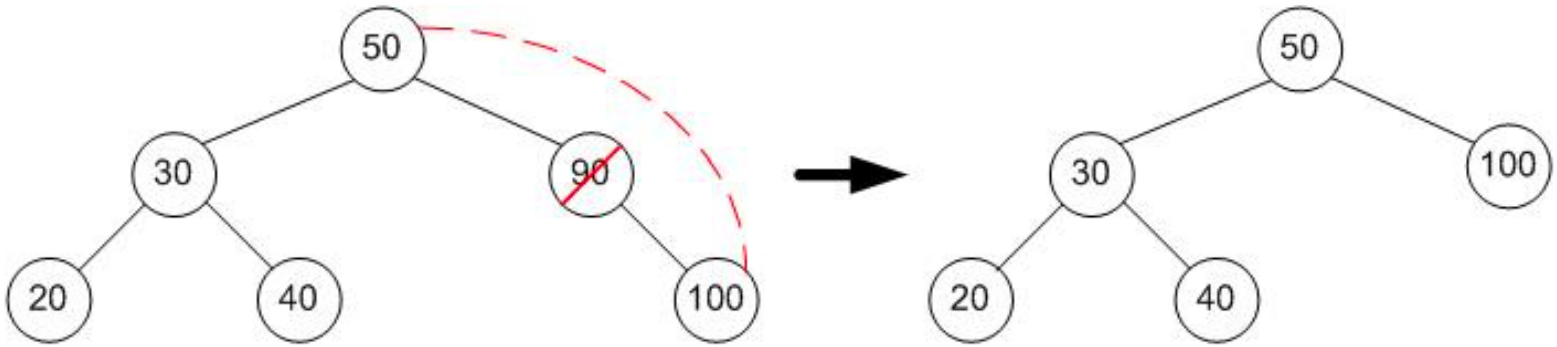
- O elemento é uma folha: Basta fazer o pai do nó apontar seu filho, esquerdo ou direto, dependendo de quem está sendo removido, para nulo.



Removendo um elemento qualquer da árvore BST

Removendo um elemento qualquer

- O elemento possui **um** filho: O filho sobe para a posição do pai.

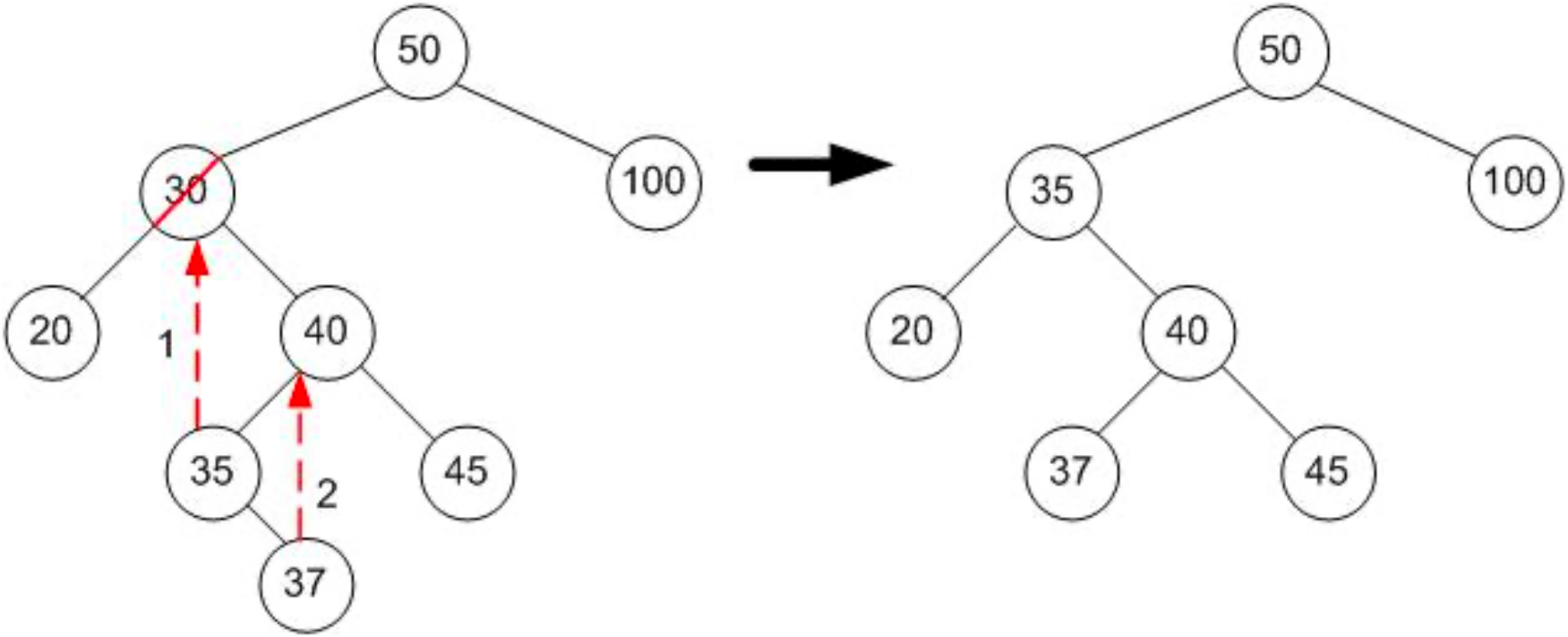


Removendo um elemento qualquer da árvore BST

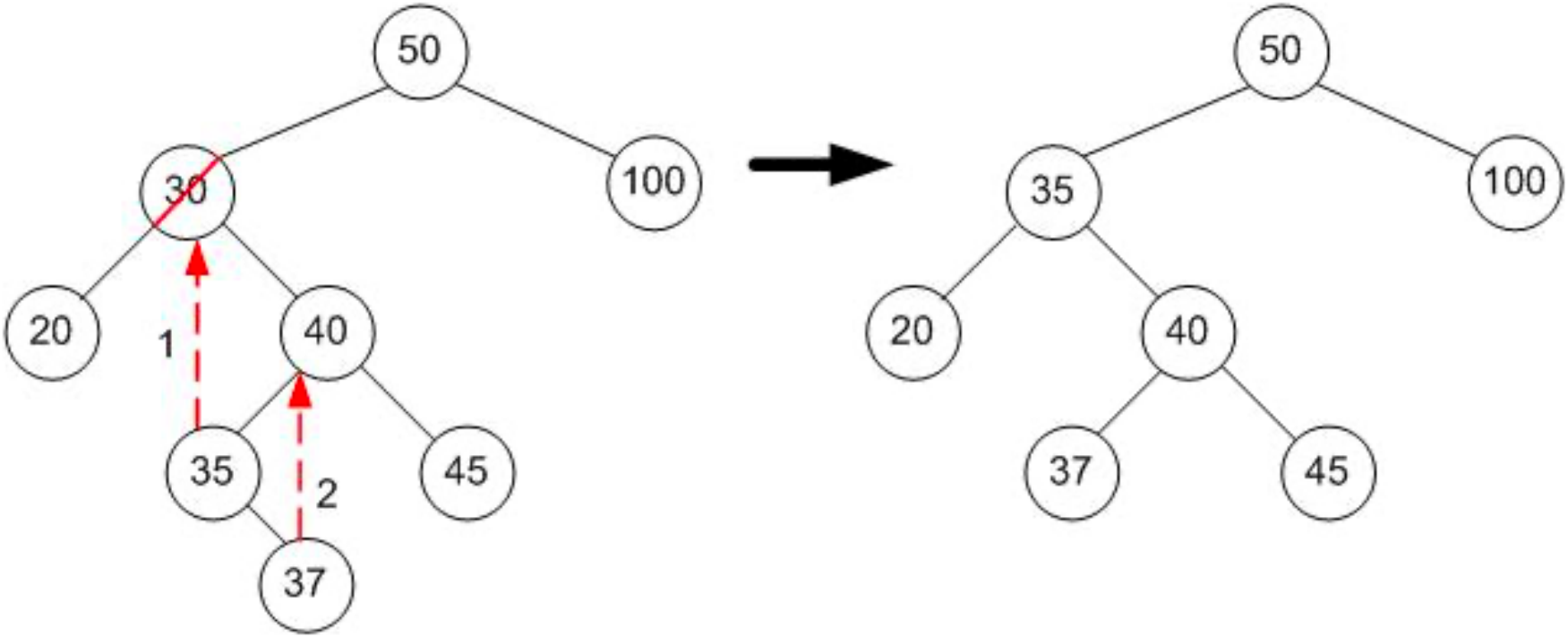
Removendo um elemento qualquer

- O elemento possui **dois** filhos: Neste caso, pode-se operar de duas maneiras diferentes. Pode-se substituir o valor do nó a ser retirado pelo valor sucessor (o nó mais à esquerda da subárvore direita) ou pelo valor antecessor (o nó mais à direita da subárvore esquerda), removendo-se aí o nó sucessor (ou antecessor), o que cairá no caso 1 ou 2.

Removendo um elemento qualquer da árvore BST



Removendo um elemento qualquer da árvore BST



O método de remoção geral é um pouco extenso e por isso será apresentado somente no exemplo completo.

Exemplo de árvore binária de busca

Baixe o exemplo aqui.

Acrescente ao exemplo

- Implemente um método que retorne a quantidade de elementos que uma árvore possui;
- Implemente um método que retorne a quantidade de folhas que uma árvore possui;
- Implemente um método que retorne a soma dos valores (atributo valor) de todos os nós da árvore.

Exercícios teóricos

1. Considere que os seguintes números foram inseridos, na ordem apresentada, em uma árvore binária de busca (BST): *15, 19, 36, 5, 6, 7, 10, 20, 3, 1* - Desenhe a árvore resultante.
2. Qual a altura da árvore gerada?
3. Gere uma nova BST inserindo os números na ordem inversa a apresentada. Considerando a árvore resultante, comente sobre o desempenho de um algoritmo para localização de um elemento nas duas árvores. Qual apresenta o melhor desempenho? Por que? O que você sugere para melhorar o desempenho?
4. Remova o elemento 36 da árvore original. Desenhe a árvore resultante.
5. Apresente os elementos da primeira árvore em **Pré-ordem**, **Ordem** e **Pós-ordem**.

Exercícios práticos (GIT)

1. Implemente uma árvore binária de busca. Cada *nodo* será um aluno contendo número de matrícula, nome e curso (além dos ponteiros necessários). O número de matrícula será usado como critério de ordenação.
2. Implemente um método que retorne o aluno com a maior matrícula da árvore.
3. Implemente um método que permita dizer se um aluno (matrícula) está ou não cadastrado.
4. Implemente um método que imprima os alunos em pré-ordem.
5. Implemente um método que exclua um aluno.
6. No **main**, leia 10 alunos e insira na árvore. Teste todos os métodos implementados.

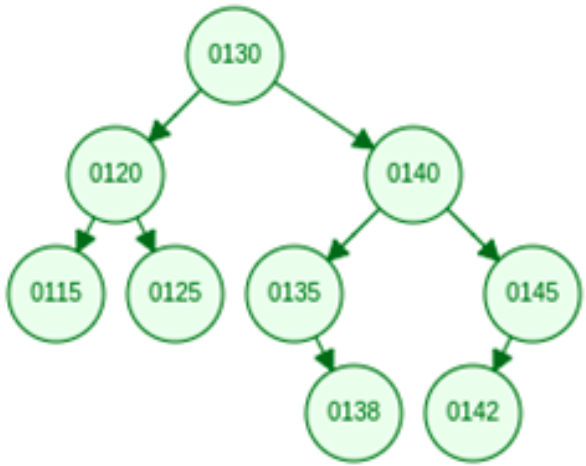
Exercícios práticos (GIT)

Na escola **Brincadeira e Sabedoria Total**, a turma do 5º ano está sempre curiosa para a chegada do recreio, pois a professora incentiva seus alunos a aprenderem de maneiras divertidas e criativas. Numa tarde de sol, ela pensou em dividir os alunos em duas filas, com os mais altos na direita e os mais baixos na esquerda. Mas depois a professora percebeu que poderia ficar mais divertido se cada fila fosse sempre subdividida em duas com esta mesma lógica. Então ela definiu as seguintes regras para a brincadeira:

1. **Início da Fila**: A primeira criança a entrar na brincadeira fica na frente da fila.
2. **Comparação de Altura**: Cada nova criança que entrar na brincadeira compara sua altura com a da primeira criança:
   - Se for mais baixa, vai para a fila da esquerda.
   - Se for mais alta, vai para a fila da direita.
3. **Subdivisões das Filas**: Cada vez que uma criança chega na posição já ocupada por outra criança, ela compara sua altura com a altura daquele/a colega até encontrar uma posição livre:
   - Se for mais baixa, continua indo para a esquerda.
   - Se for mais alta, continua indo para a direita.
   - Ao encontrar uma posição livre, fica lá.

Exercícios práticos (GIT)

Por exemplo, dada a seguinte sequência de crianças entrando na brincadeira, com a respectiva altura anotada em centímetros [130, 120, 140, 115, 125, 135, 145, 142, 138] a turma ficaria organizada com a seguinte disposição:



Como a brincadeira se mostrou divertida, a professora solicitou que você escrevesse um programa que, dada a ordem de entrada dos alunos na brincadeira, exibe qual é a altura da menor criança que está em cada um dos níveis desta estrutura. Entende-se por nível a distância entre a que está no início da fila (a primeira a entrar) e cada uma das demais, considerando-se um espaçamento uniforme.

Exercícios práticos (GIT)

Entrada

A entrada contém um único caso de teste. A primeira linha contém um inteiro **N** ($1 \leq \mathbf{N} < 250$) indicando a quantidade de alunos na turma. A segunda linha vai conter **N** inteiros $H_1, H_2, ..., H_n$ representando a altura em centímetros de cada um dos alunos ($50 \leq \mathbf{H_i} \leq 300$). Considerar que não existem dois estudantes com exatamente a mesma altura em uma turma.

Saída

A saída é formada por uma sequência de linhas contendo dois inteiros separados por espaço: o número do nível (iniciando em 0) e a altura da menor criança que está posicionada naquele nível.

Exemplos de Entrada	Exemplos de Saída
9 130 120 140 115 125 135 145 142 138	0 130 1 120 2 115 3 138
3 100 120 115	0 100 1 120 2 115

Exercícios práticos (GIT Ao Vivo)

Exercícios práticos (GIT Ao Vivo)

Exercícios práticos (GIT Ao Vivo)


thank you
danke
謝謝
рахмат
ngiyabonga
teşekkür ederim
Баярлалаа
спасибо
gracias
dank je
merci
obrigado
dziękuję
bedankt
hvala
mauruuru
sukriya
kop khun krap
go raibh maith agat
mochchakkeram
terima kasih
grazie
arigatō
dakujem
tapadh leat
chnorakaloutioun
gratias ago
gracies
sagolun
taiku
takk
mersi
barka
wetalin
tack
vinaka
blagodaram
misaotra
matondo
paldies
grazzi
mahalo
kiitos
dankie
dhanyavad
nanni
nandri
koszonom
enkosi
bayarlalaa
gracie
akun
dankon
ačiū
djiere dieuf
tau
дякую
mamnun
obrigada
tenki
chokrane
murakoze
хвала
asante
manana
sobodi
dekuji
mèsi
didi madloba
najis tuke
kam sah hamnida
rahmat
tanemirt
rahmet
ありがとう
ευχαριστώ
diolch
dhanyavadagalu
shukriya
merce
мерси
trugarez
xiexie
감사합니다
তোমাকে ধন্যবাদ