CMP1054 – EDI
Java

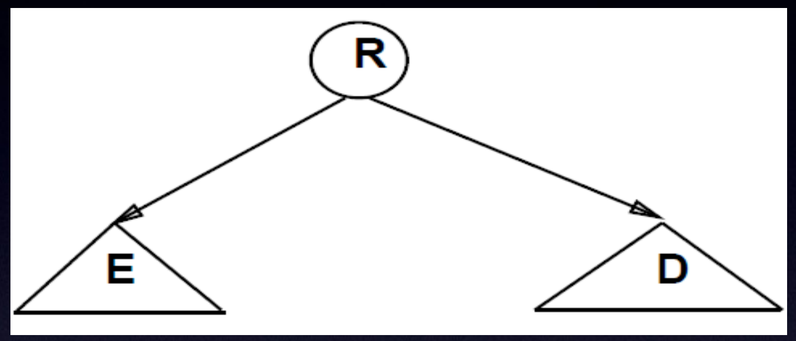
Árvores Binárias de Pesquisa.

Máximo, mínimo, antecessor, sucessor e remover.

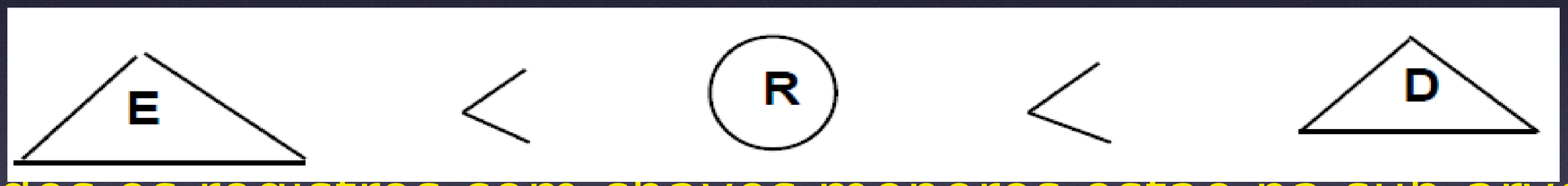
Prof. Dr. José Olimpio Ferreira

Árvores Binárias de Pesquisa

- Para qualquer nó que contenha um registro

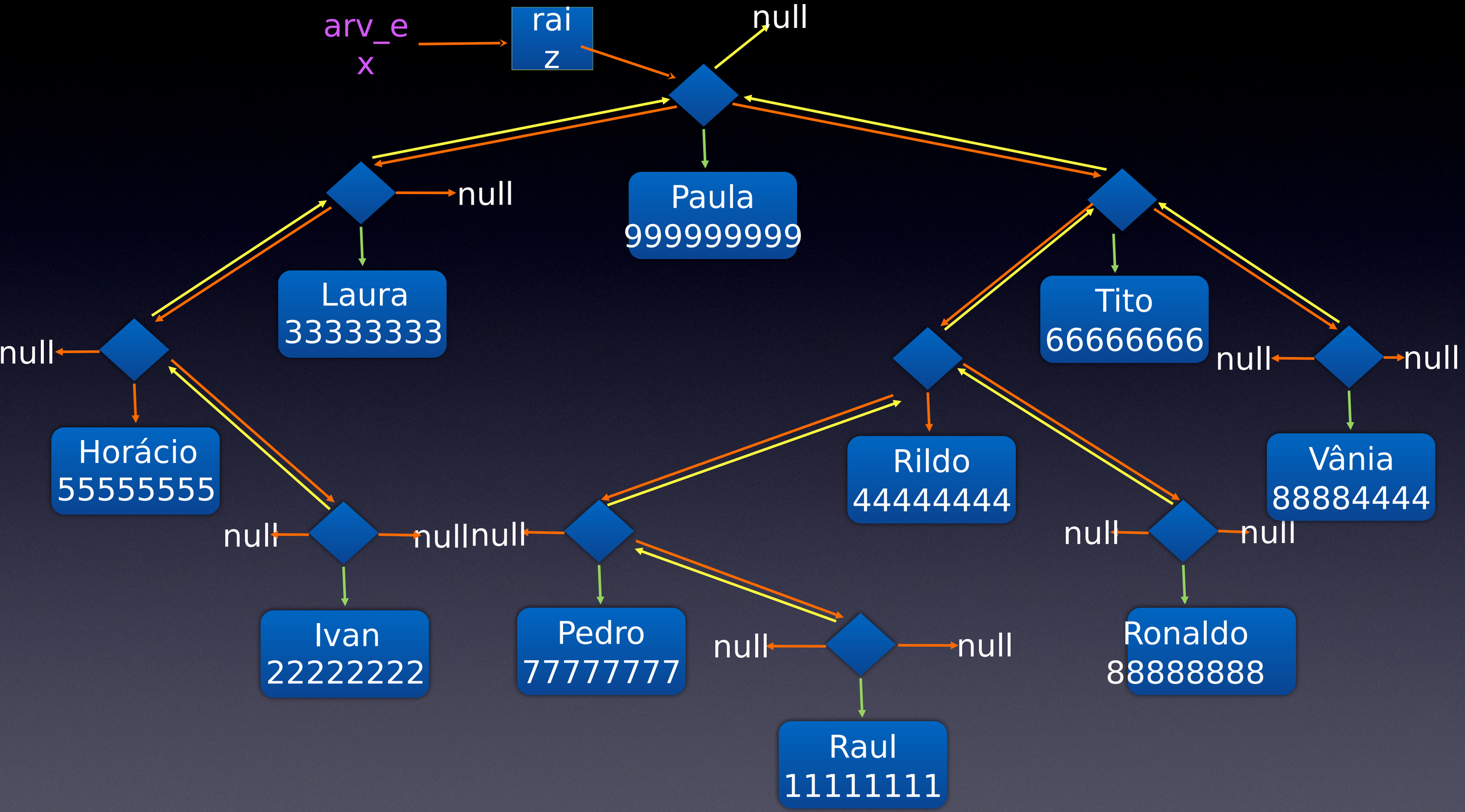


- Temos a relação invariante

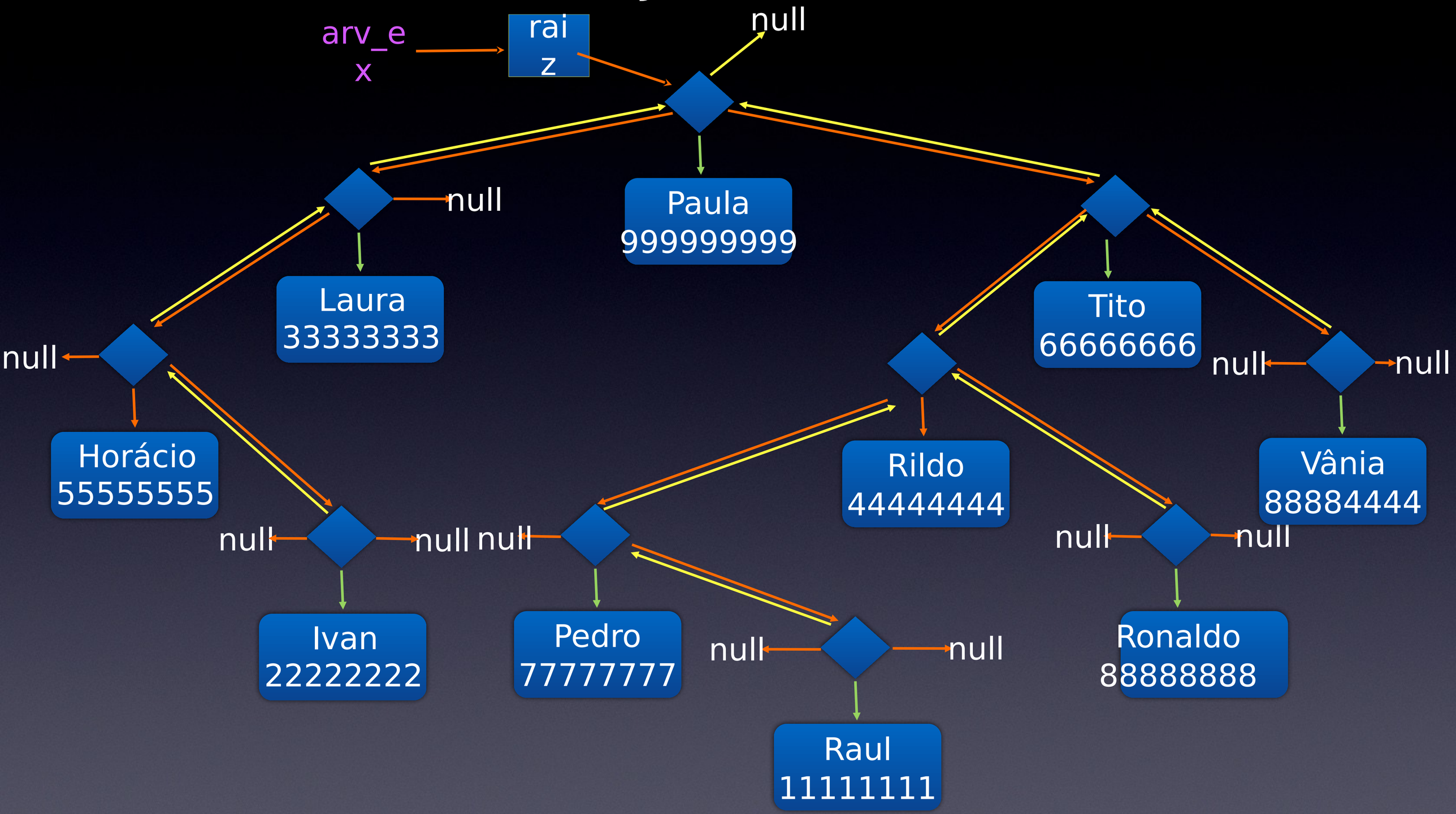


- Todos os registros com chaves menores estão na sub-arvore à esquerda.
- Todos os registros com chaves maiores estão na sub-árvore à direita.

Árvore Binária de Pesquisa

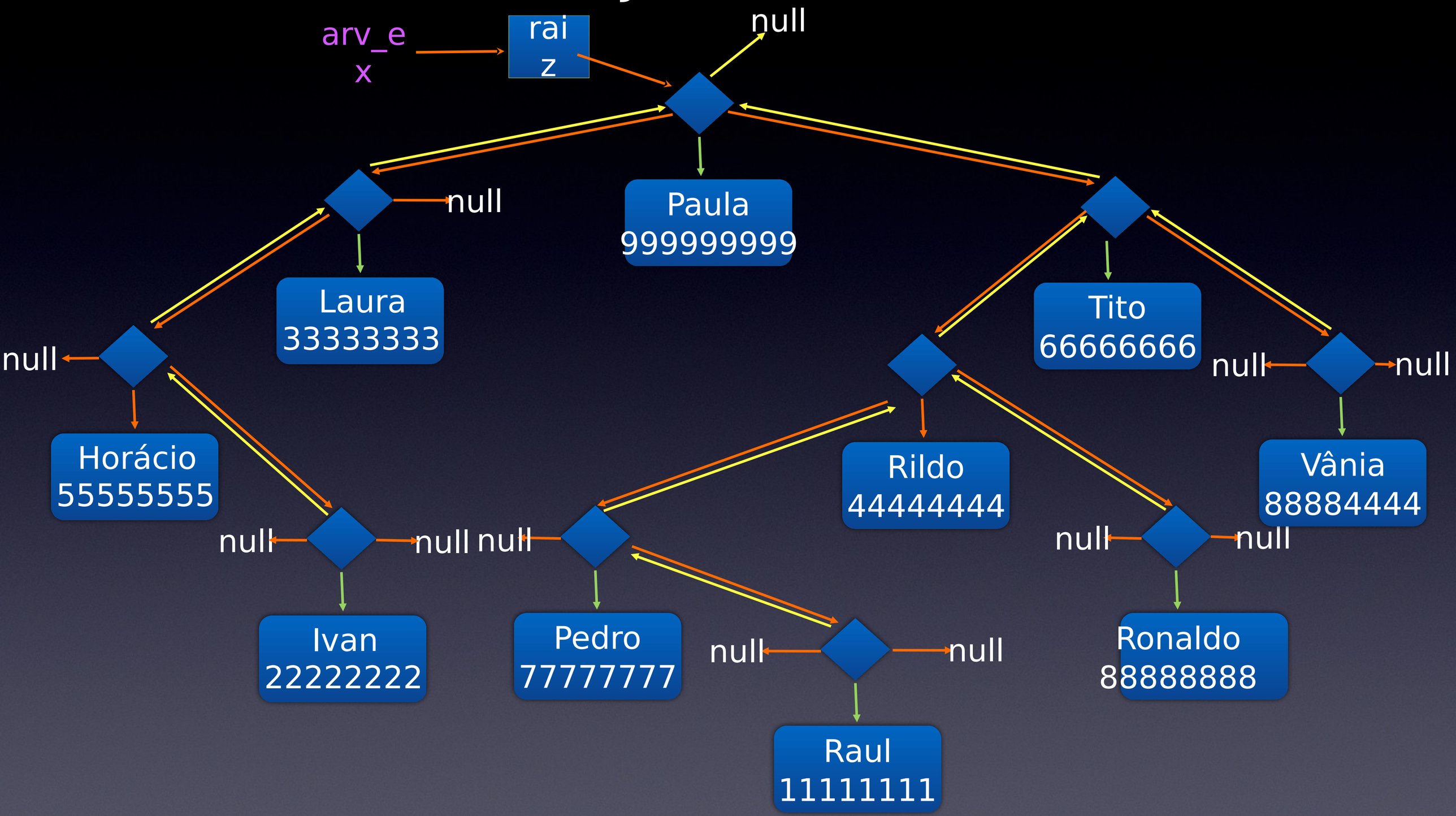


No maximo(No obj)



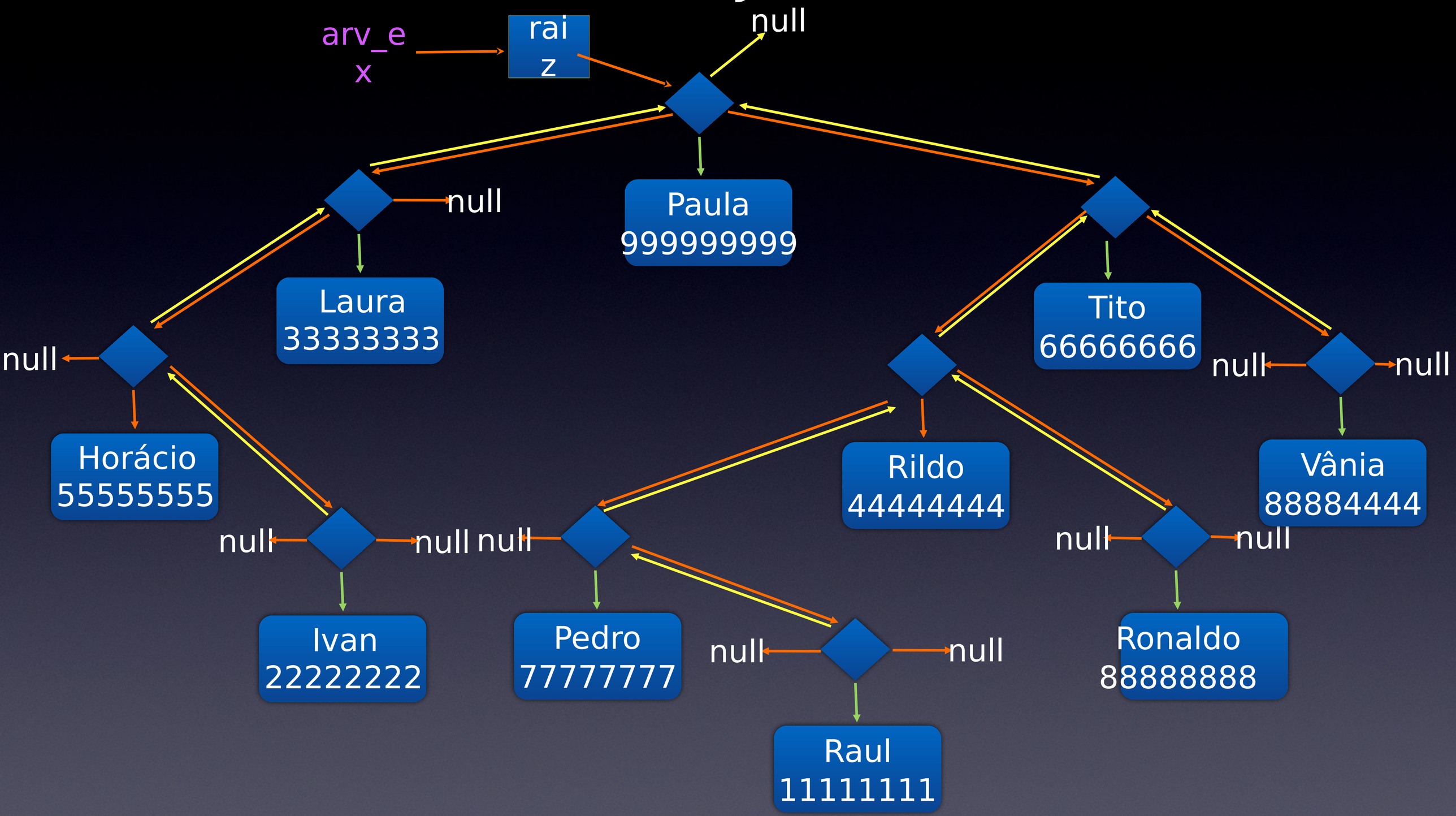
```
private No maximo(No obj) {
  if(obj == null) return null;
  // é necessário usar outra referencia
  // para não alterar a referencia passada no
paramentro
  No atual = obj;
  // laço para encontrar o máximo
  while(atual.fd != null) {
      atual = atual.fd;
  }
  return atual; // maior valor na árvore
}
```

No minimo(No obj)



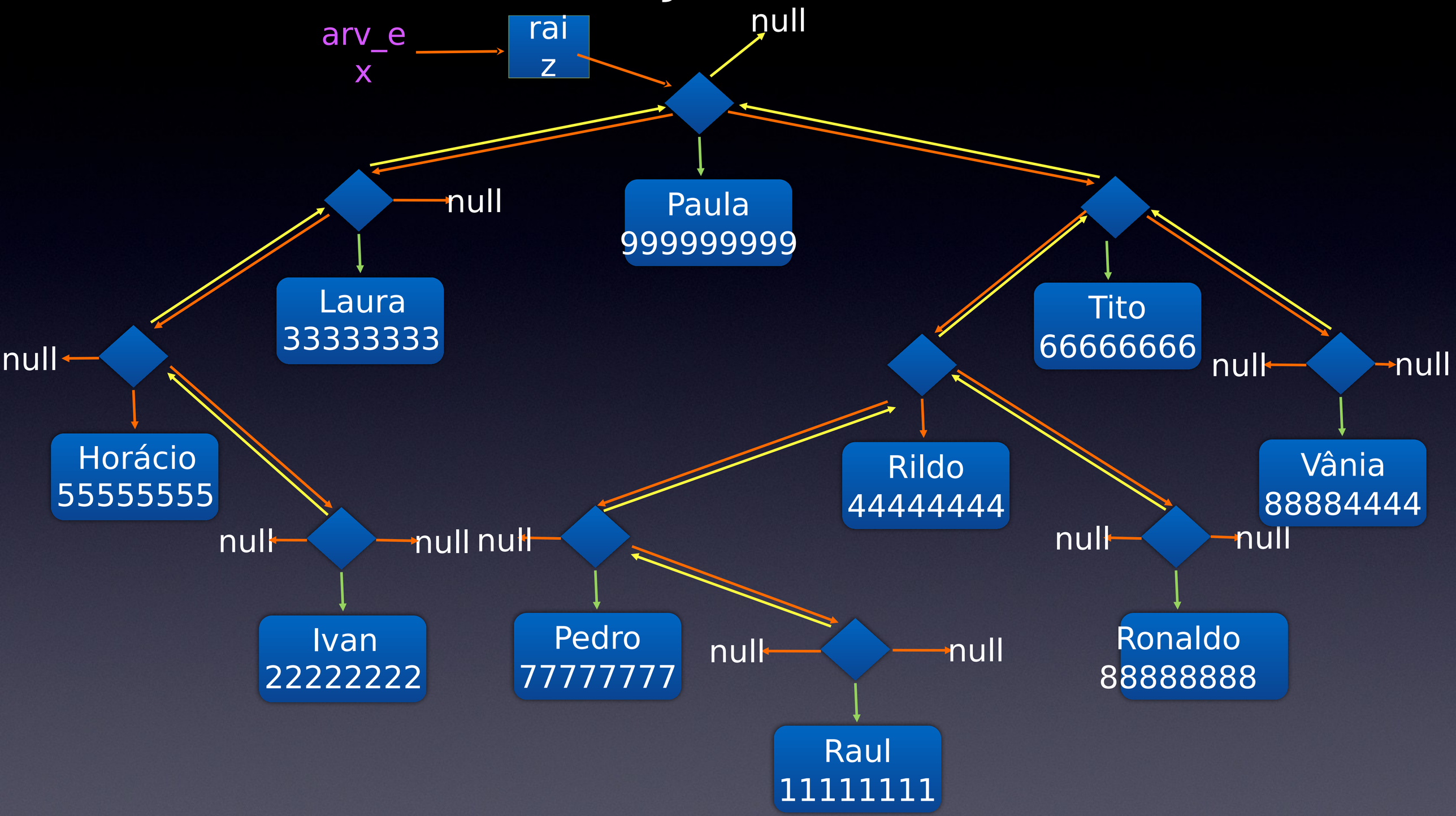
```
private No minimo(No obj) {
  if(obj == null) return null;
  // é necessário usar outra referencia
  // para não alterar a referencia passada no
paramentro
  No atual = obj;
  // laço para encontrar o mminimo
  while(atual.fe != null) {
      atual = atual.fe;
  }
  return atual; // menor valor na árvore
}
```

No antecessor(No obj)



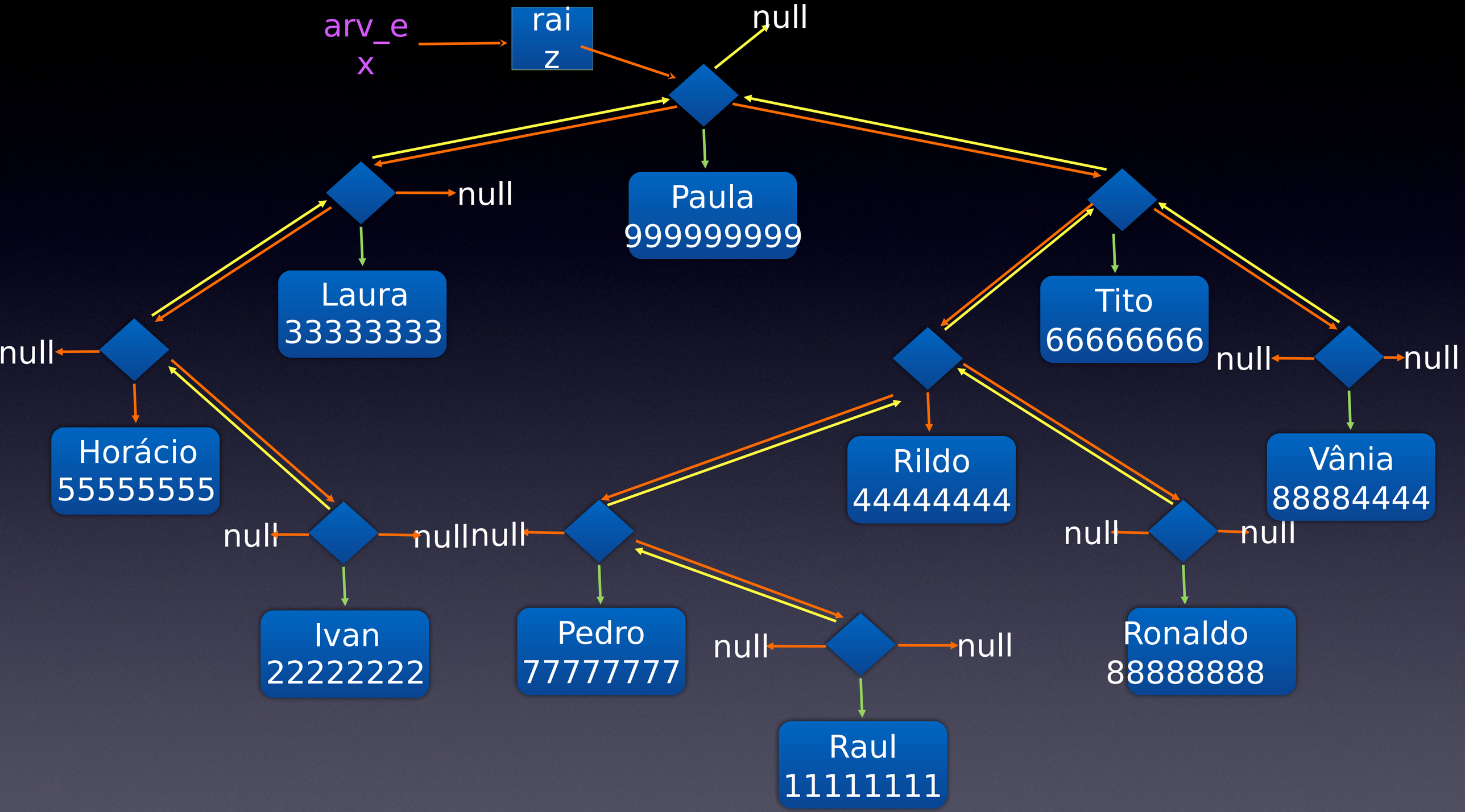
```java
private No antecessor(No obj) {
  if(obj == null) return null;
  //Se tem filho a esquerda o antecessor é o
  //máximo da sub-árvore da esquerda
  if(obj.fe != null) return (maximo(obj.fe));
  //Caso contrário o antecessor pode estar nos ancestrais
  //O antecessor é o primeiro ancestral do qual o
  //nó é filho a direita
  //Pode não ter antecessor
  No atual = obj.pai;
  No ant = obj;
  while(atual != null && ant == atual.fe) {
    ant = atual;
    atual = atual.pai;
  }
  //Se atual é nulo então não existe antecessor
  //Existe antecessor caso atual seja diferente de nulo
  return atual;
}
```

No sucessor(No obj)



```java
private No sucessor(No obj) {
  if(obj == null) return null;
  //Se tem filho a direito o sucessor é o mínimo
  //da sub-árvore da direita
  if(obj.fd != null) return (minimo(obj.fd));
  //Caso contrário o sucessor pode estar nos ancestrais
  //O sucessor é o primeiro ancestral do qual o
  //nó é filho a esquerda
  //Pode não ter sucessor
  No atual = obj.pai;
  No ant = obj;
  while(atual != null && ant == atual.fd) {
    ant = atual;
    atual = atual.pai;
  }
  //Se atual é nulo então não existe sucessor
  //Existe Sucessor caso atual seja diferente de nulo
  return atual;
}
```

Árvore Binária de Pesquisa



Item retirar(Item obj)

Nós com:

- **0 filho(s)**
- **1 filho(s)**
- **2 filhos**

arv_e
x

rai
z

null

null

Paula
999999999

Laura
33333333

null

Tito
66666666

null

null

Horácio
55555555

null

null

Ivan
22222222

null

Pedro
77777777

null

null

Raul
11111111

Rildo
44444444

null

null

Vânia
88884444

Ronaldo
88888888

Item retirar(Item obj)

- Faz uma consulta para ver se “Item obj” existe na árvore.
  - Retorna um “No z” que contém null ou o endereço do No.
- Se a tiver sucesso.
  - Salva-se os dados numa variável “Item aux”.
  - Desliga-se “No z” da árvore.
  - Retorna “No aux”.
- Se não tiver sucesso.
  - Retorna “null”.

Item retirar(Item obj)

- Nós com:
  - 0 filho(s)
  - 1 filho
  - 2 filhos

Item retirar(Item obj)

Nós com:

- **0 filho(s)**
- **1 filho(s)**
- **2 filhos**

arv_e
x

rai
z

null

null

Paula
999999999

Laura
33333333

null

Tito
66666666

null

null

Horácio
55555555

null

null

Ivan
22222222

null

Pedro
77777777

null

null

Rildo
44444444

null

null

Vânia
88884444

Ronaldo
88888888

Raul
11111111

Item retirar(Item obj)

- Retirar Nós com nenhum filho.
  - No z = consultar(obj);
  - Sucesso:
    - Salva-se os dados numa variável “Item aux”.
    - Desliga-se “No z” da árvore.
      - Se z for filho a esquerda faz-se:
        - z.pai.fe = null
      - Se z for filho a direita faz-se:
        - z.pai.fd = null
    - Retorna “Item aux”.
  - Insucesso: retorna null

Item retirar(Item obj)

Nós com:

- **0 filho(s)**

Remover Ronaldo

arv_e
x

rai
z

null

null

Paula
999999999

Laura
33333333

null

Horácio
55555555

null

null

Ivan
22222222

null

null

Pedro
77777777

null

null

Raul
11111111

Tito
66666666

null

null

Rildo
44444444

Vânia
88884444

null

null

Ronaldo
88888888

Item retirar(Item obj)

Nós com:

- **0 filho(s)**

Remover Ronaldo

arv_e
x

rai
z

null

null

Paula
999999999

Laura
33333333

null

Tito
66666666

null

null

Horácio
55555555

null

null

null

null

Rildo
44444444

z

Vânia
88884444

null

null

Ivan
22222222

Pedro
77777777

null

null

Ronaldo
88888888

aux

Raul
11111111

Item retirar(Item obj)

Nós com:

- **0 filho(s)**

Remover Ronaldo

arv_e
x

rai
z

null

null

Paula
999999999

Laura
33333333

null

Tito
66666666

null

null

Horácio
55555555

null

null

null

null

Rildo
44444444

null

Vânia
88884444

Ivan
22222222

Pedro
77777777

null

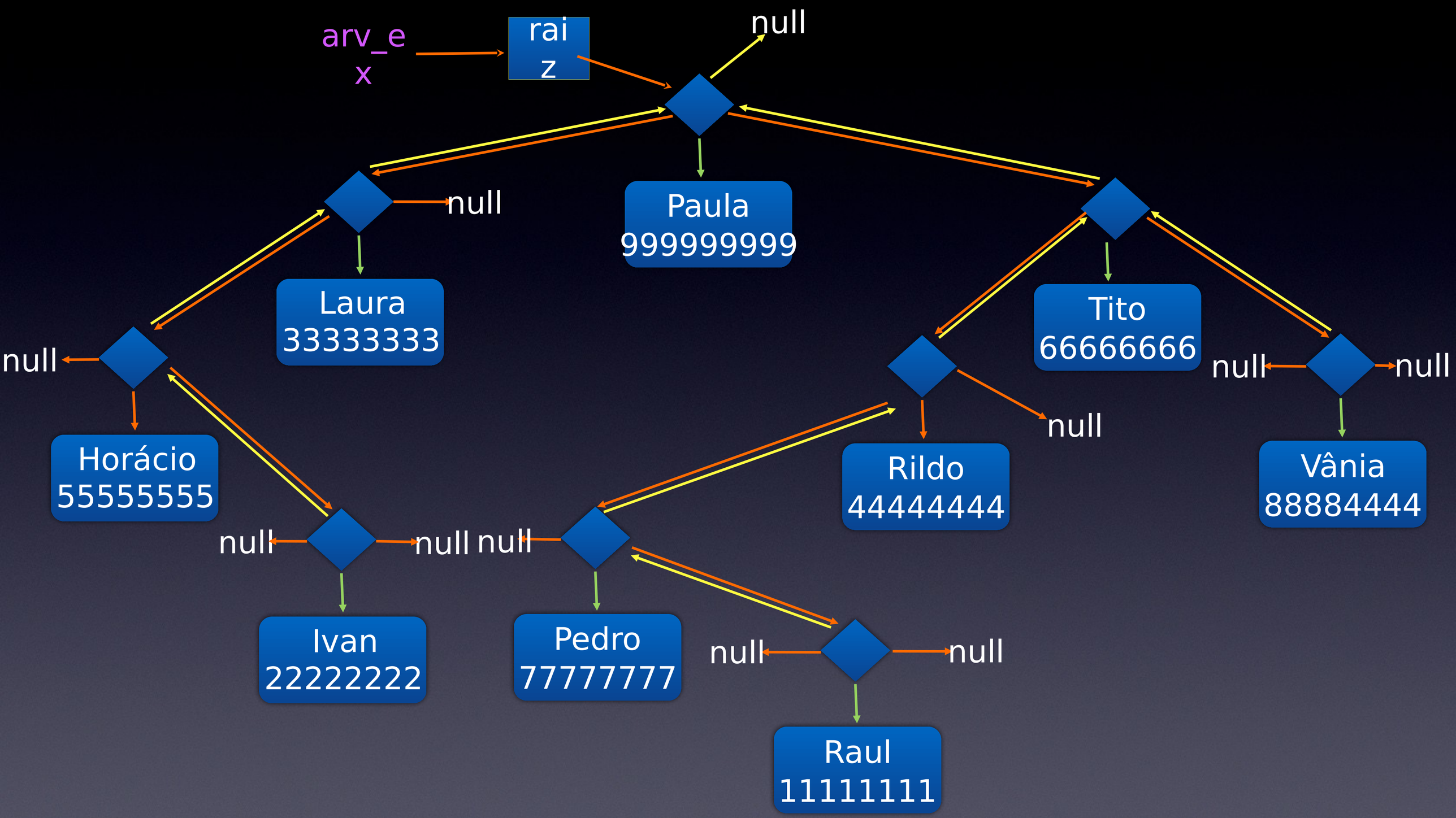
null

Raul
11111111

Ronaldo
88888888

aux

Árvore após 1 remoção



Item retirar(Item obj)

- Retirar Nós com 1 filho.
  - No z = consultar(obj);
  - Sucesso:
    - Salva-se os dados numa variável “Item aux”.
  - Desliga-se “No z” da árvore.
    - Se z for filho a esquerda faz-se:
      - Se z tiver um filho a esquerda:
        - z.pai.fe = z.fe
        - z.fe.pai = z.pai
      - Se z tiver um filho a direita:
        - z.pai.fe = z.fd
        - z.fd.pai = z.pai
    - Se z for filho a direita faz-se:
      - Se z tiver um filho a esquerda:
        - z.pai.fd = z.fe
        - z.fe.pai = z.pai
      - Se z tiver um filho a direira:
        - z.pai.fd = z.fd
        - z.fd.pai = z.pai
    - Retorna “Item aux”.
  - Insucesso:
    - Retorna null

Item retirar(Item obj)

Nós com:

- **1 filho**

Remover Horácio

arv_e
x

rai
z

null

null

Paula
999999999

z

Laura
33333333

null

Tito
66666666

null

null

Horácio
55555555

aux

null

null

null

null

Rildo
44444444

null

Vânia
88884444

Ivan
22222222

Pedro
77777777

null

null

Raul
11111111

Item retirar(Item obj)

Nós com:

- **1 filho**

Remover Horácio

arv_e
x

rai
z

null

null

Paula
999999999

Laura
33333333

null

null

Ivan
22222222

Tito
66666666

null

null

null

Rildo
44444444

Vânia
88884444

null

Pedro
77777777

null

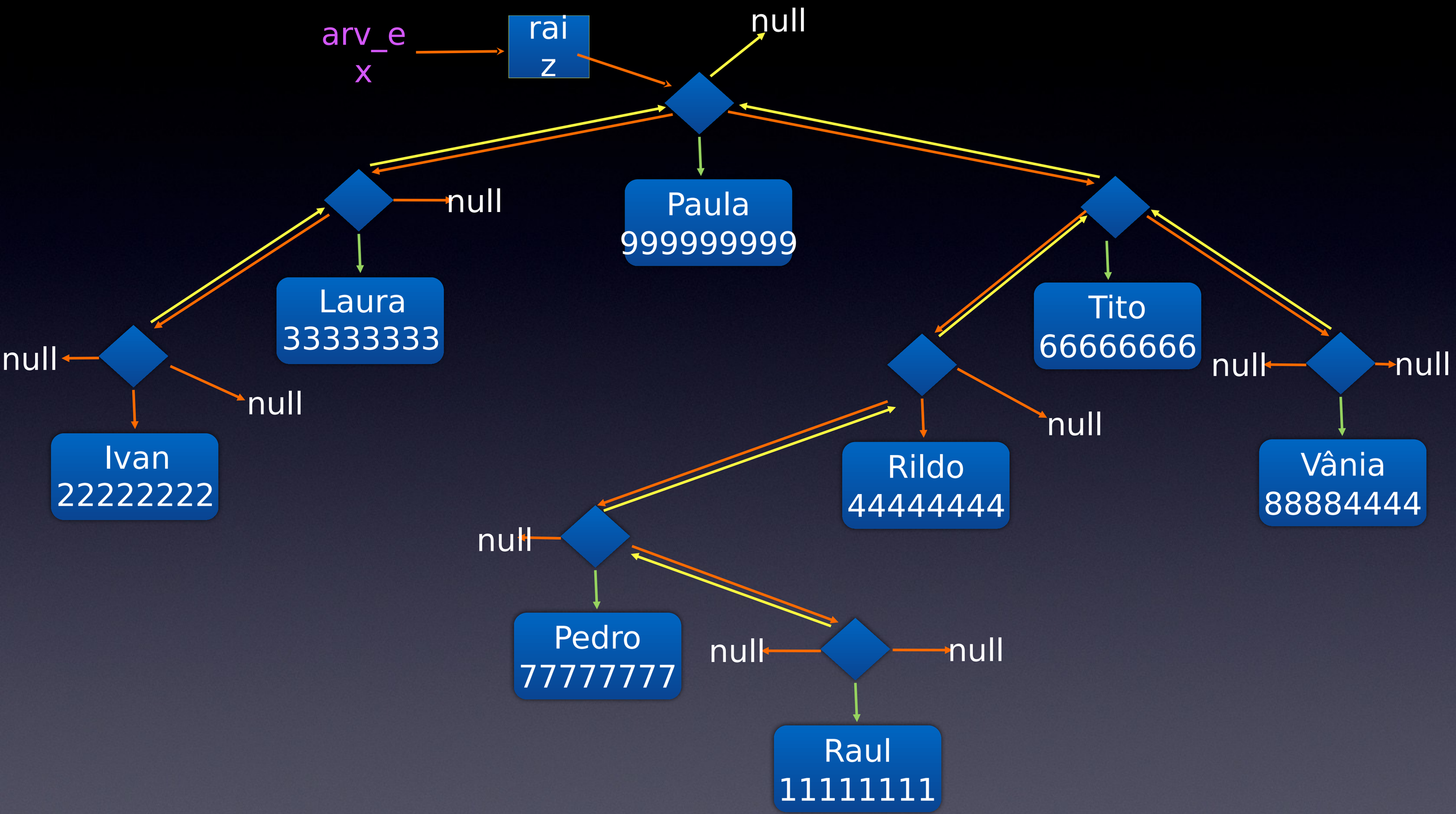
null

Raul
11111111

aux

Horácio
55555555

Árvore após 2 remoções



Item retirar(Item obj)

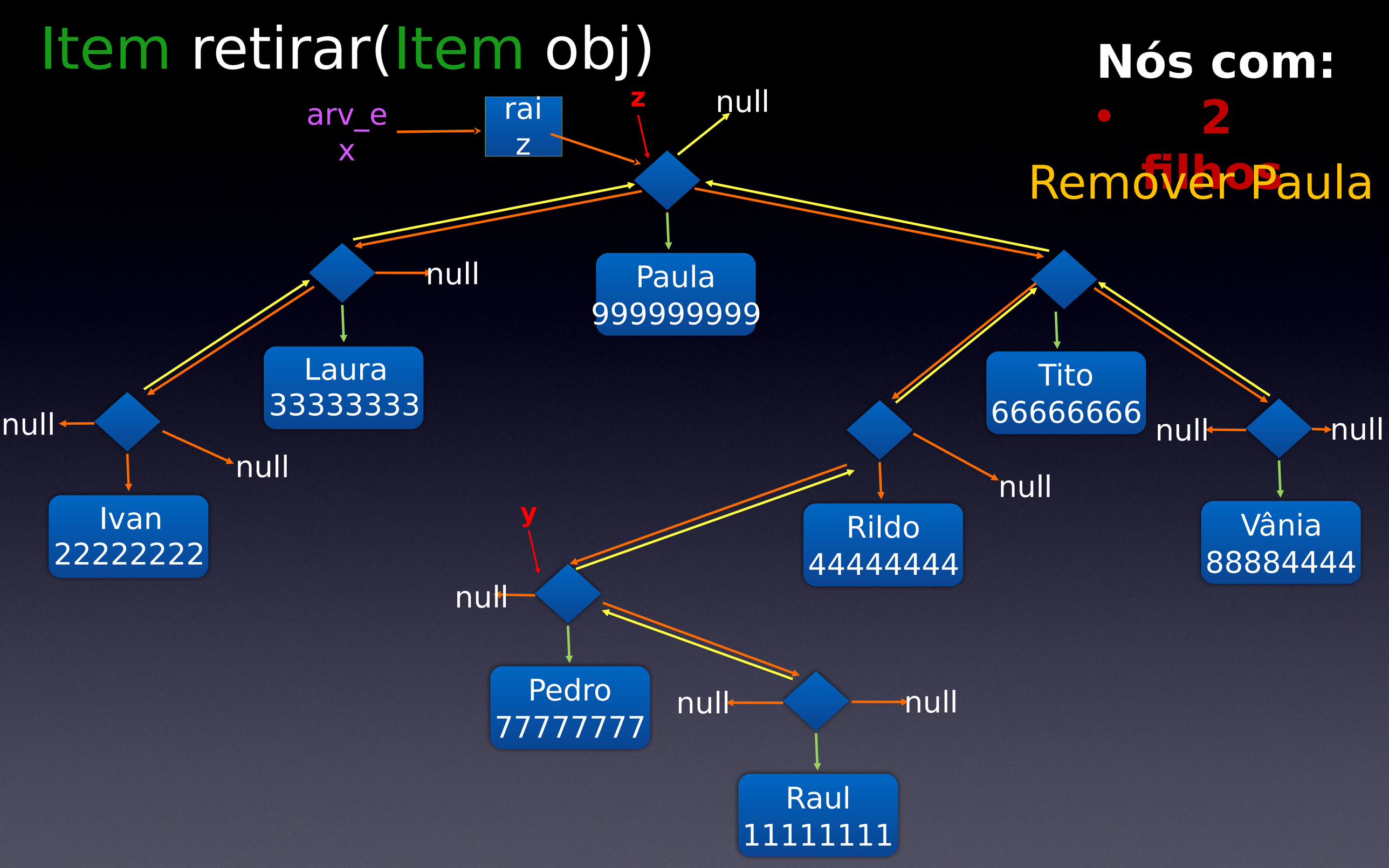
- Retirar Nós com 2 filhos.
  - No z = consultar(obj);
  - Sucesso:
    - Salva-se os dados numa variável “Item aux”.
    - Busca o sucessor de “No z” chamando:
      - No y = sucessor(z) ou No y = mínimo(z.fd)
    - Copia dos dados de No y para No z
    - Desliga-se “No y” da árvore.
      - No y pode ter somente:
        - nenhum filho ou 1
        - 1 filho
    - Retorna “Item aux”.
- Insucesso
  - Retorna null.

Item retirar(Item obj)

Nós com:

- **2 filhos**

Remover Paula

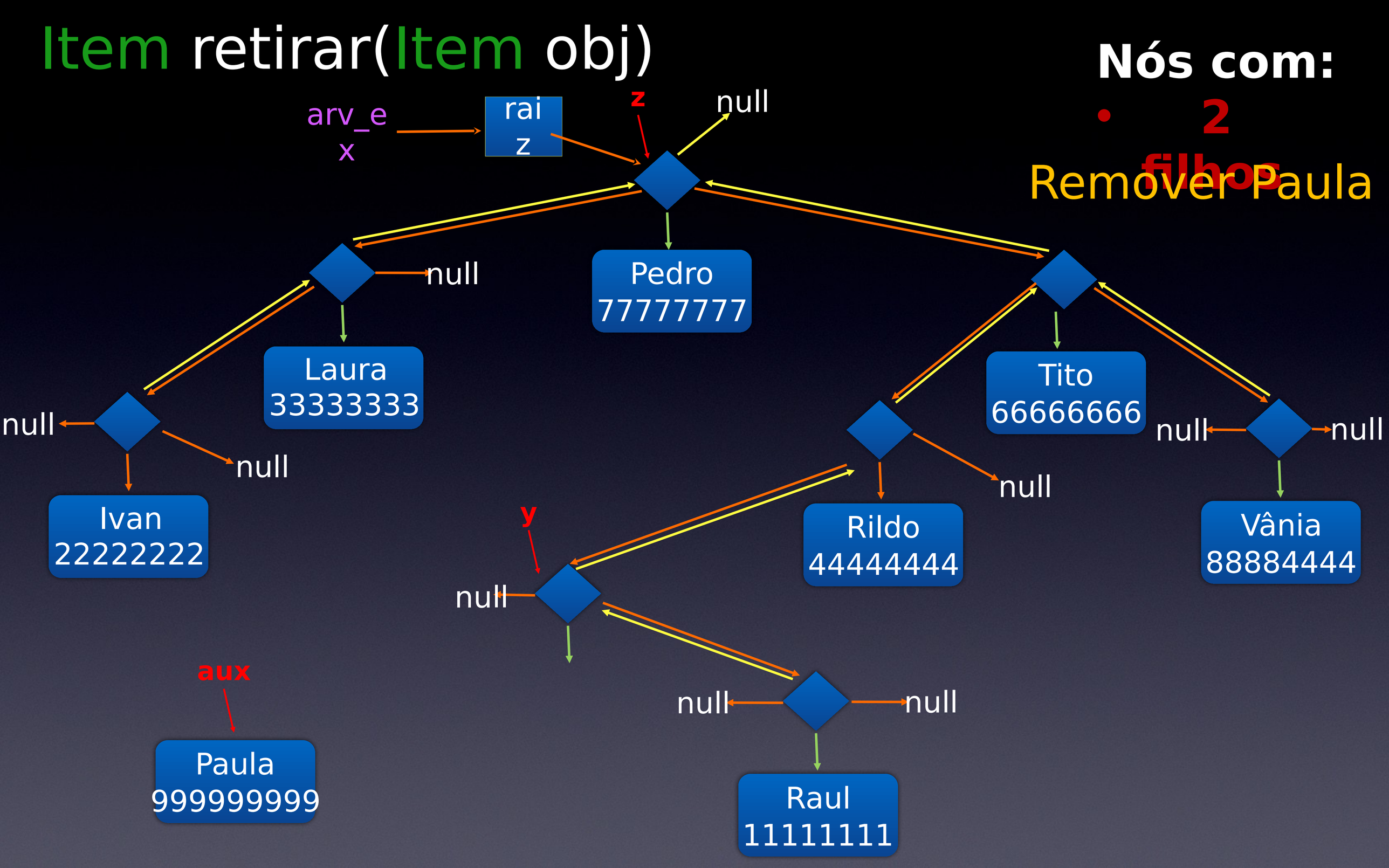


Item retirar(Item obj)

Nós com:

- **2 filhos**

Remover Paula

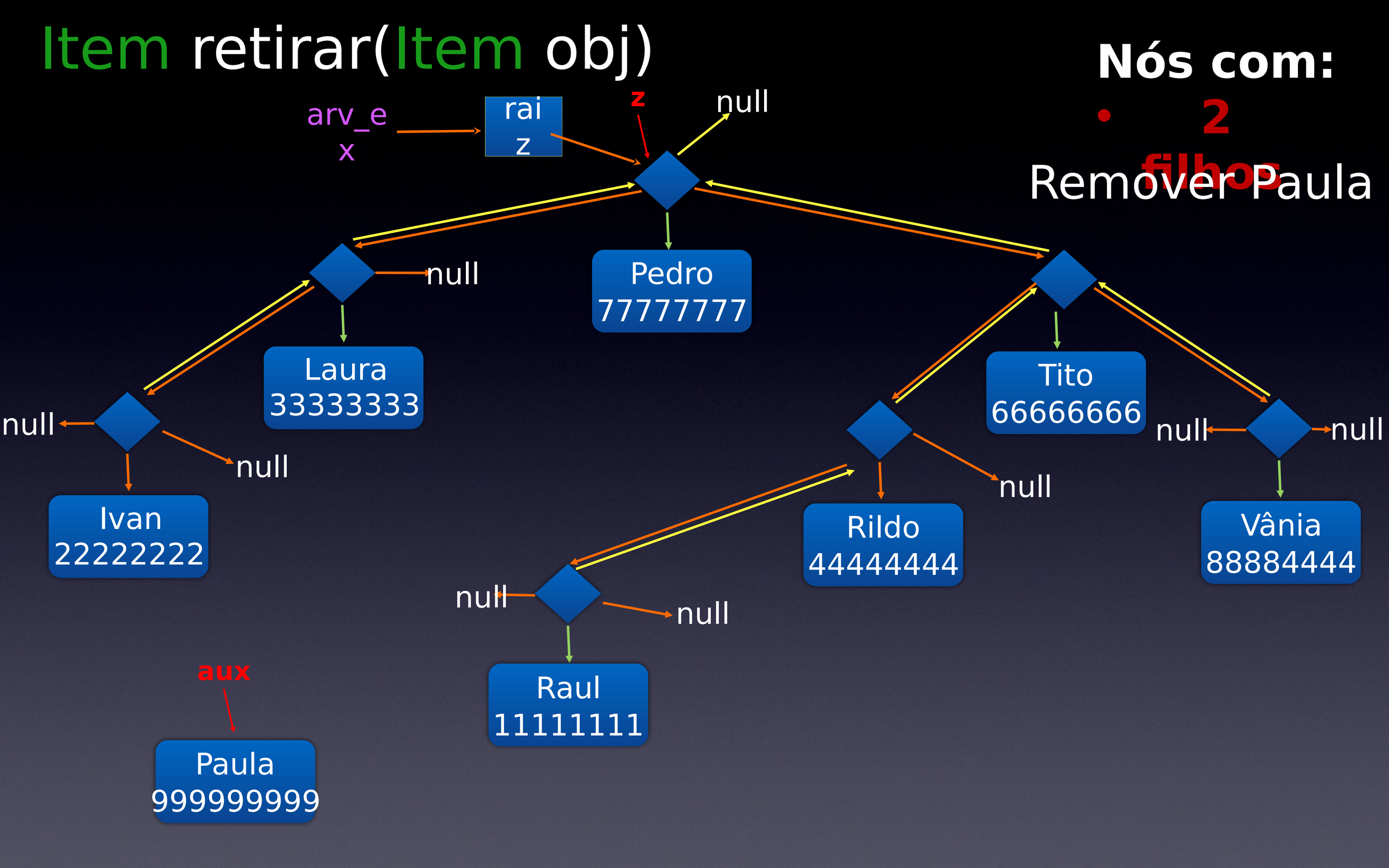


Item retirar(Item obj)

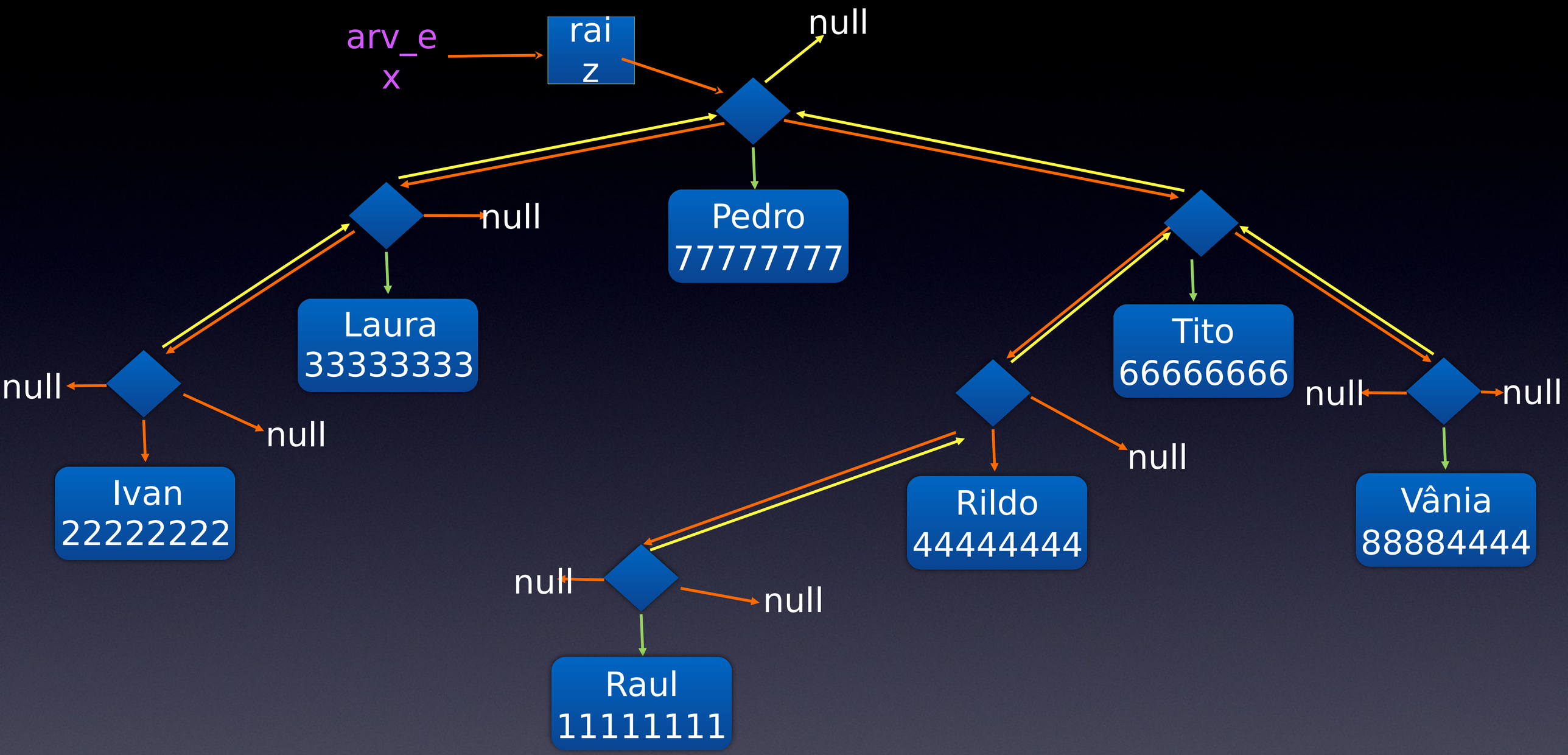
Nós com:

- **2 filhos**

Remover Paula



Árvore após 3 retiradas



```java
public Item retirar(Item obj) {
   Item aux = null;
   No z = consultar(obj);
   if(z != null) {
     aux = z.dados;
     No y = null;
     No x = null;
     if(z.fd == null || z.fe == null) { y = z; } // z tem 1 filho só ou nenhum filho
     else { y = sucessor(z); } // z tem dois filhos
     if(y.fe != null) { x = y.fe; }
     else { x = y.fd; }
     if(x != null) { x.pai = y.pai; } // pois y tem um filho
     if(y.pai == null) {//y é a raiz
       raiz = x;
       if(x != null) { x.pai = null; } // pois y tem um filho
     }
     else {//y não é raiz
        if(y == y.pai.fe) { y.pai.fe = x; }
        else { y.pai.fd = x; }
     }
     if(y != z) { z.dados = y.dados; } // y é o sucessor de z --> copia dados de y
para z
     tamanho--;
   }
   return aux;
}
```