

QuadTree

Conceito e Implementação

Fernando Morgado Pires Neto

Árvore



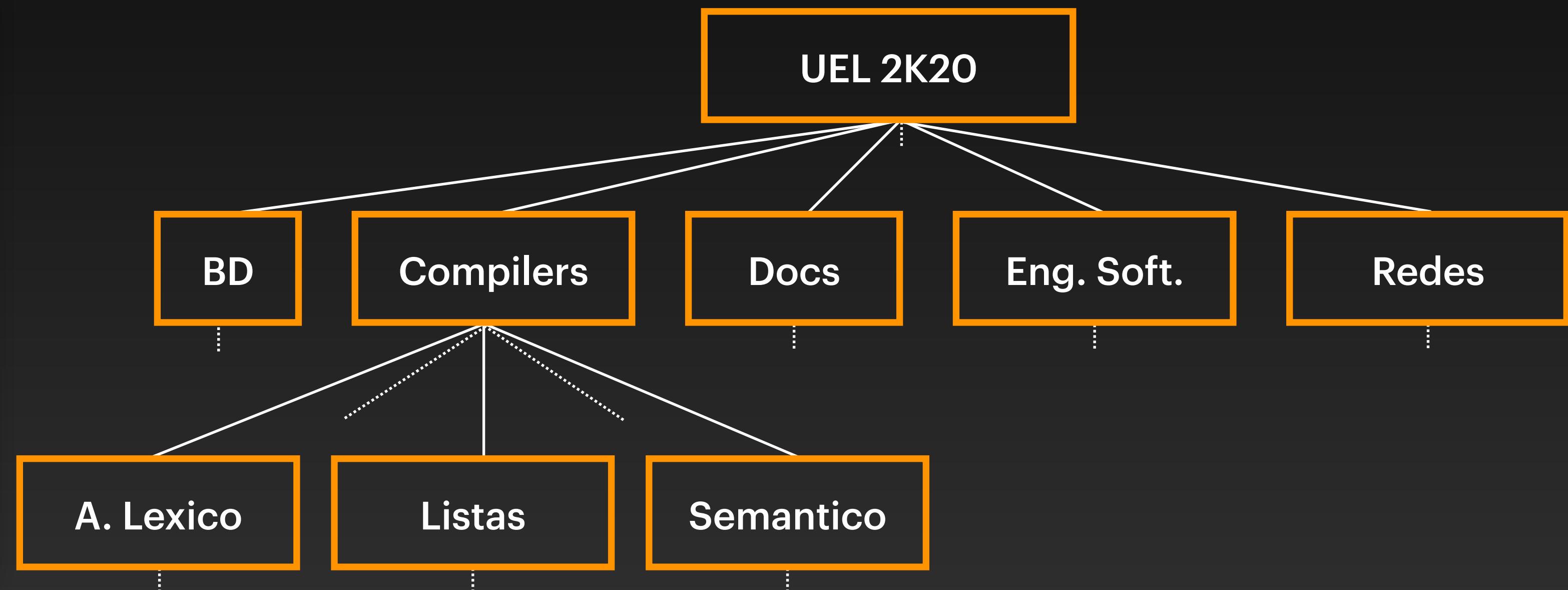
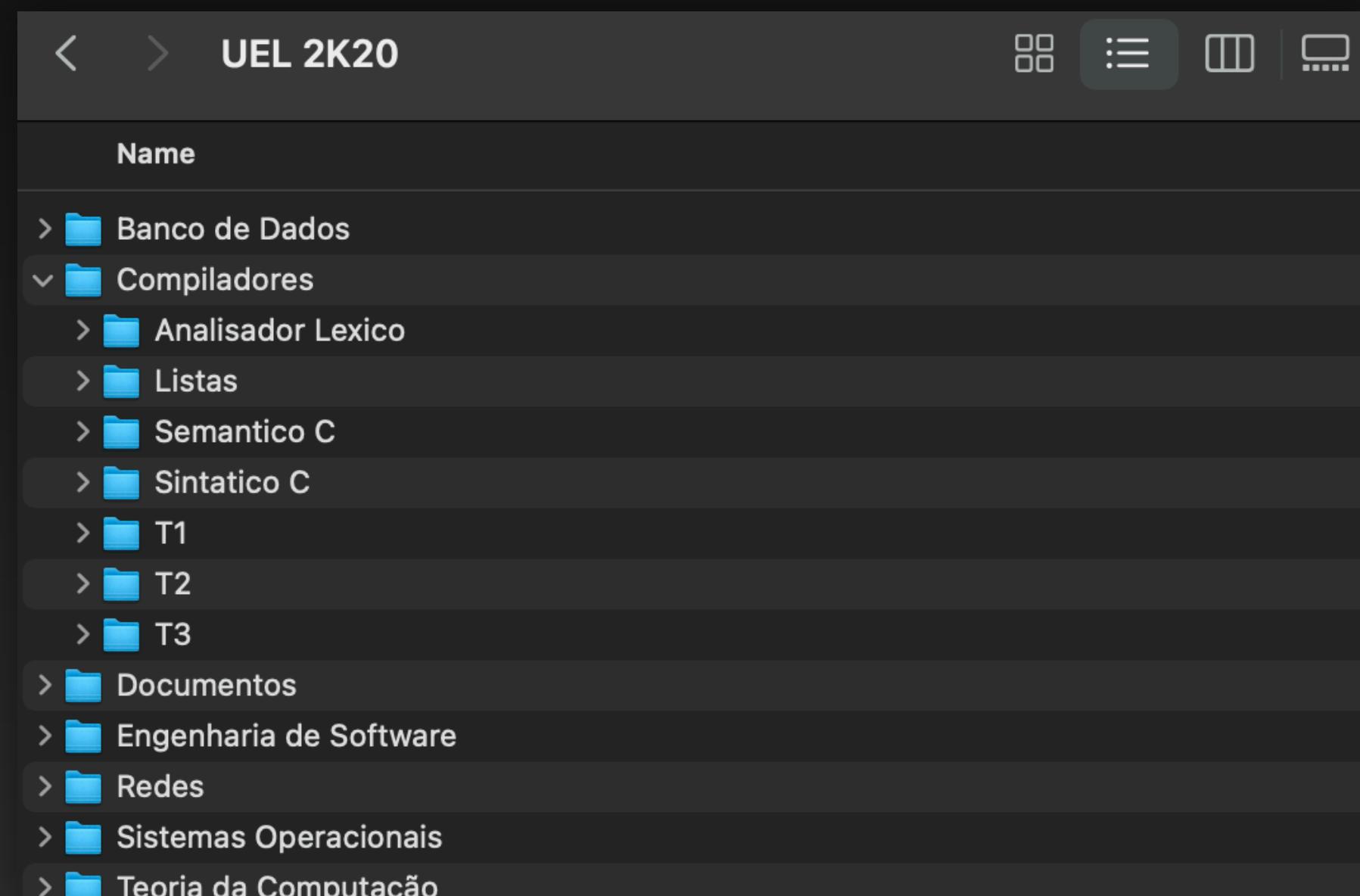
Árvore

Definição

- Uma das estruturas de dados mais importantes
- Organiza seus dados de forma hierárquica
- Diversas situações do mundo real podem ser reproduzidas em uma estrutura de árvore

Árvore

Exemplo - Estrutura de Pastas

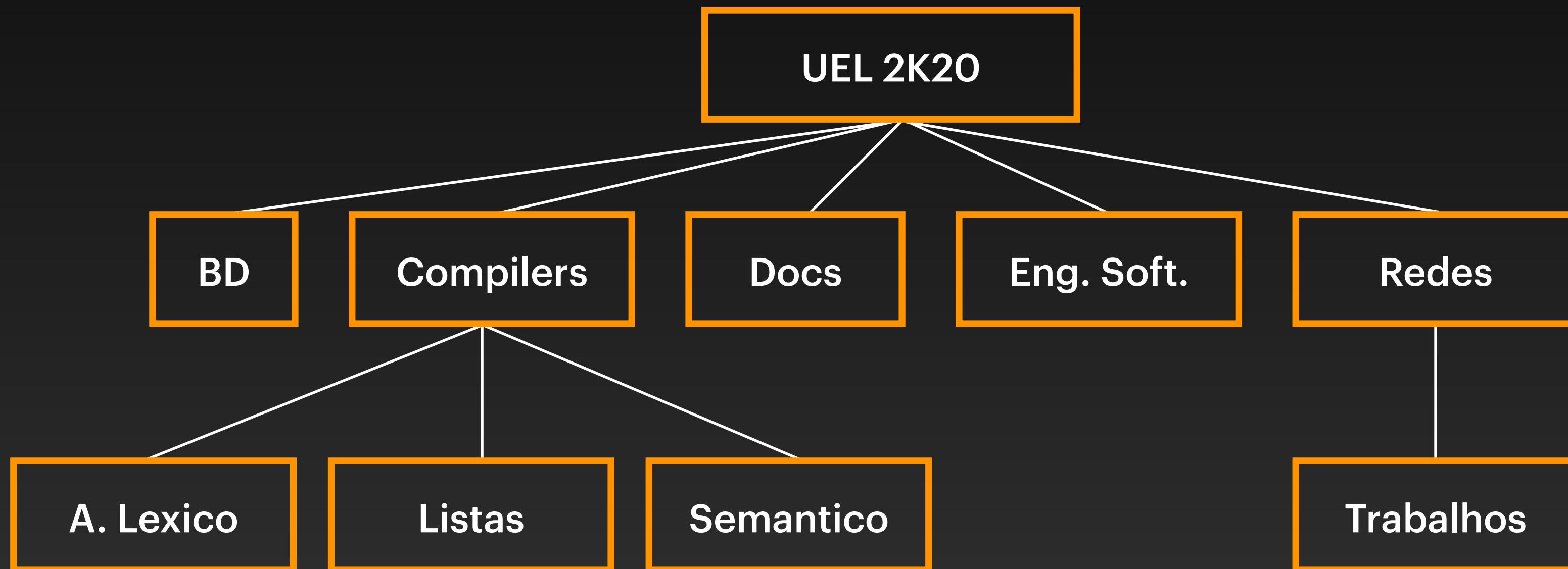


Árvore

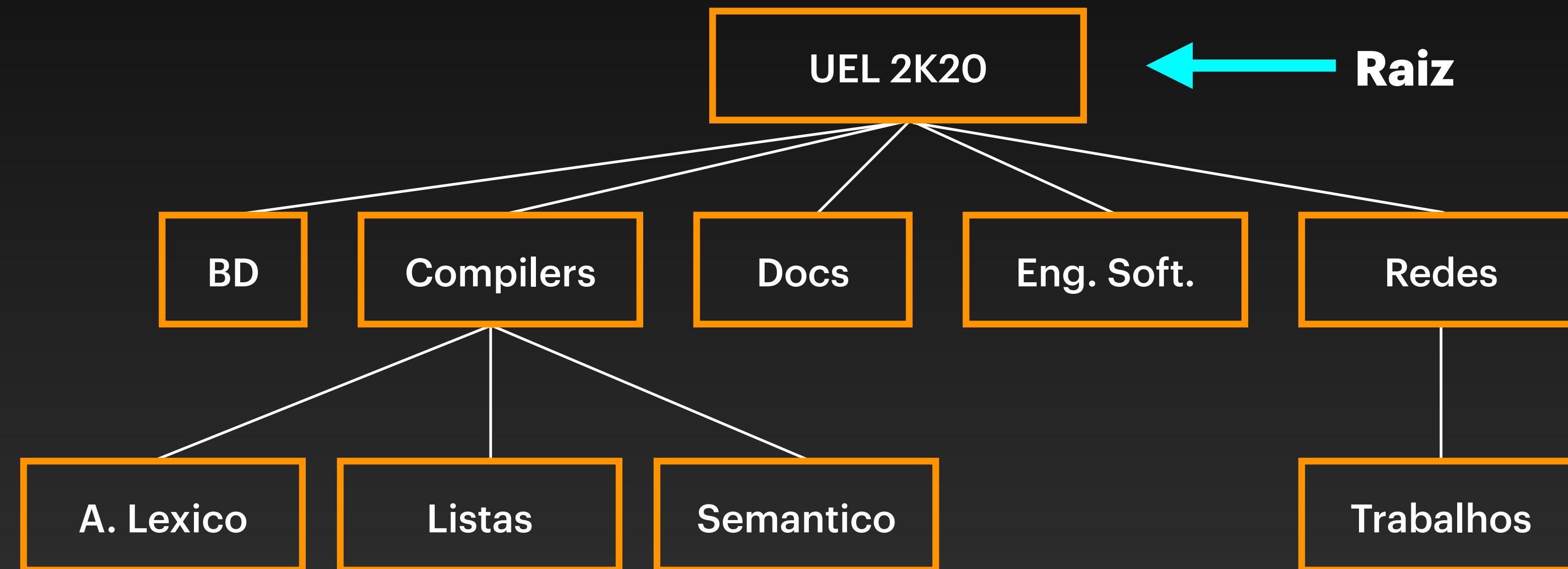
Estrutura

- Uma árvore é formada por um conjunto de elementos chamados de **nó**.
- Toda árvore possuí **um** nó chamado **raiz**.
 - A raiz é a **origem** de toda a árvore.
- Todo **nó** possui ligação a outros elementos chamados de **filhos**.
 - O último nó de uma ligação é chamado de **folha**.

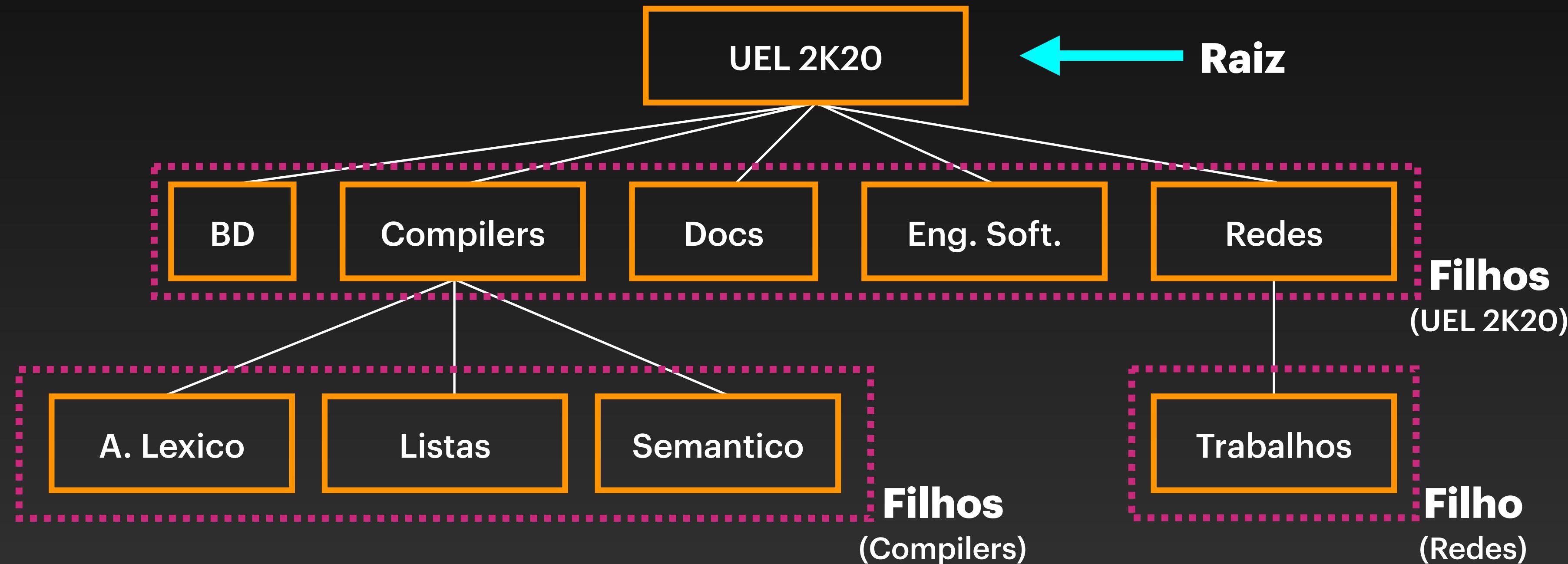
Árvore Estrutura



Árvore Estrutura

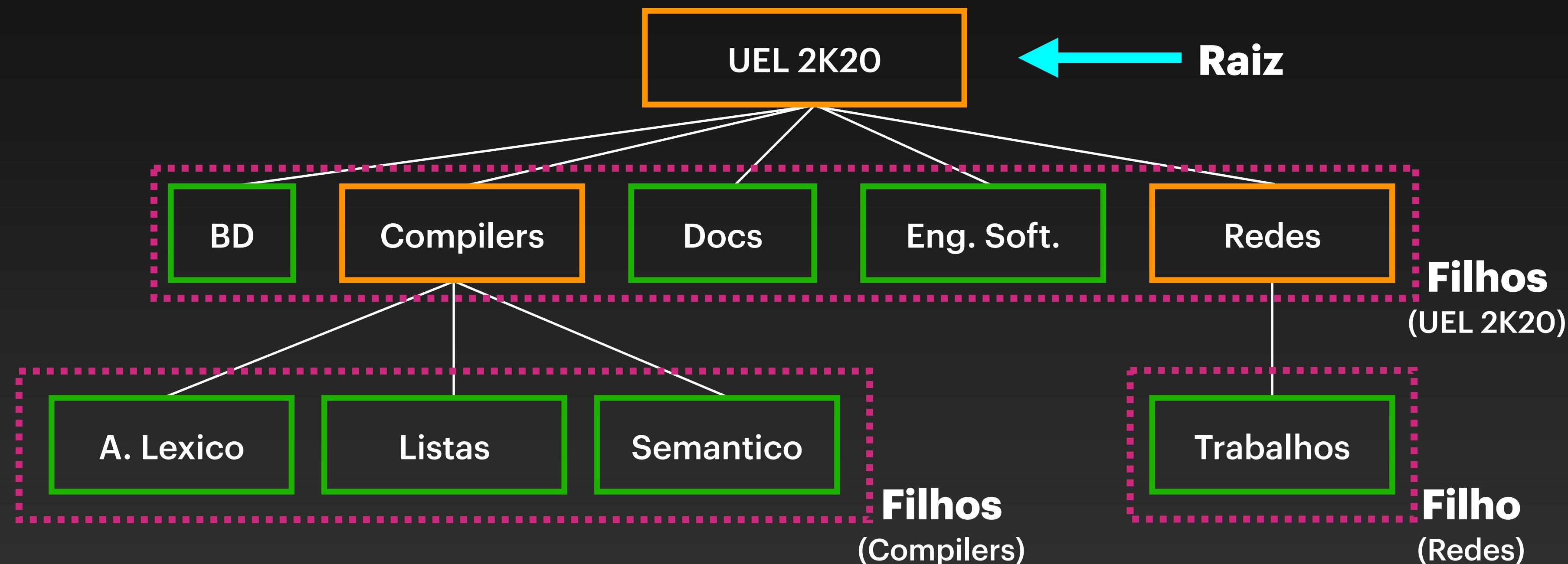


Árvore Estrutura



Filhos

Árvore Estrutura



Filhos

Folhas

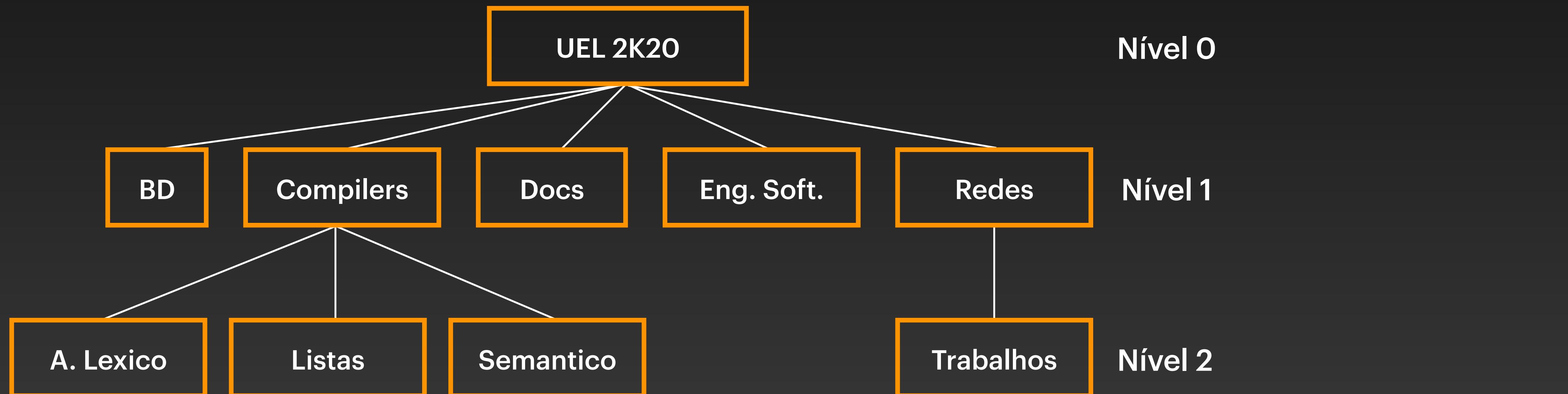
Árvore

Estrutura

- Cada nó (exceto a raiz) tem exatamente um antecessor imediato (**pai**)
- Cada nó tem sucessores imediatos (**filhos**)
 - Se um nó não tem sucessor = **folha**
- Os nós que são filhos do mesmo pai são **irmãos**

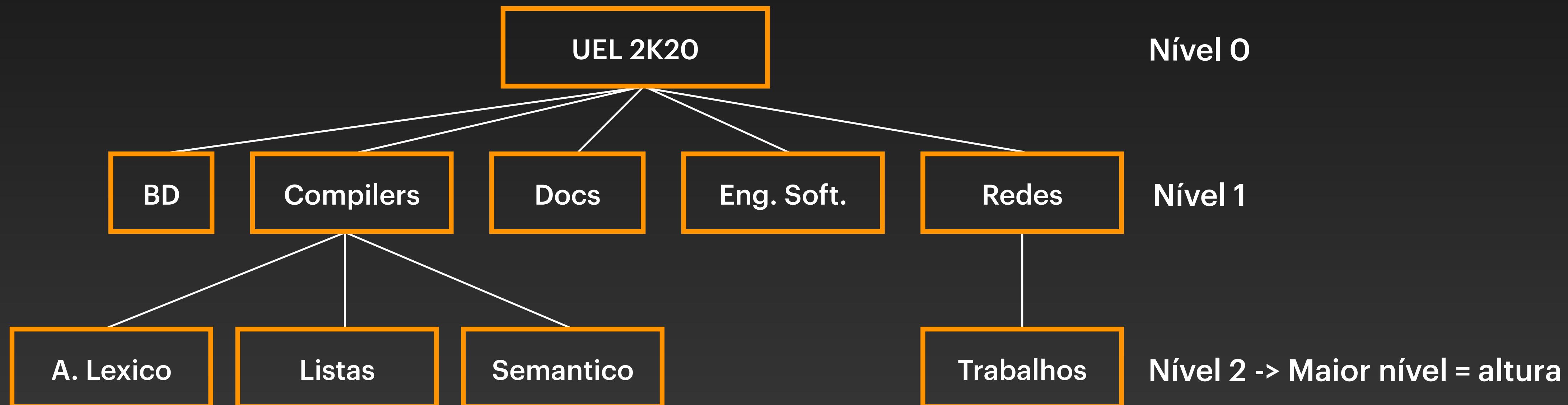
Árvore Estrutura

- Altura de uma árvore corresponde ao **maior nível**
- É a maior distância entre a raiz e qualquer nó



Árvore Estrutura

- Altura de uma árvore corresponde ao **maior nível**
- É a maior distância entre a raiz e qualquer nó



Manipulando uma Árvore



Manipulando uma Árvore

- A árvore, como qualquer estrutura de dados, aceita os quatro tipos de manipulação comuns: inserção, remoção, busca/leitura.
- Uma coisa que é comum entre todos os tipos de árvore é o uso da **recursão** para implementar essas operações.
- Contudo, o algoritmo de implementação varia de acordo com o **tipo de árvore** e a **estrutura hierárquica** a ser implementada.

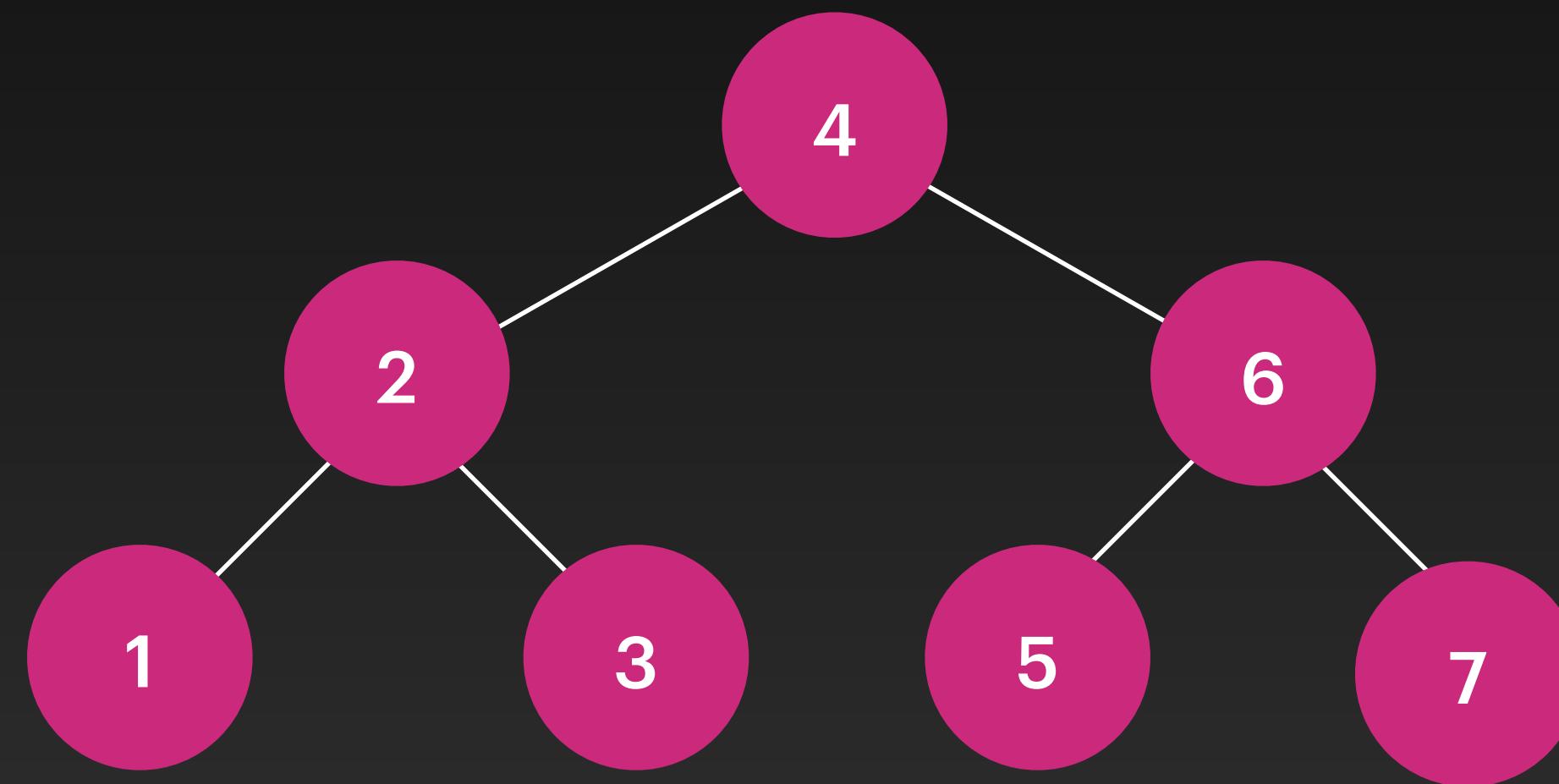
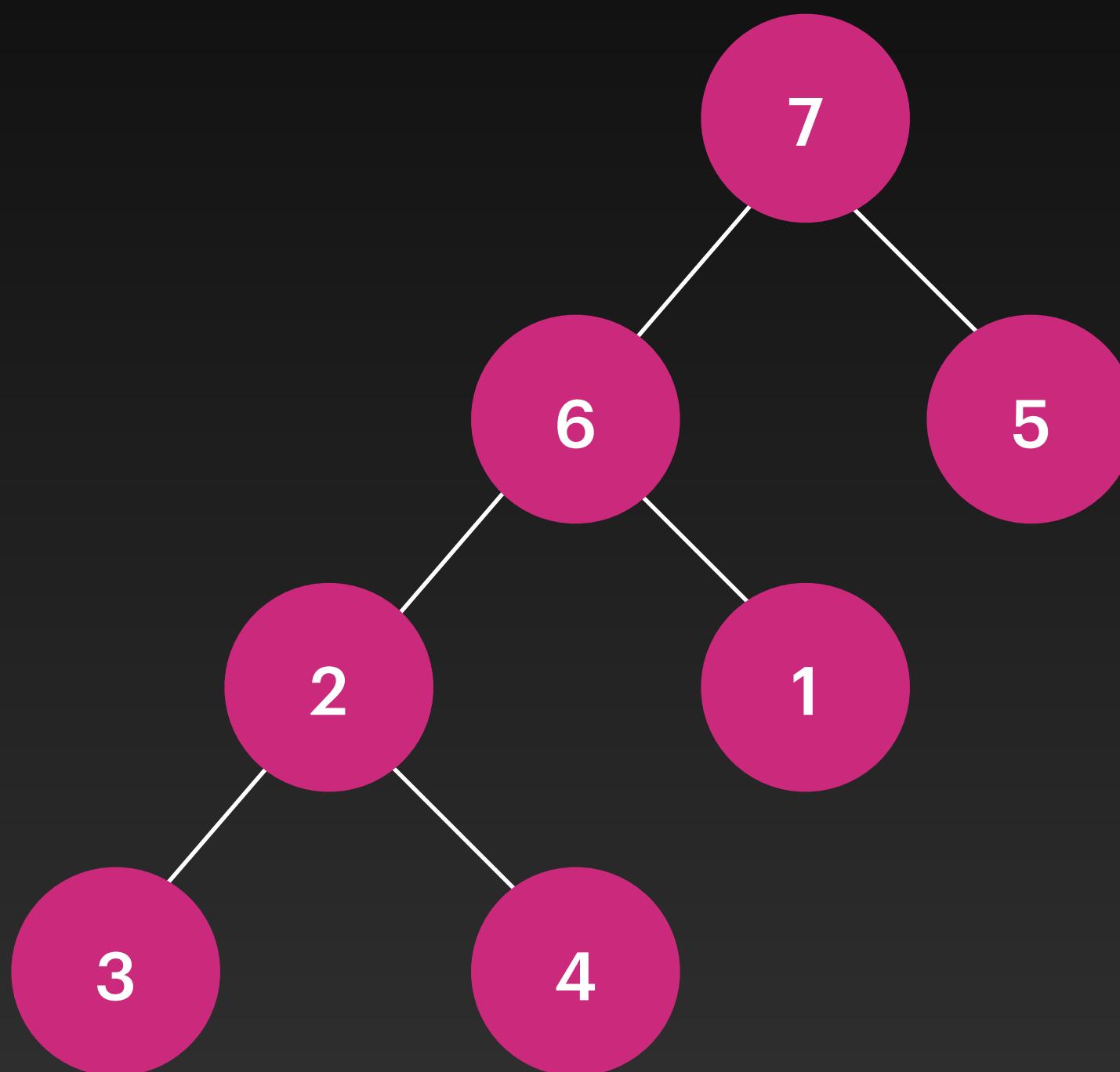
Manipulando uma Árvore

Inserção

- A **inserção** é um dos pontos mais críticos durante a criação de uma árvore.
- Isso pois, para manter sua eficiência na hora da busca, os nós de uma árvore devem estar **balanceados**.

Manipulando uma Árvore

Inserção



Qual é a diferença entre as duas árvores?

Manipulando uma Árvore

Inserção

- O algoritmo de inserção varia de acordo com o **tipo de árvore** e a **hierarquia a ser considerada**.
- Porém, para manter uma árvore balanceada, a dica é sempre considerar um **valor** que consiga se relacionar com a hierarquia de forma a atingir esse objetivo.

Manipulando uma Árvore

Remoção

- Remover um nó da árvore também varia de acordo com o tipo da árvore.
- Porém, algumas coisas devem ser levadas em consideração:
 - Em uma árvore **balanceada**, sempre que um nó for removido, o algoritmo de balanceamento deve ser executado novamente para manter o equilíbrio.
 - Caso a **raiz da árvore** seja removida, ela deve ser substituída por outro nó que satisfaça a mesma condição de balanceamento.

Manipulando uma Árvore

Busca

- Busca é uma operação comum em todas as árvores, e sua implementação normalmente indefere do tipo de árvore.
- Ela ocorre de maneira **recursiva** (busca em profundidade) ou então com o auxílio de uma **fila** (busca em largura).
- As implementações de busca também podem ser implementadas para percorrerem a árvore por completo.
 - No caso do método percorre, ele não retorna nenhum valor.

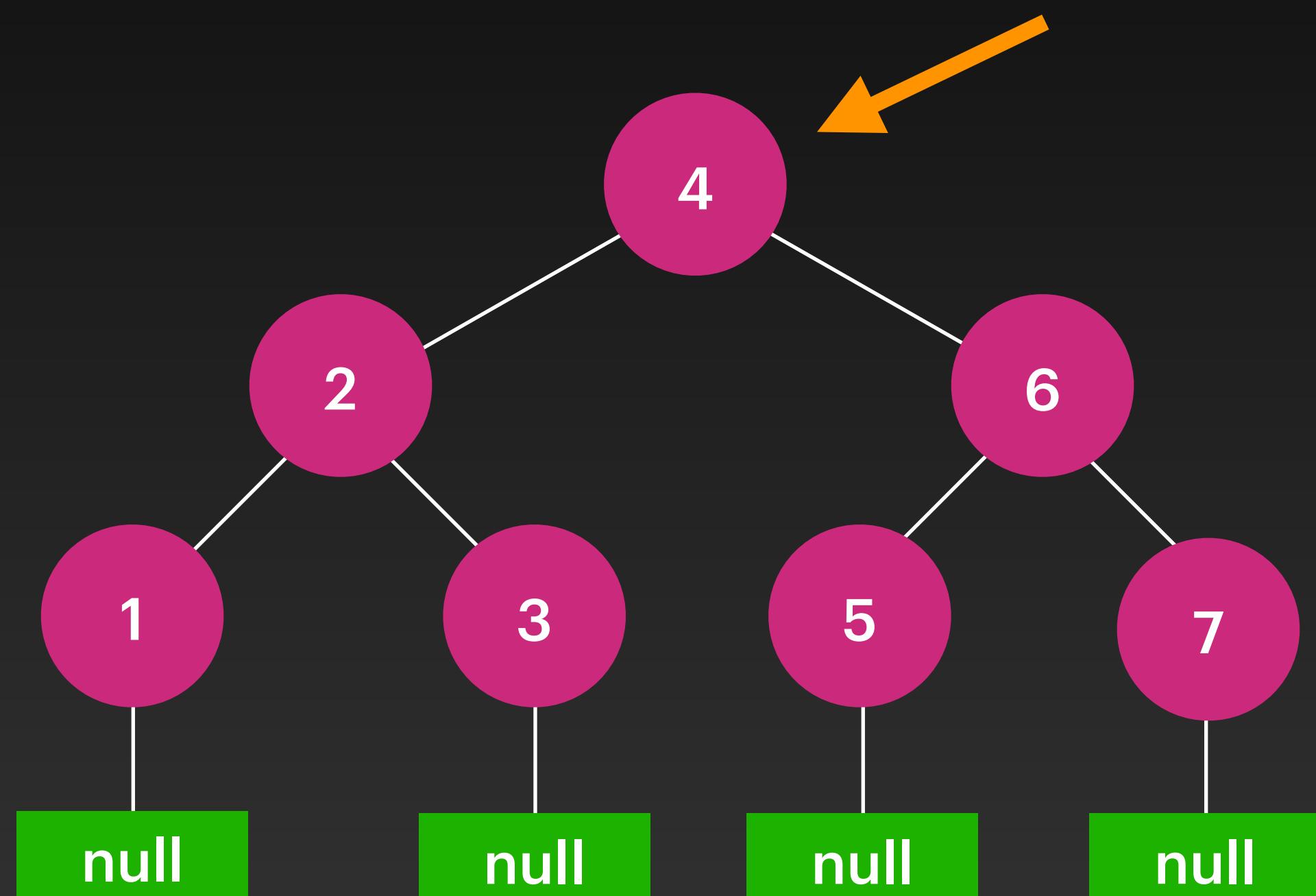
Manipulando uma Árvore

Busca em Profundidade

- Visito todos os filhos de um pai em busca do nó que desejo procurar.
- Caso chegar em um nó folha, volto no seu antecessor e procuro nos filhos dele.
- Operação **recursiva**.

Manipulando uma Árvore

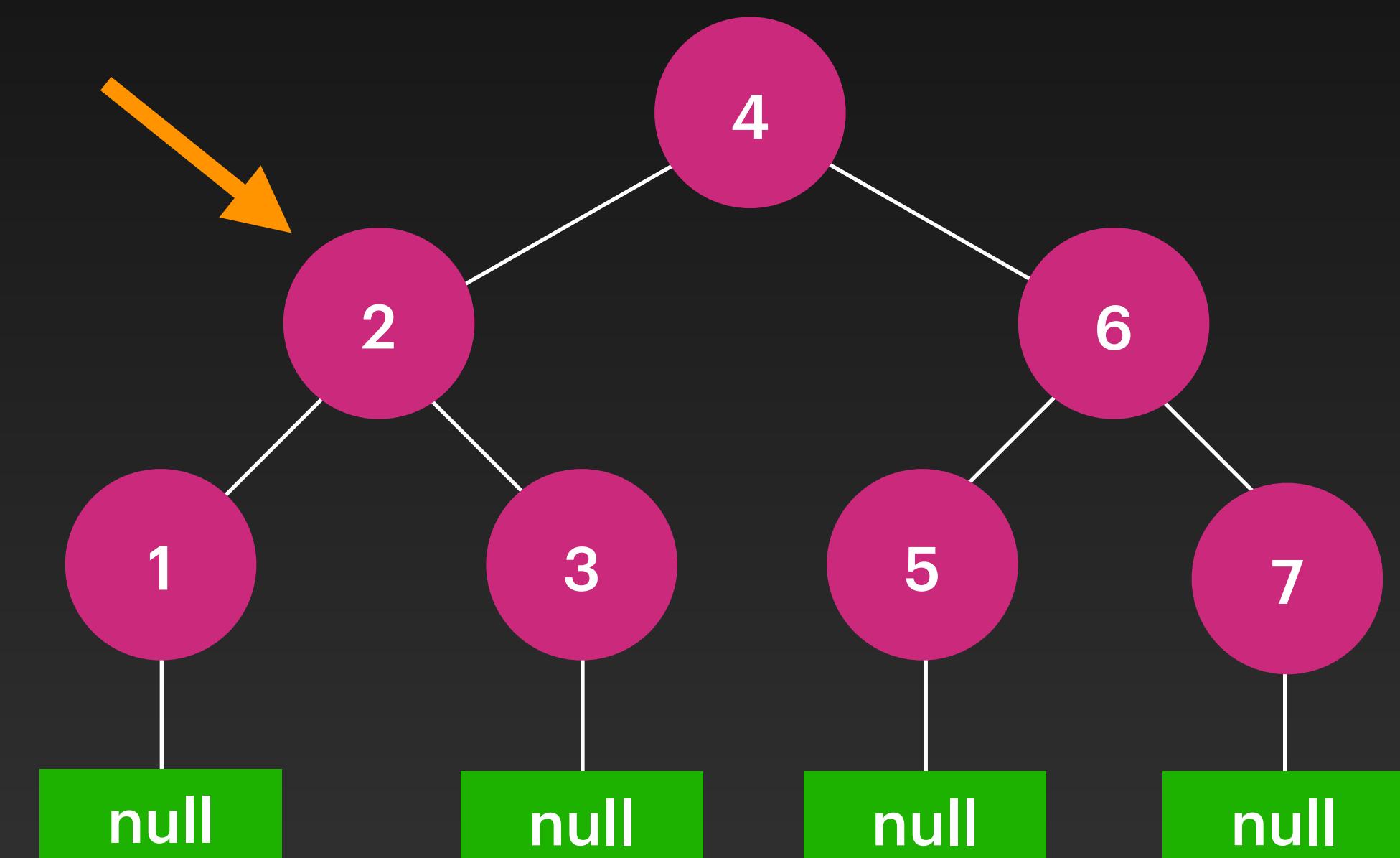
Busca em Profundidade - Exemplo 1



buscando o valor 3

Manipulando uma Árvore

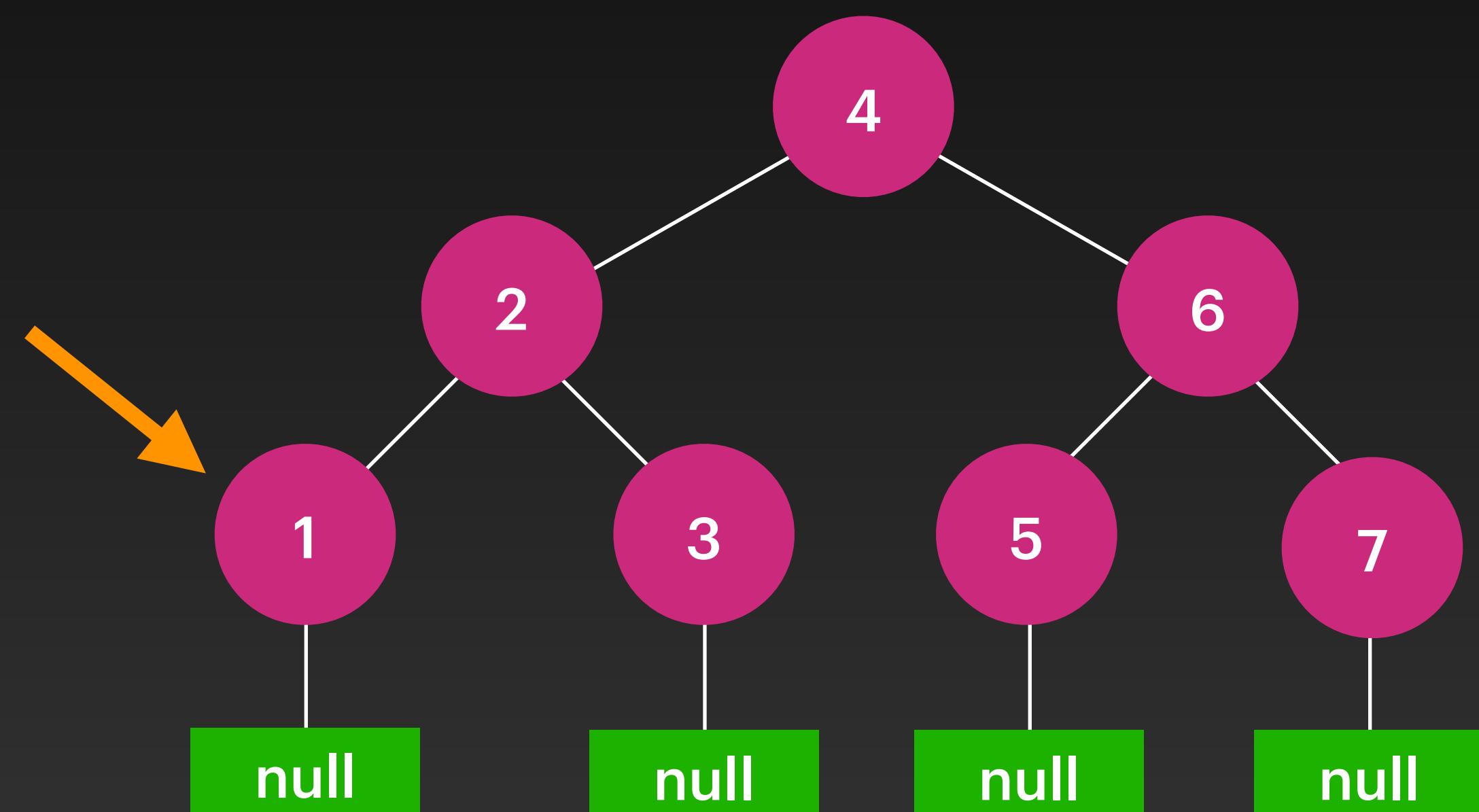
Busca em Profundidade - Exemplo 1



buscando o valor **3**

Manipulando uma Árvore

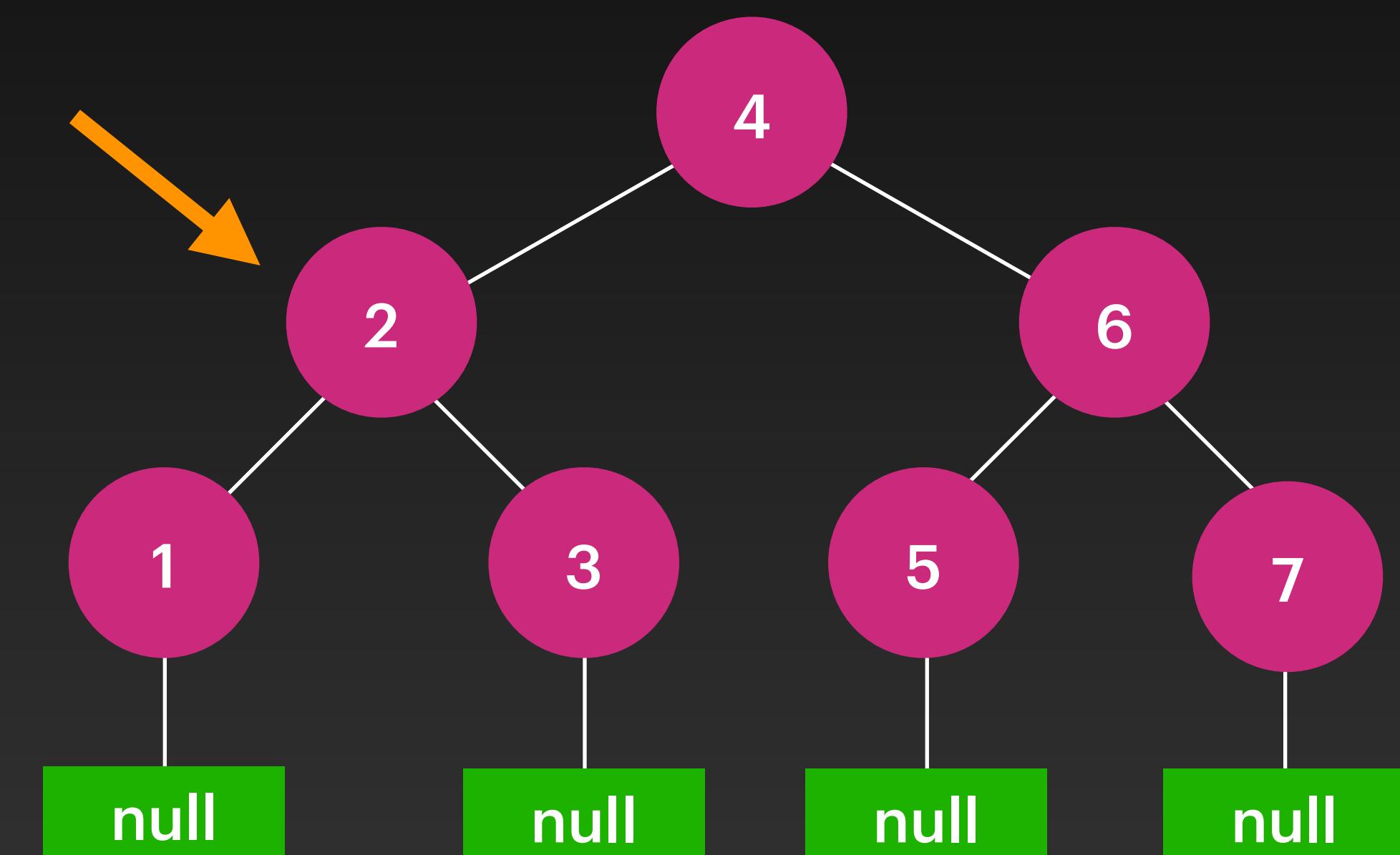
Busca em Profundidade - Exemplo 1



buscando o valor **3**

Manipulando uma Árvore

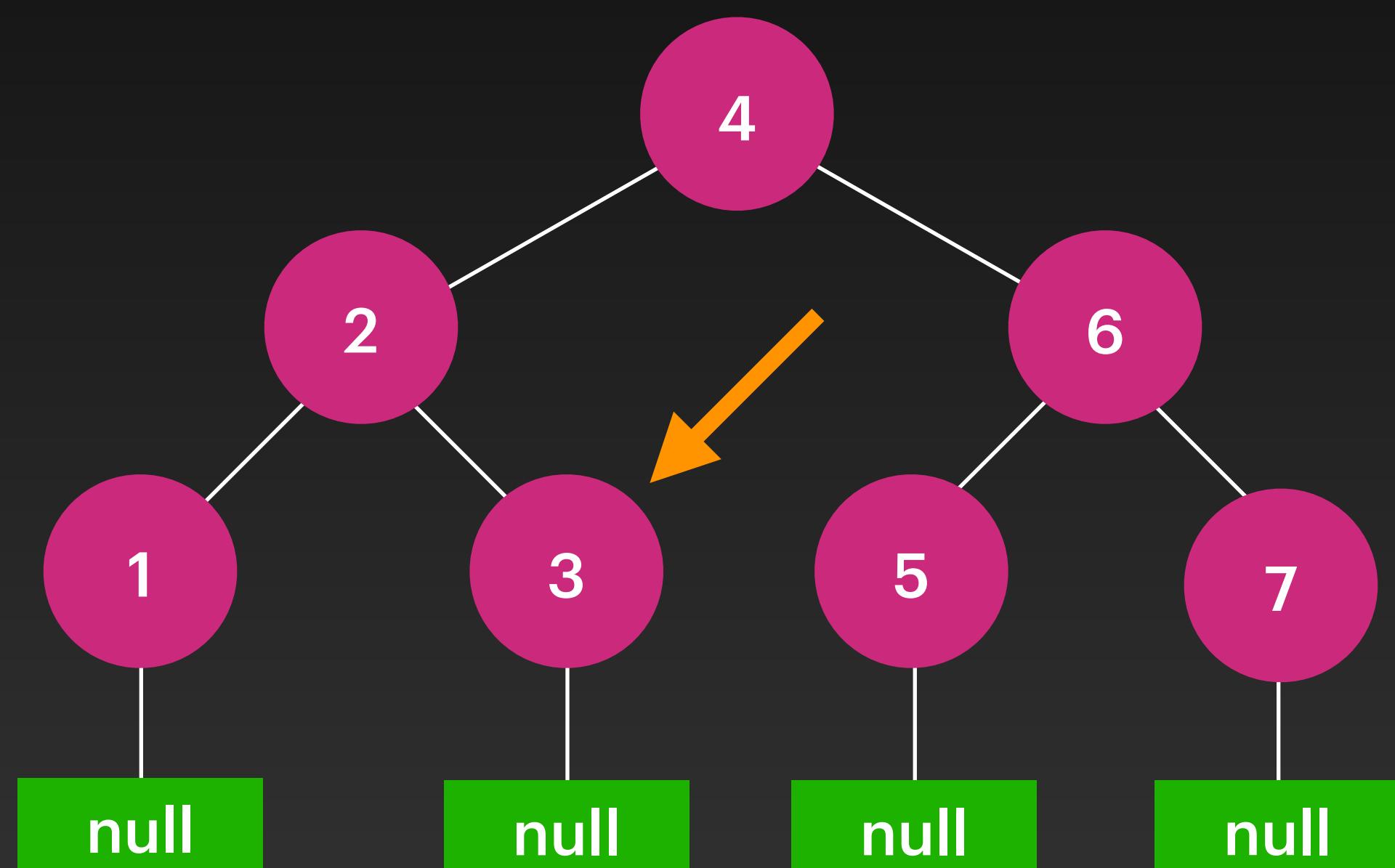
Busca em Profundidade - Exemplo 1



buscando o valor **3**

Manipulando uma Árvore

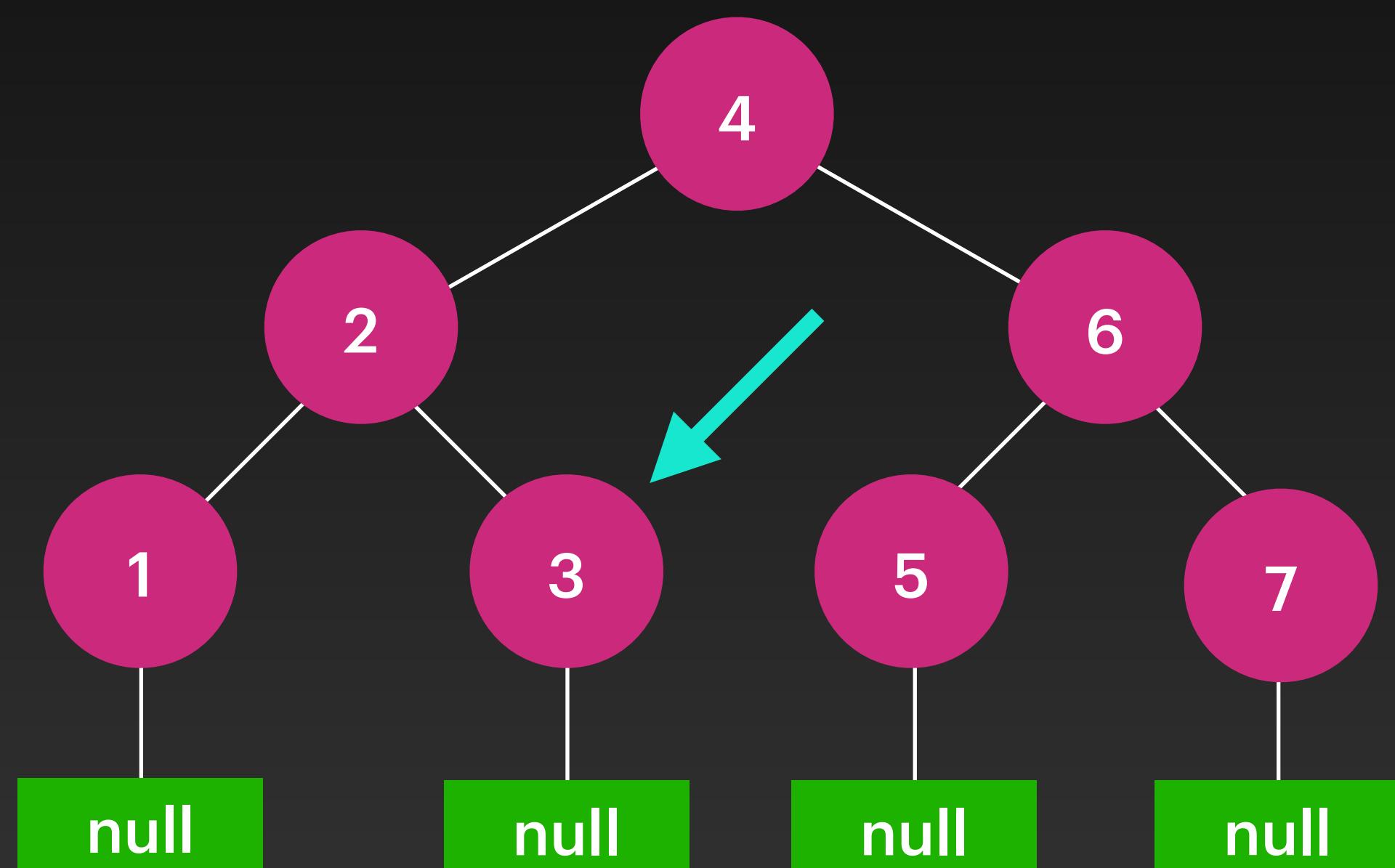
Busca em Profundidade - Exemplo 1



buscando o valor 3

Manipulando uma Árvore

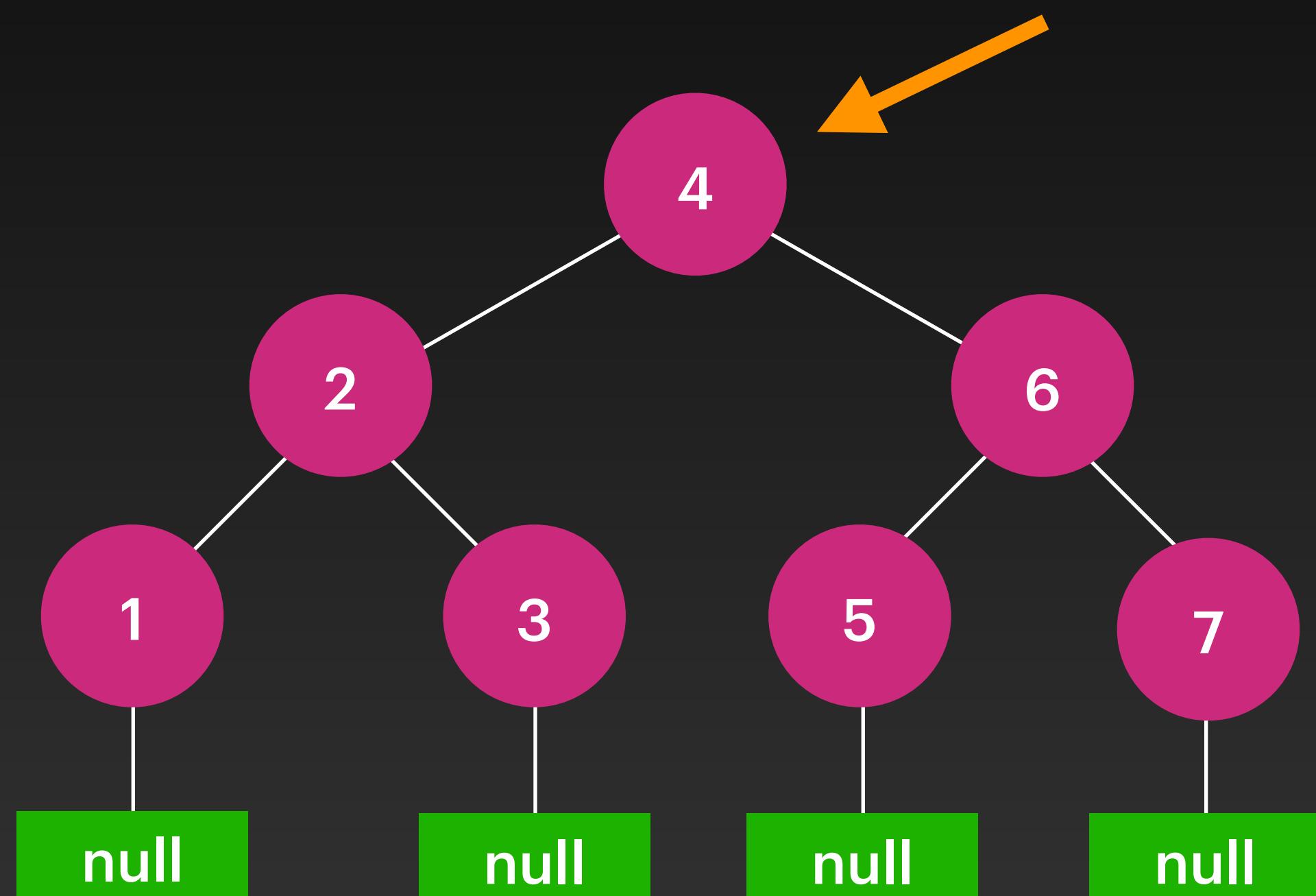
Busca em Profundidade - Exemplo 1



buscando o valor 3 ✓

Manipulando uma Árvore

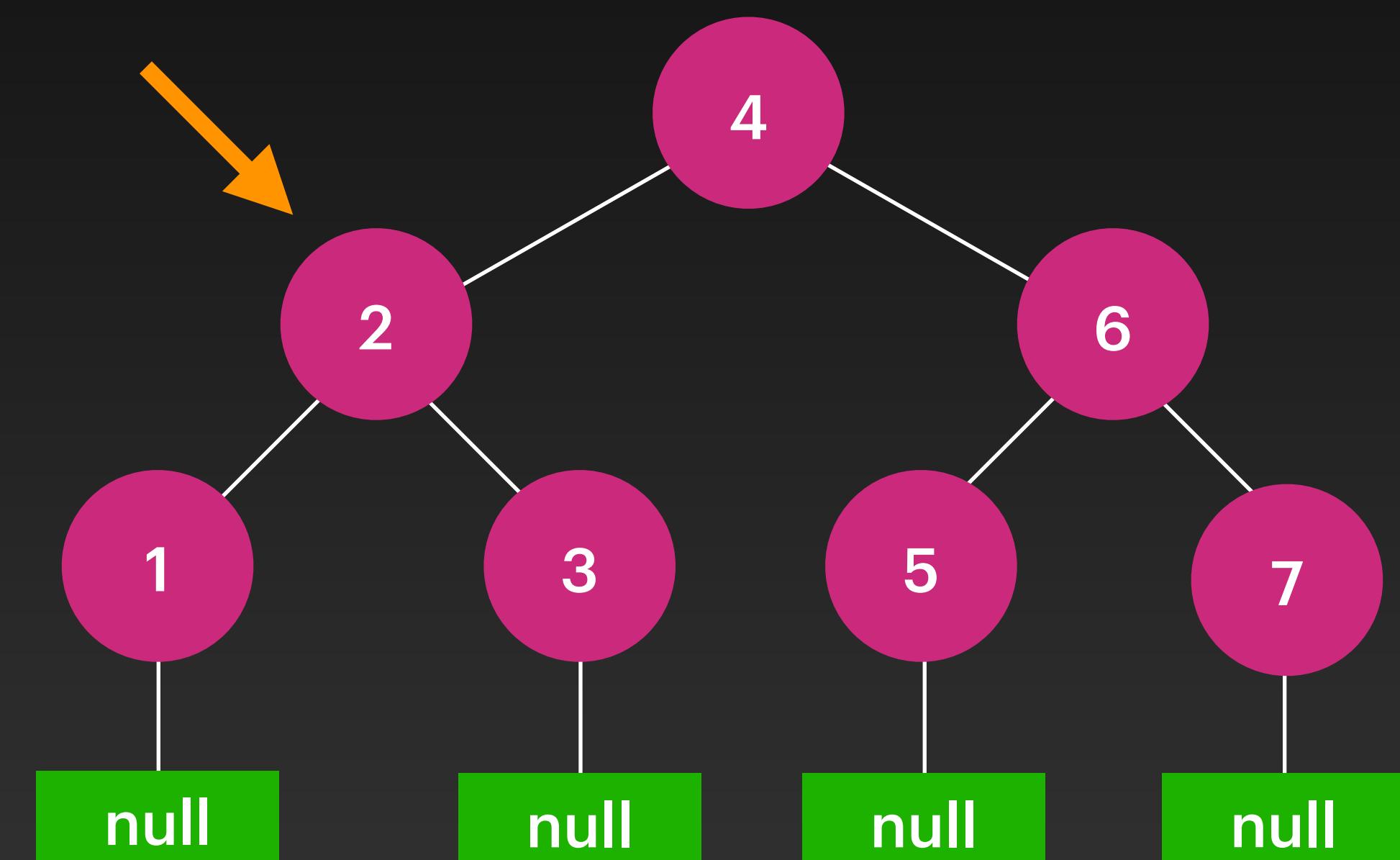
Busca em Profundidade - Exemplo 2



buscando o valor 3

Manipulando uma Árvore

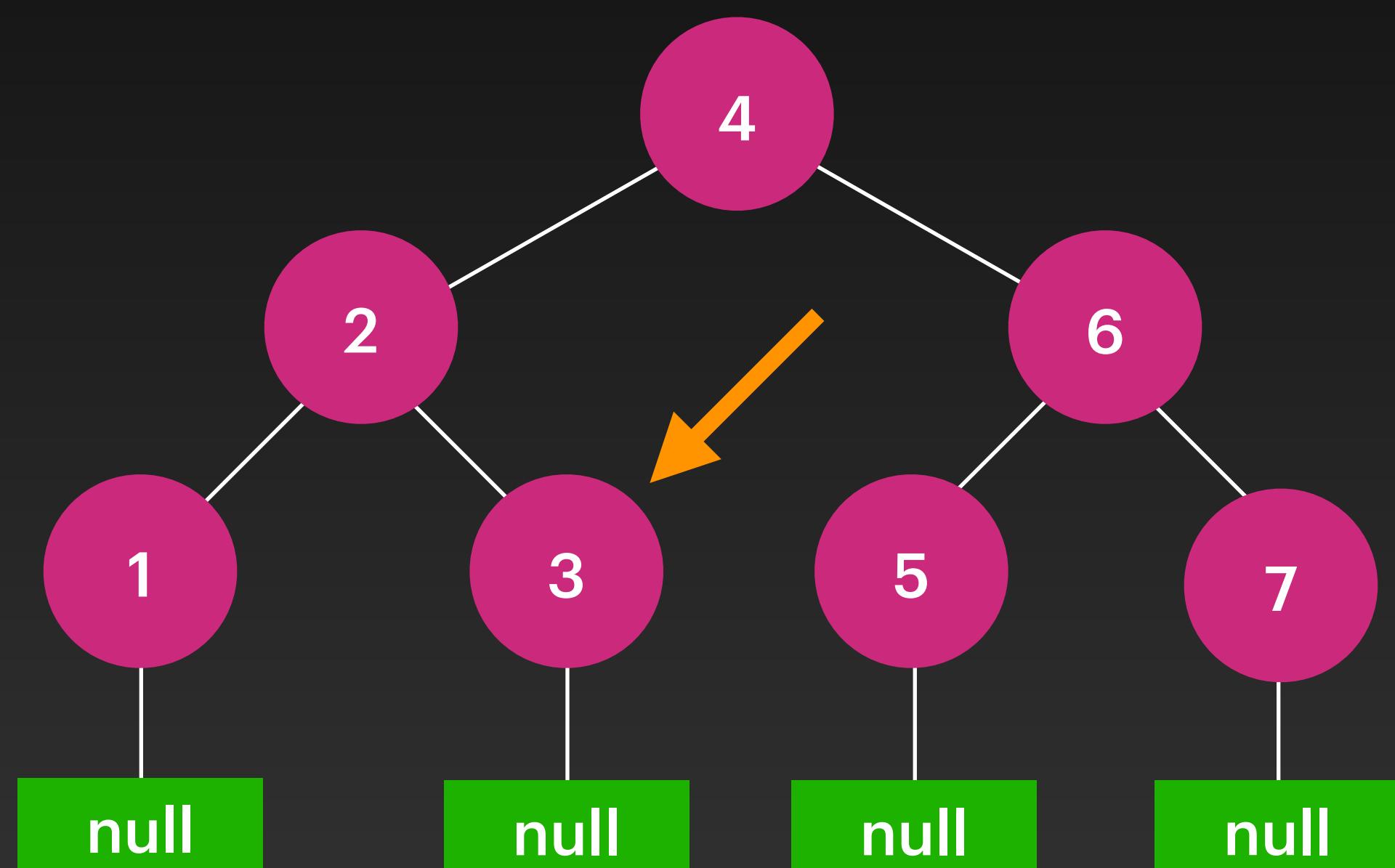
Busca em Profundidade - Exemplo 2



buscando o valor 3

Manipulando uma Árvore

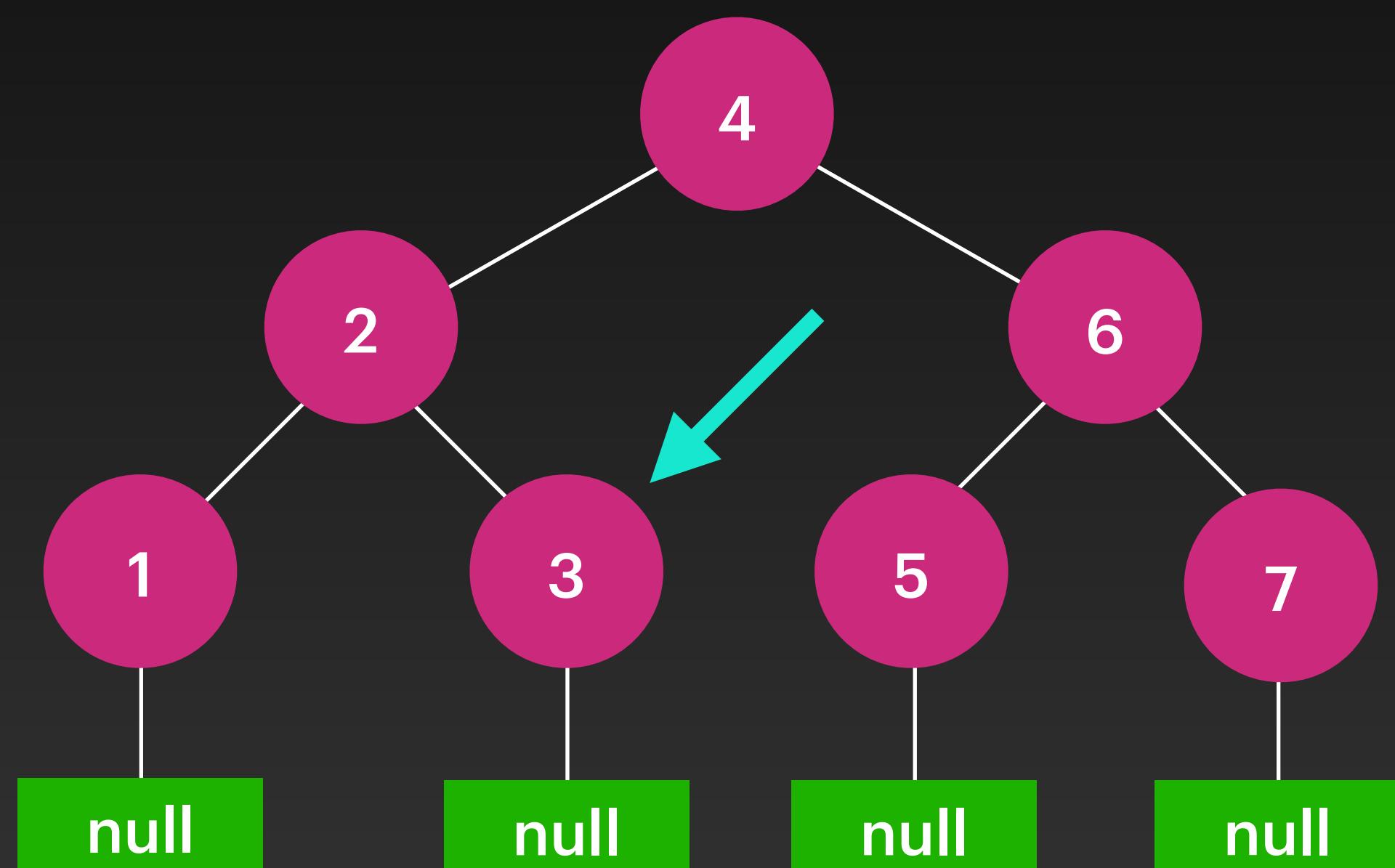
Busca em Profundidade - Exemplo 2



buscando o valor 3

Manipulando uma Árvore

Busca em Profundidade - Exemplo 2



buscando o valor 3 ✓

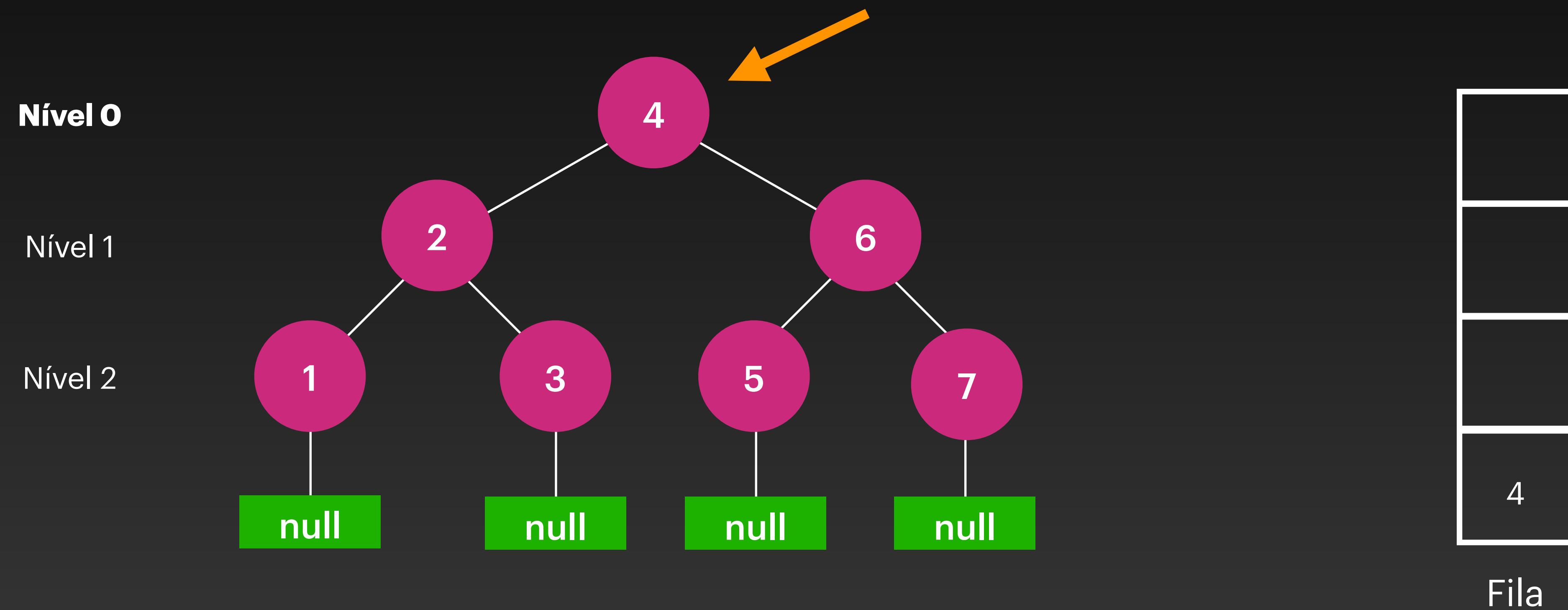
Manipulando uma Árvore

Busca em Largura

- Visito todos os nós referentes a um nível da árvore, da esquerda pra direita.
- Começo na raiz, visito os filhos da raiz, depois os filhos dos filhos, e assim sucessivamente...
- Utiliza a estrutura de fila para armazenar os nós referente ao nível que estou visitando.

Manipulando uma Árvore

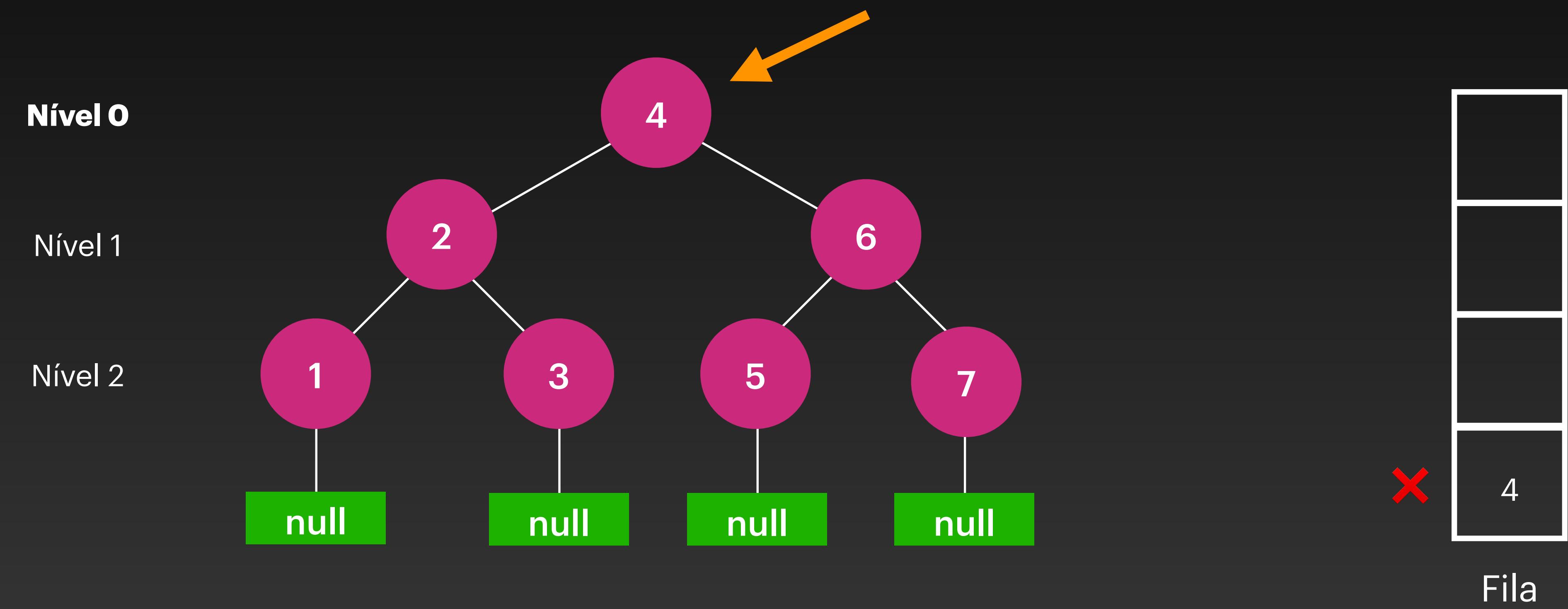
Busca em Largura - Exemplo



buscando o valor **3**

Manipulando uma Árvore

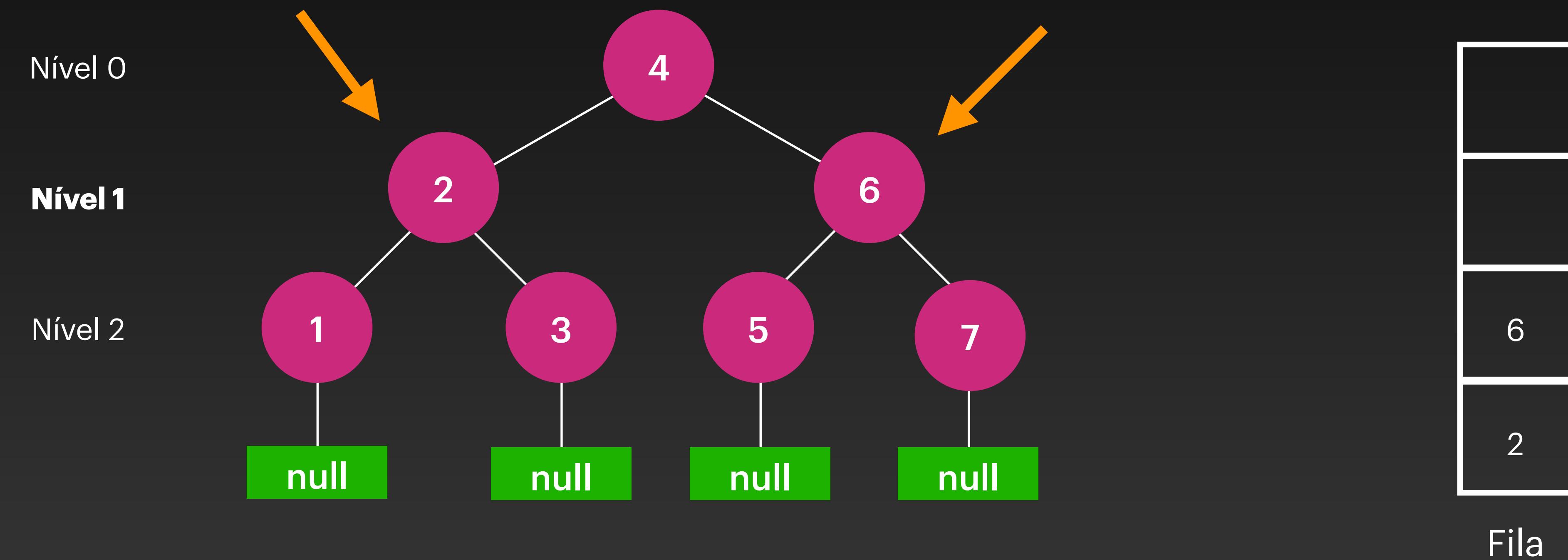
Busca em Largura - Exemplo



buscando o valor 3

Manipulando uma Árvore

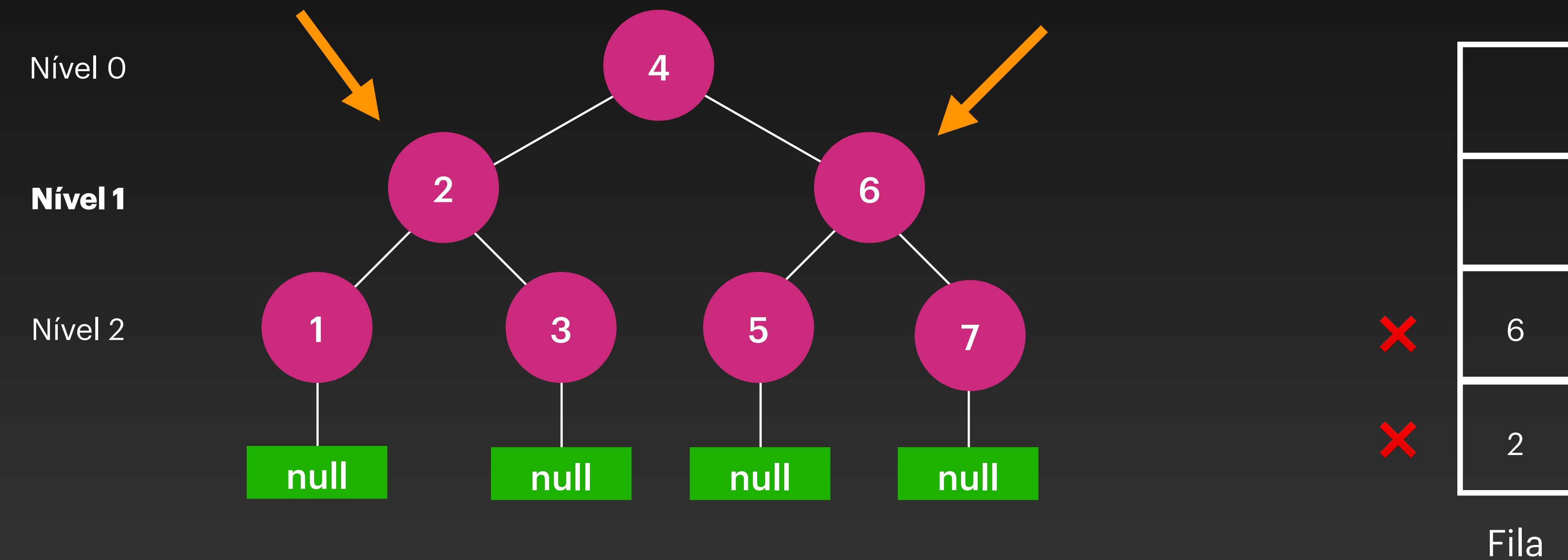
Busca em Largura - Exemplo



buscando o valor 3

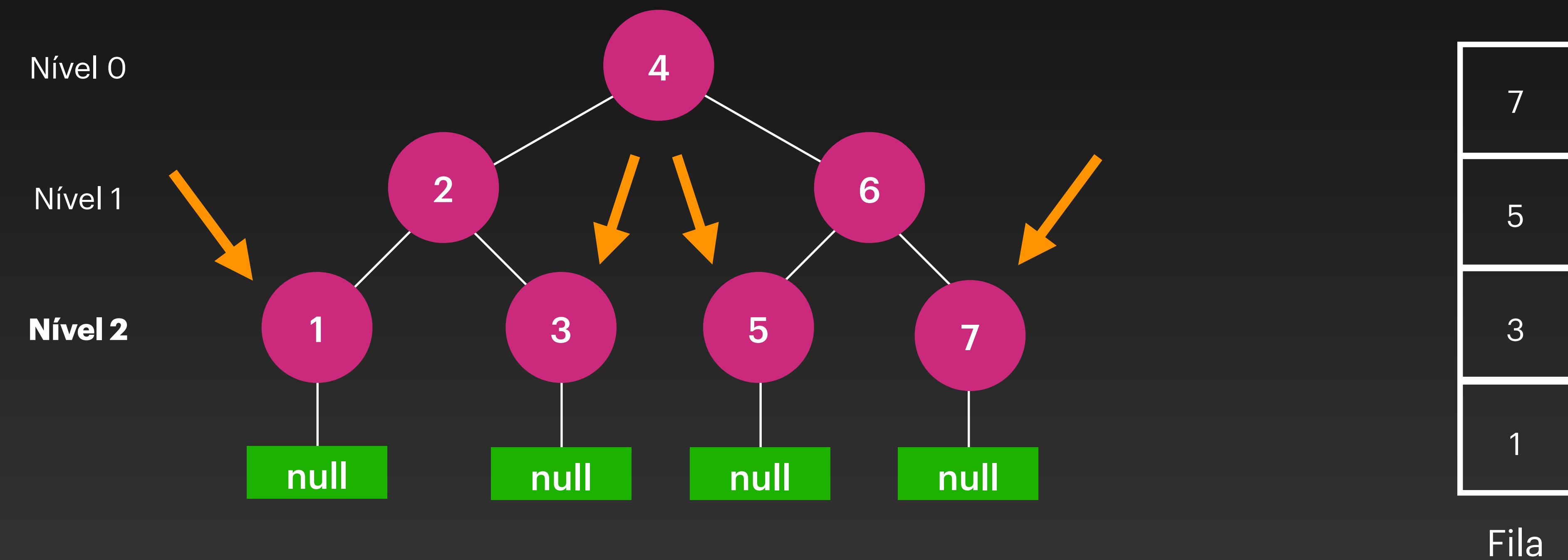
Manipulando uma Árvore

Busca em Largura - Exemplo



Manipulando uma Árvore

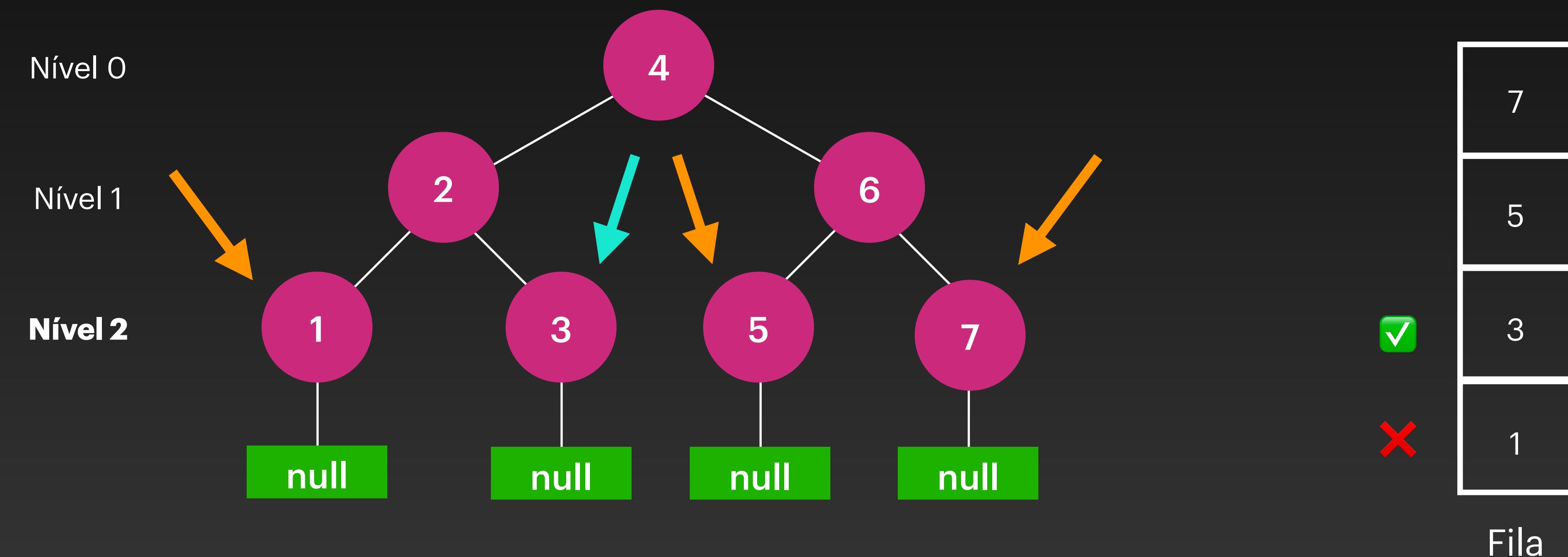
Busca em Largura - Exemplo



buscando o valor **3**

Manipulando uma Árvore

Busca em Largura - Exemplo



buscando o valor 3 ✓

QuadTree



QuadTree

Definição

- Uma quadtree é uma árvore onde cada **nó interno** possui exatamente **quatro filhos**.
- São comumente utilizadas para representar um espaço bidimensional (x,y), dividindo sua região recursivamente em quatro quadrantes.

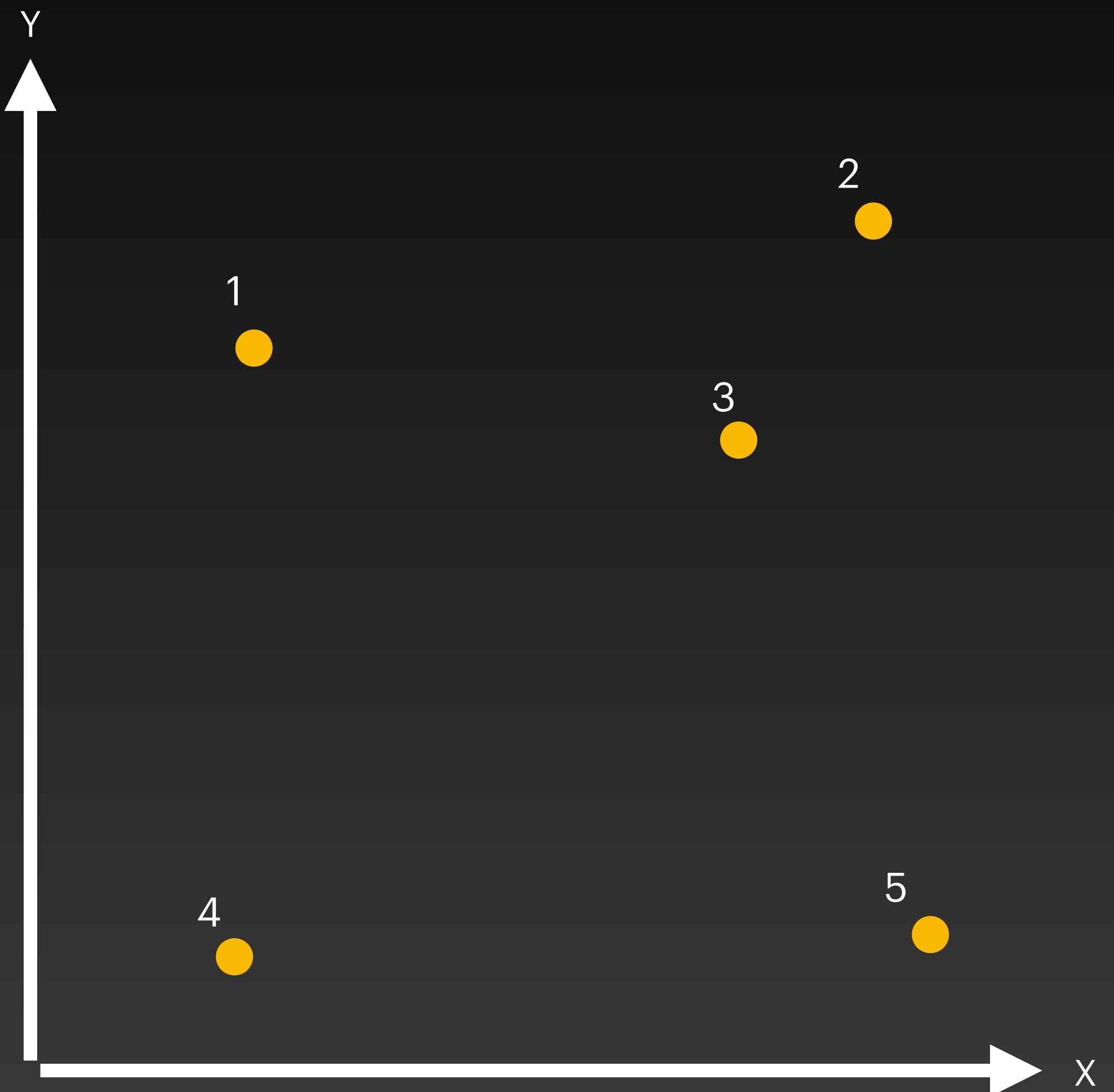
Point QuadTree

Definição

- Uma point quadtree é uma adaptação de uma árvore binária para poder representar pontos em um espaço bidimensional.
- Ela divide o espaço em quatro com base no ponto (x,y) dado no momento da inserção, e utiliza essa referência para adicionar os pontos subsequentes de acordo com a localização deles no plano.
- Pode se tornar extremamente eficiente em termos de busca, dependendo da forma na qual os pontos são inseridos.

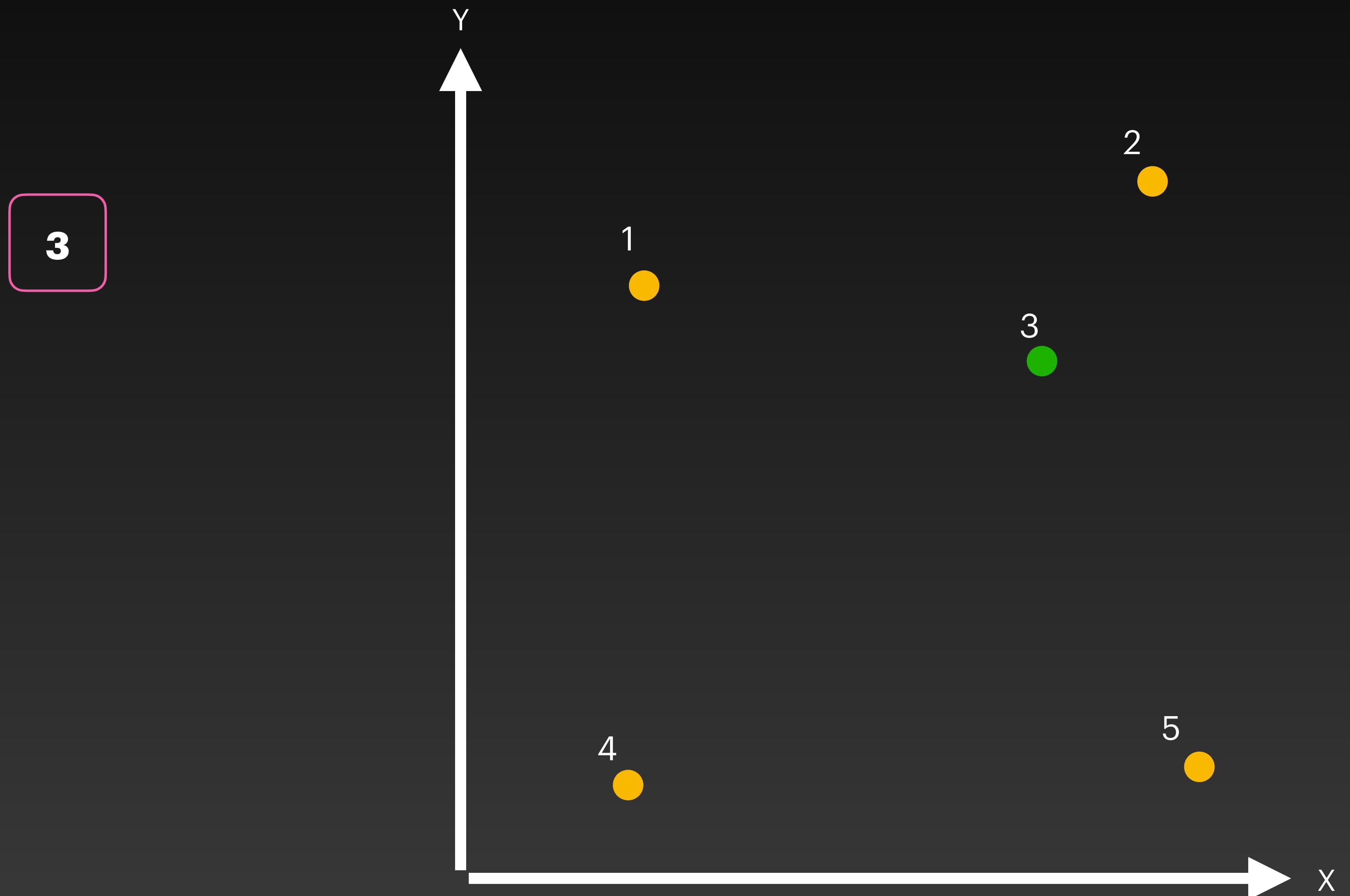
Point QuadTree

Exemplo



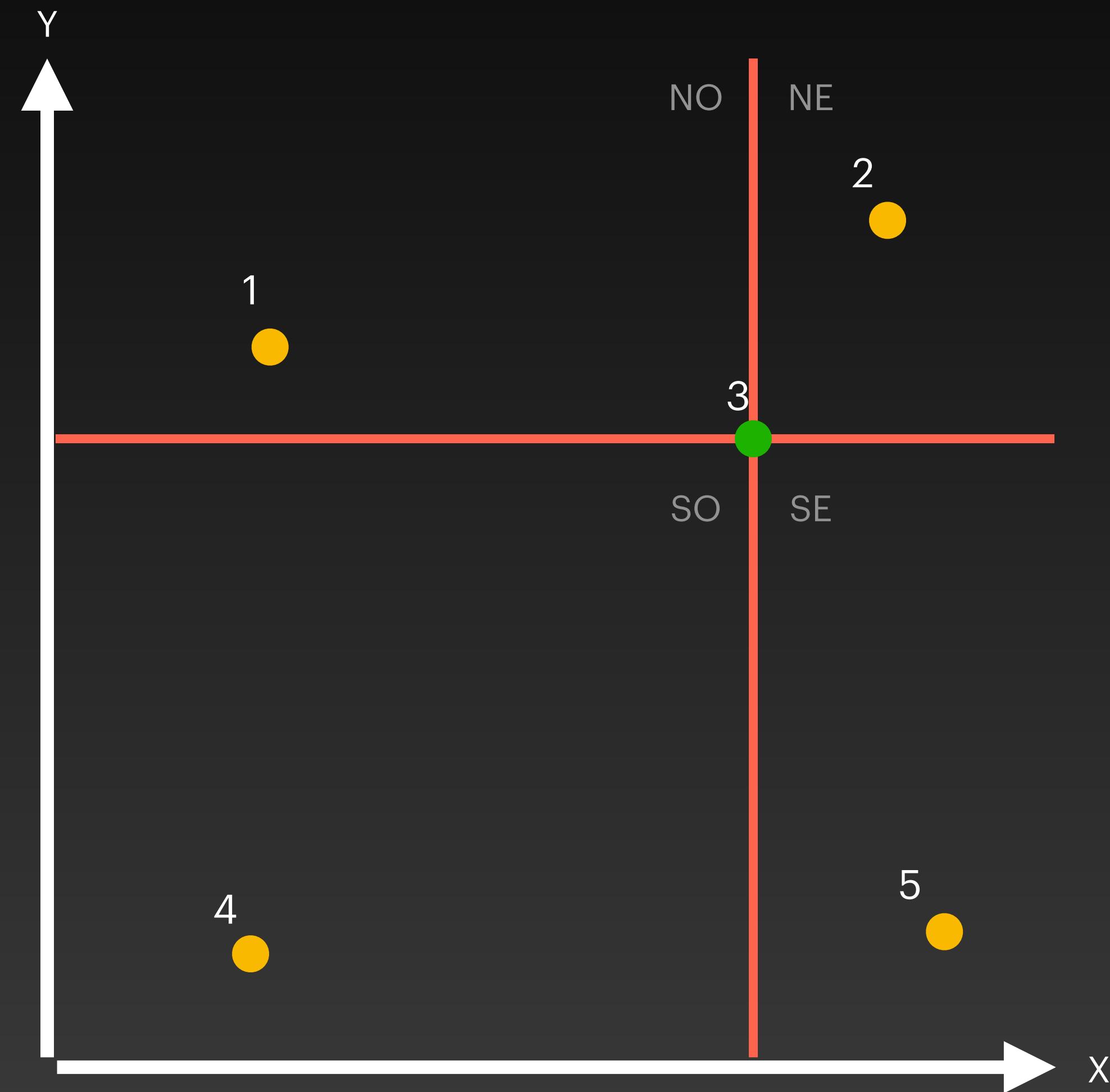
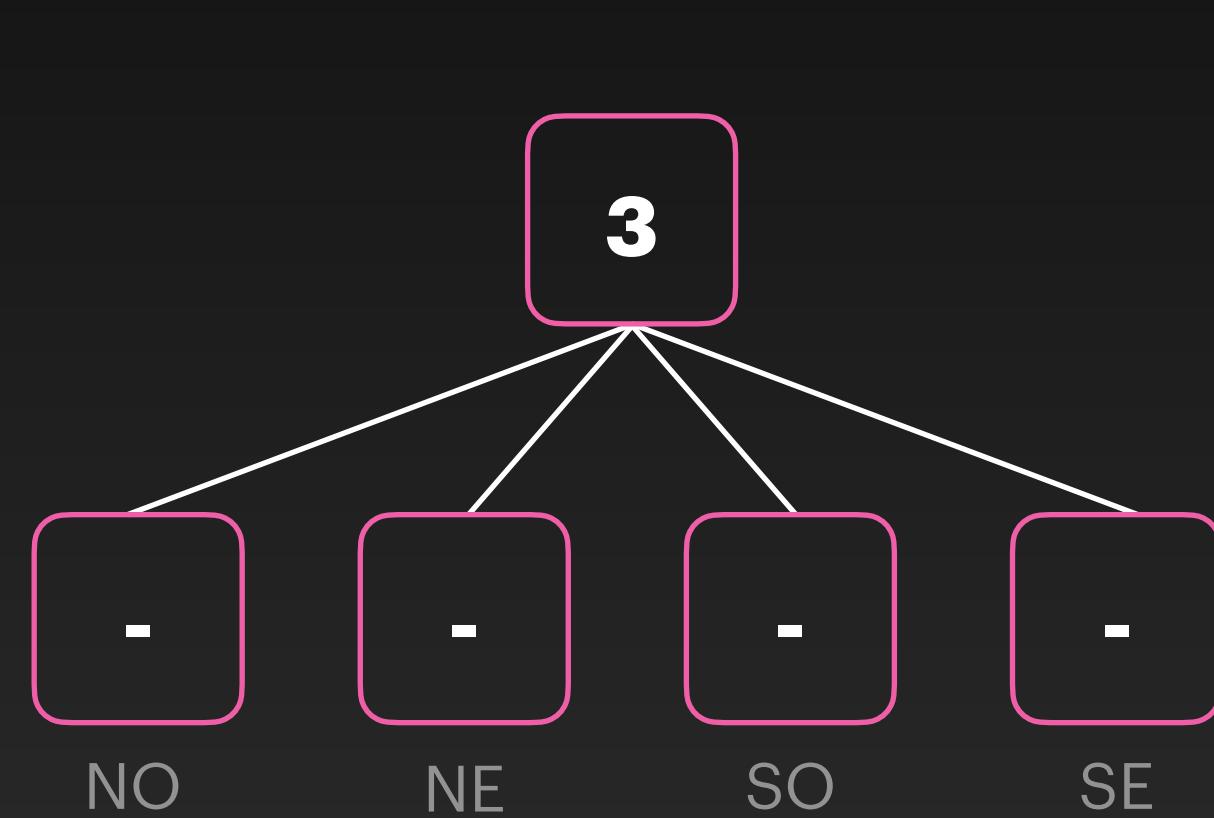
Point QuadTree

Exemplo



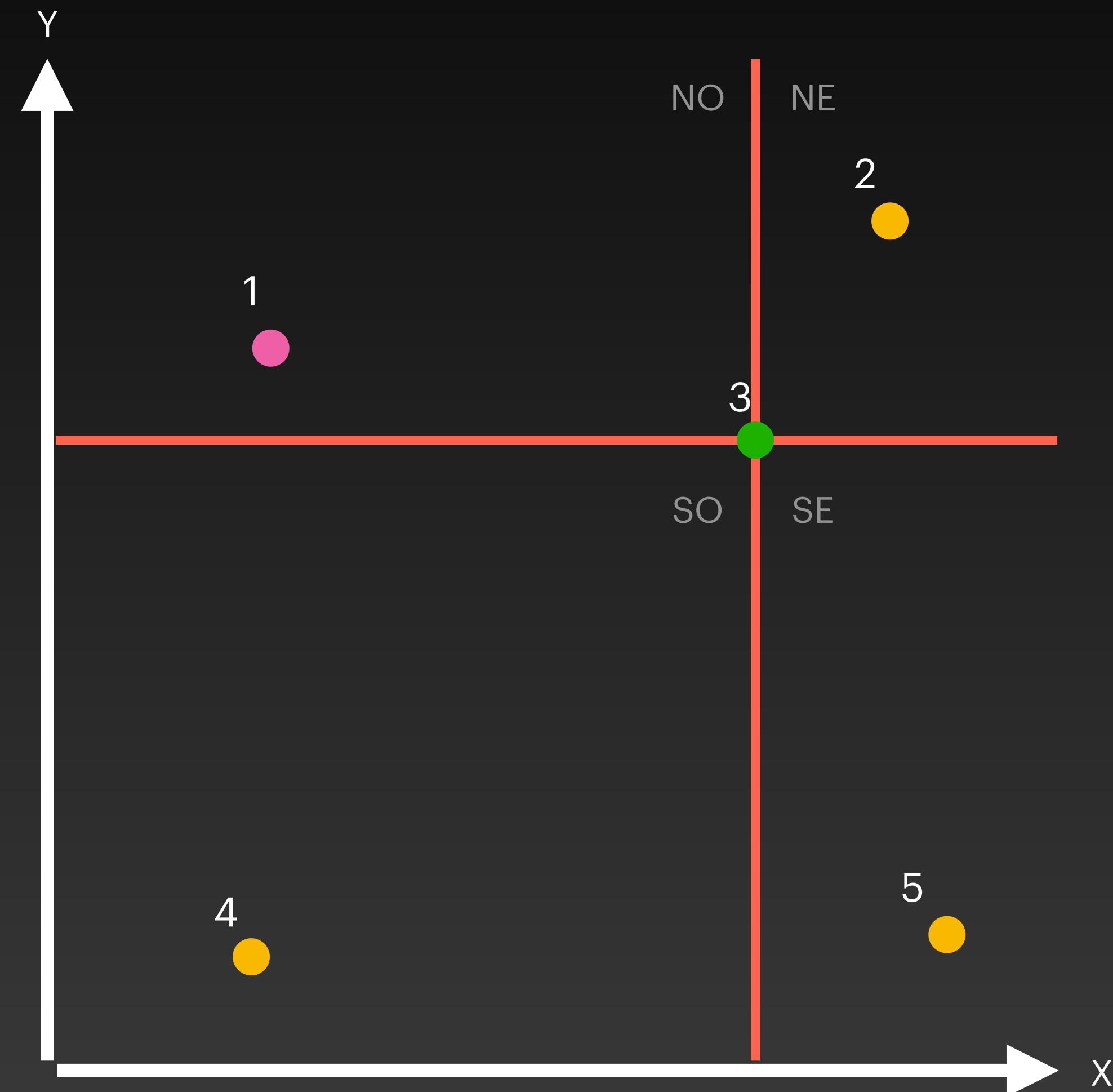
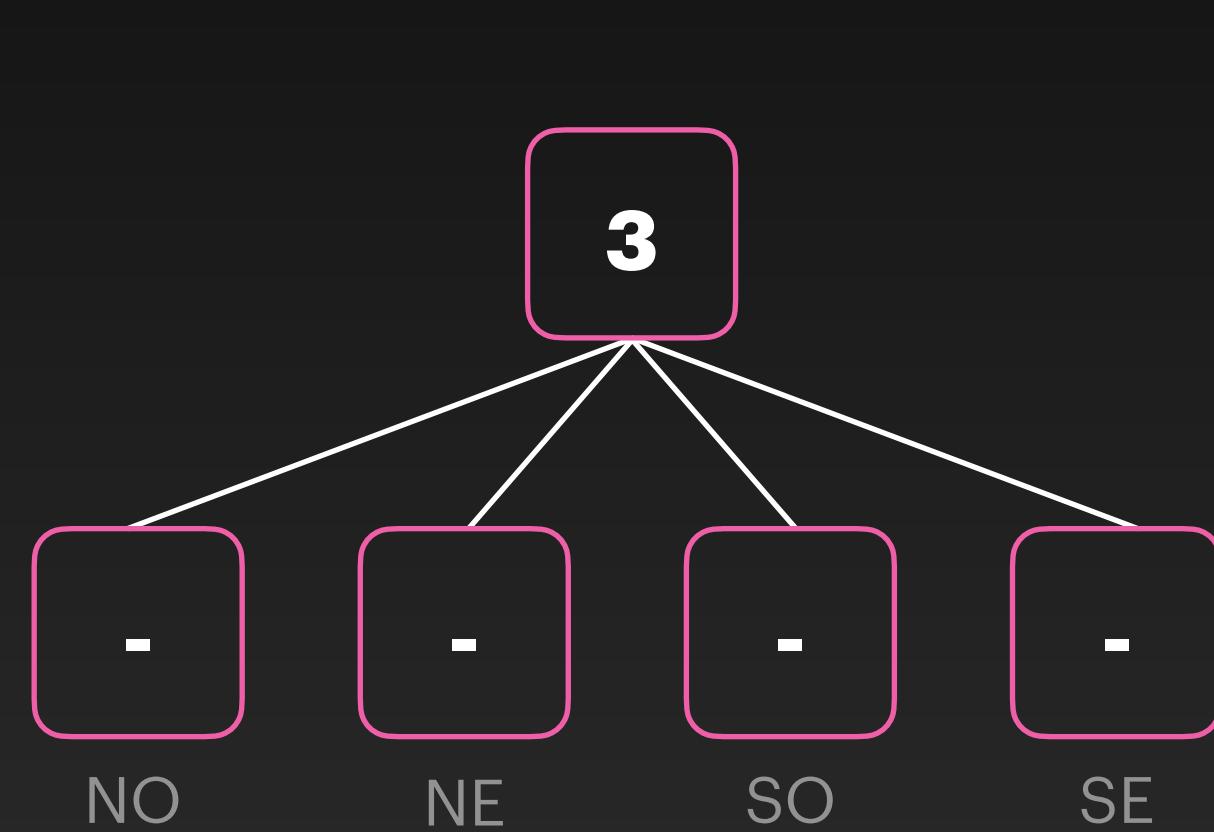
Point QuadTree

Exemplo



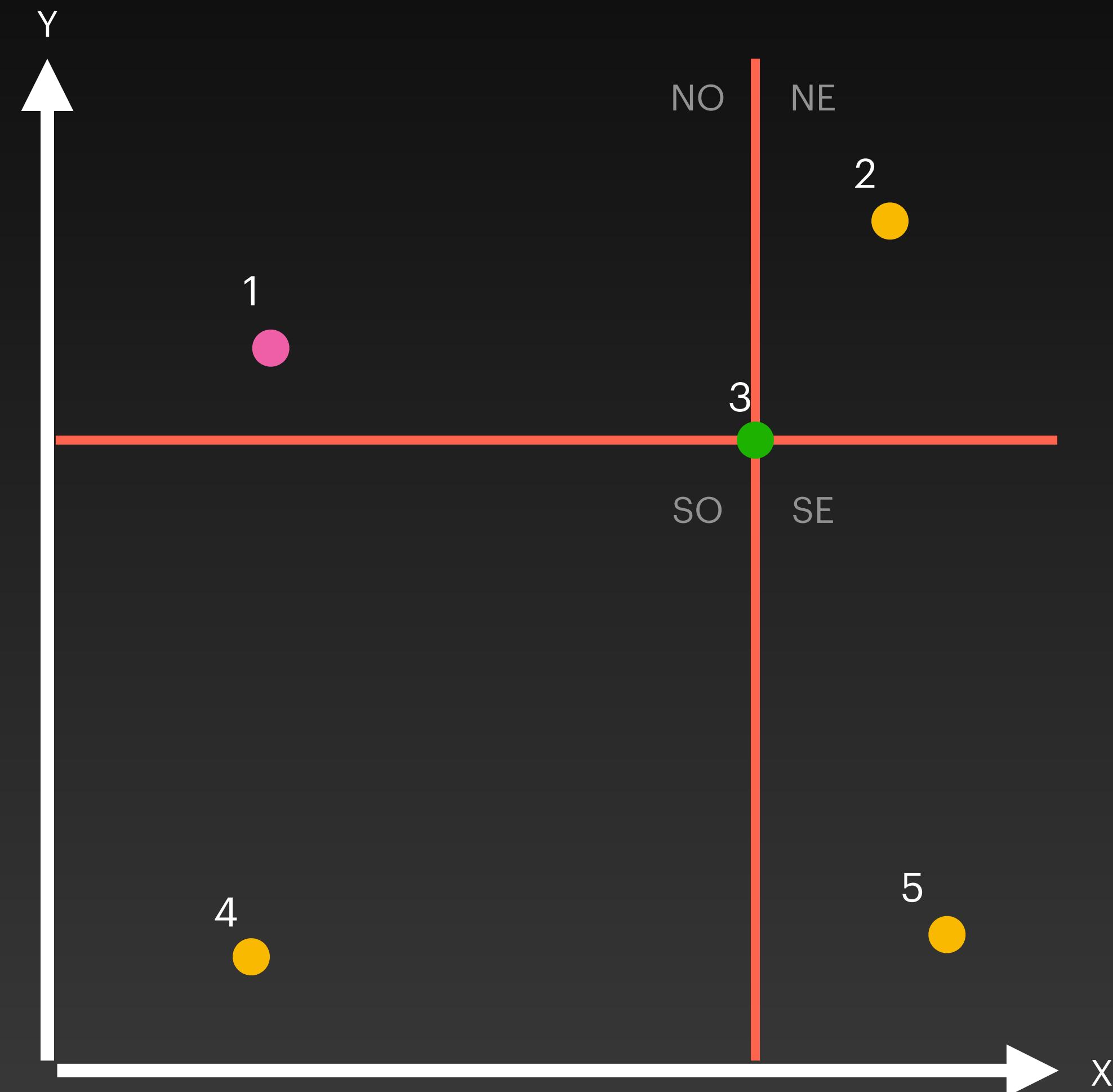
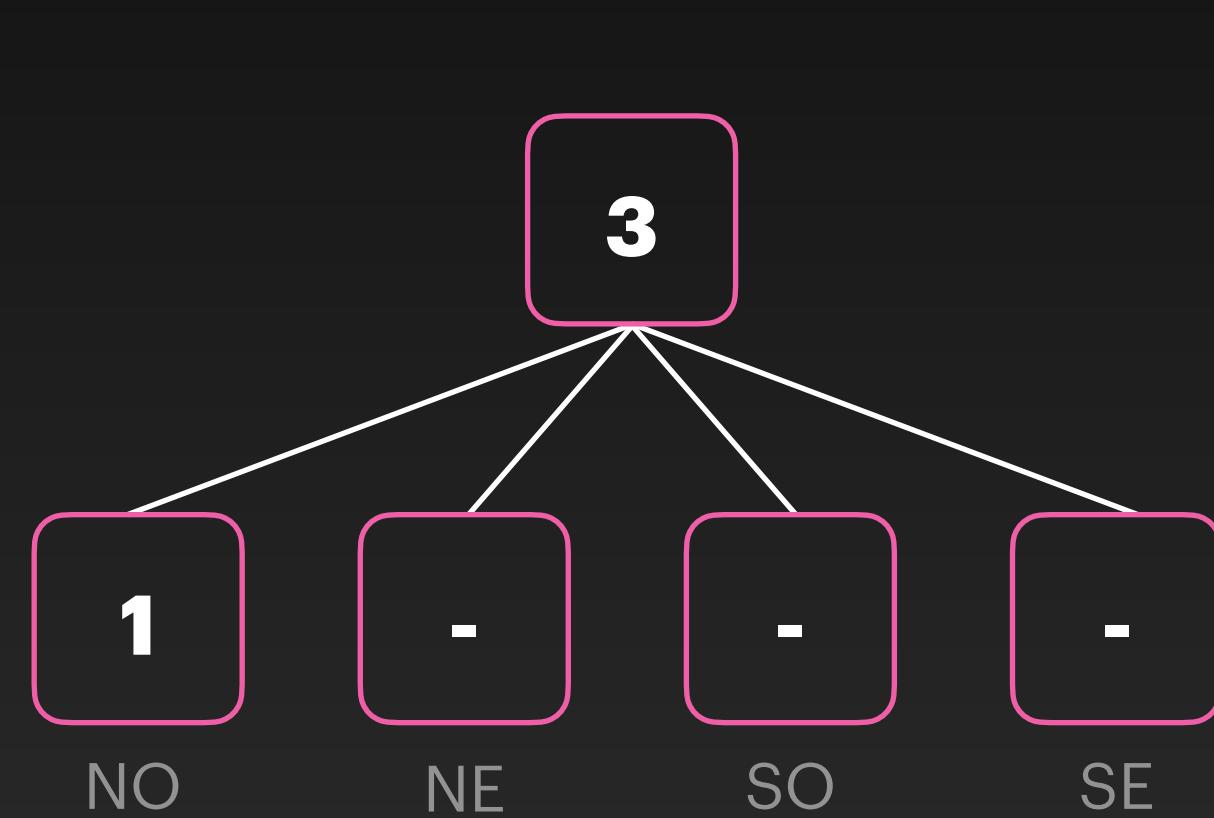
Point QuadTree

Exemplo



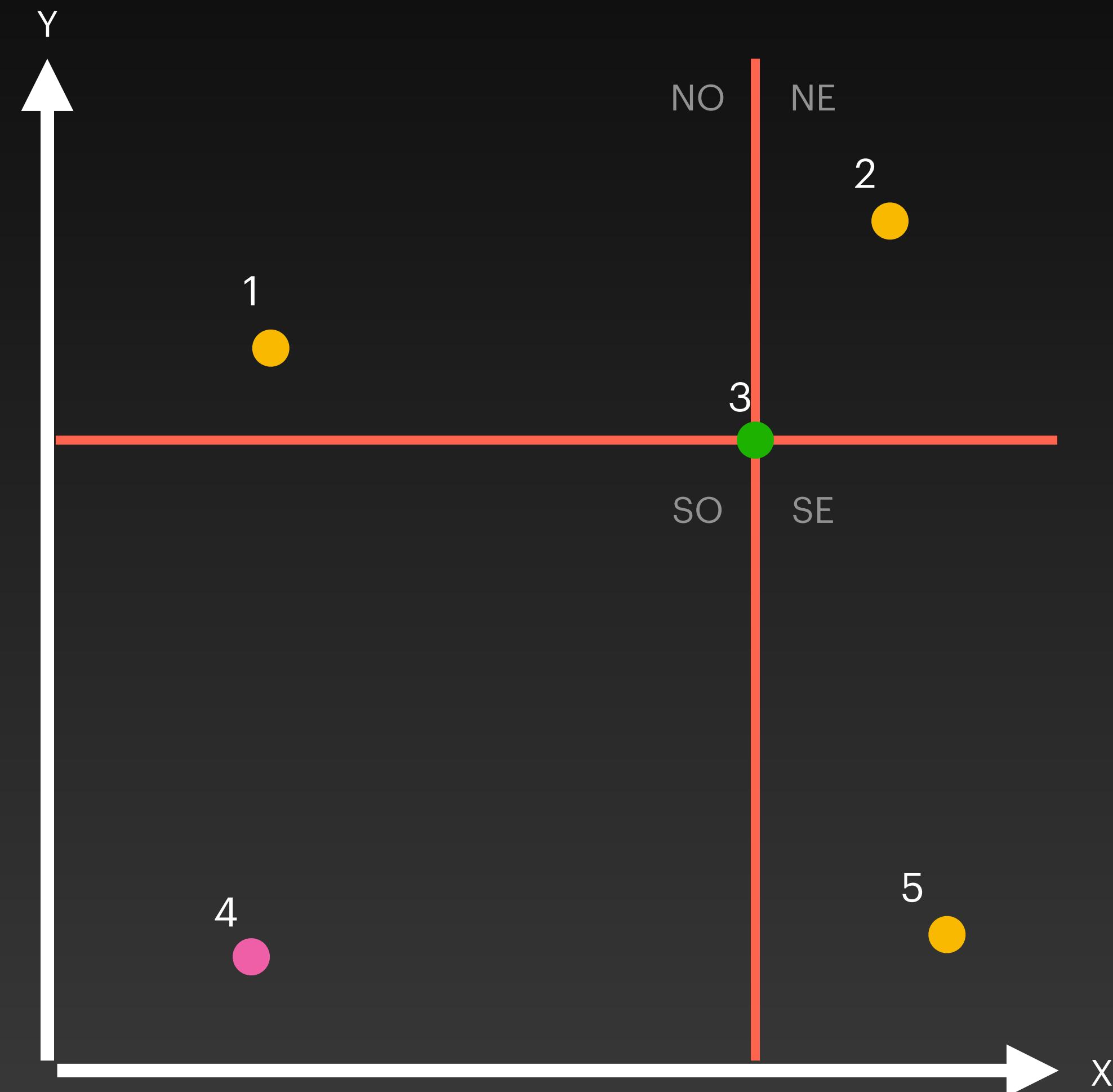
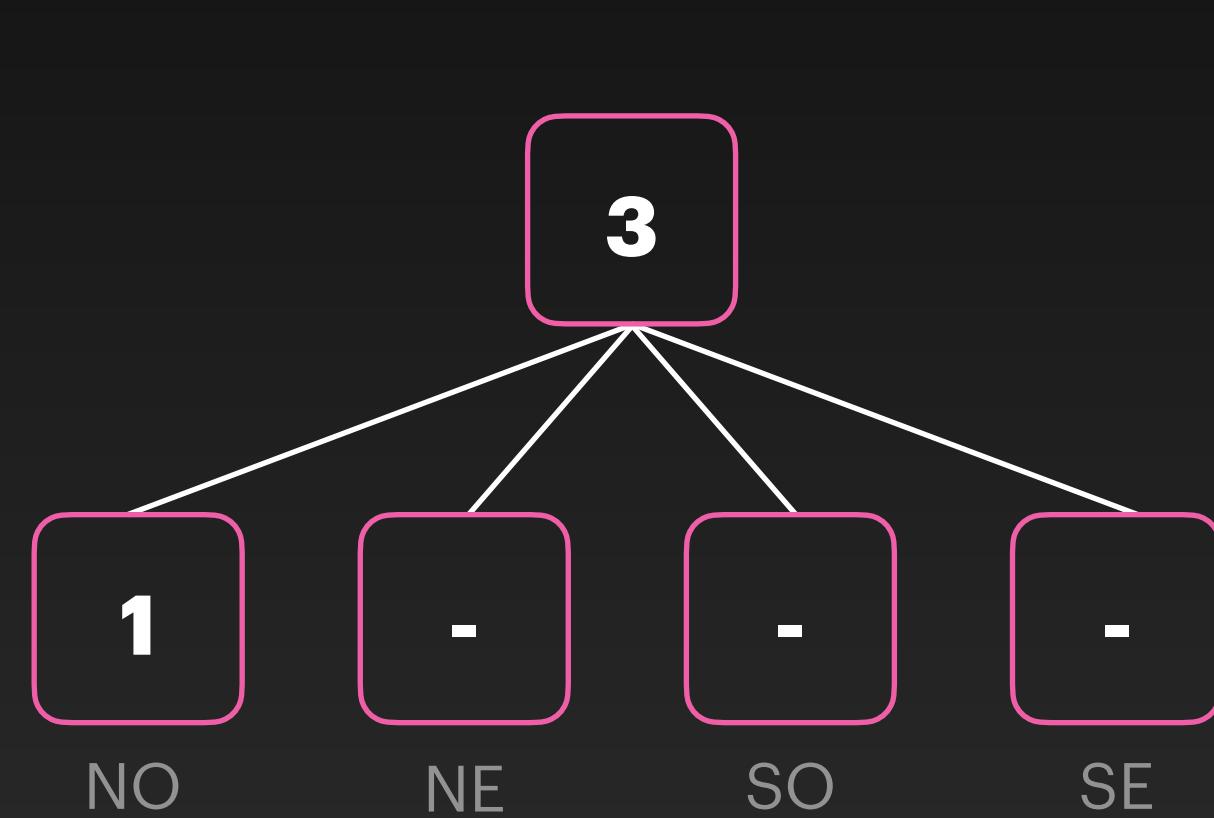
Point QuadTree

Exemplo



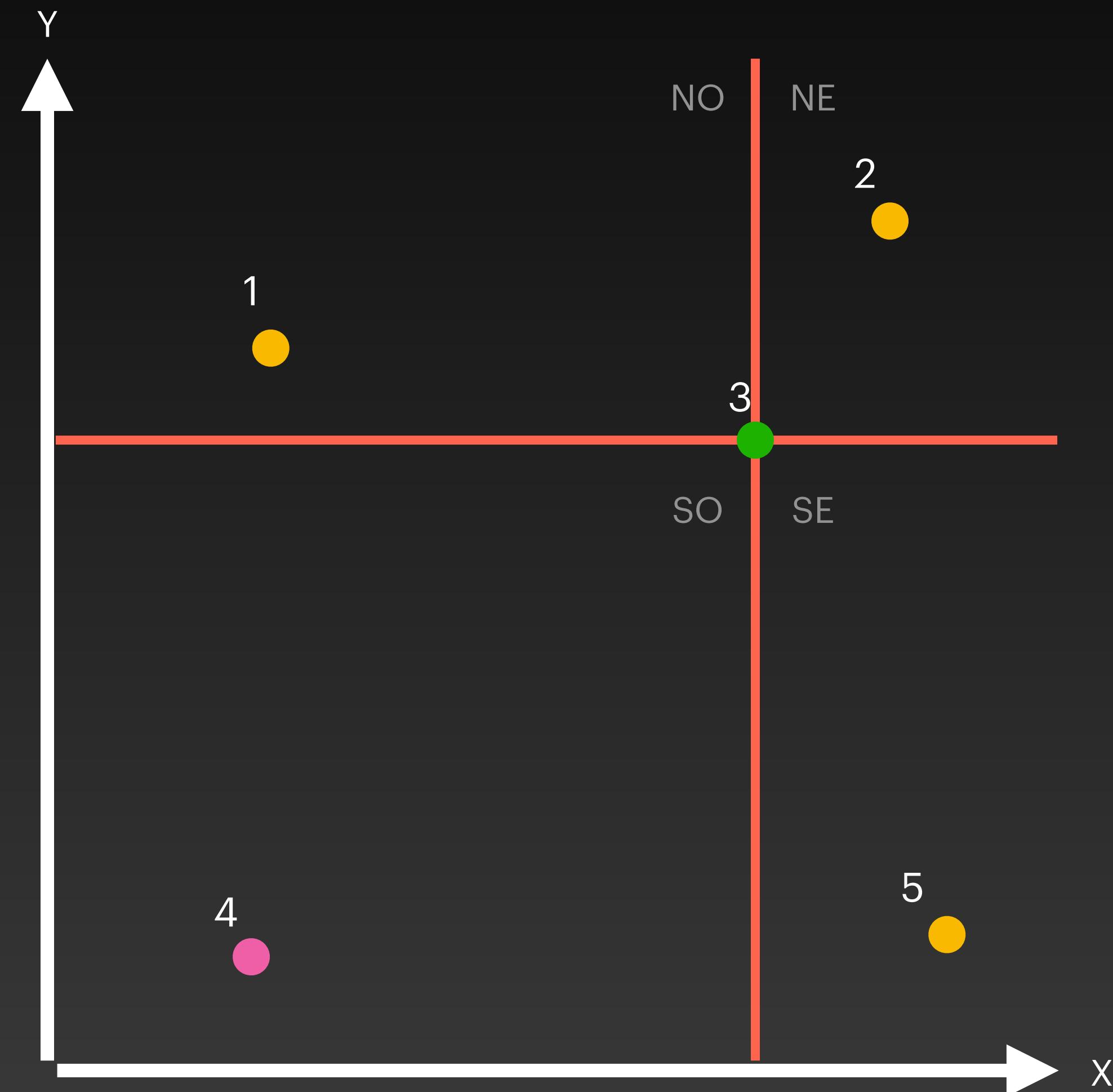
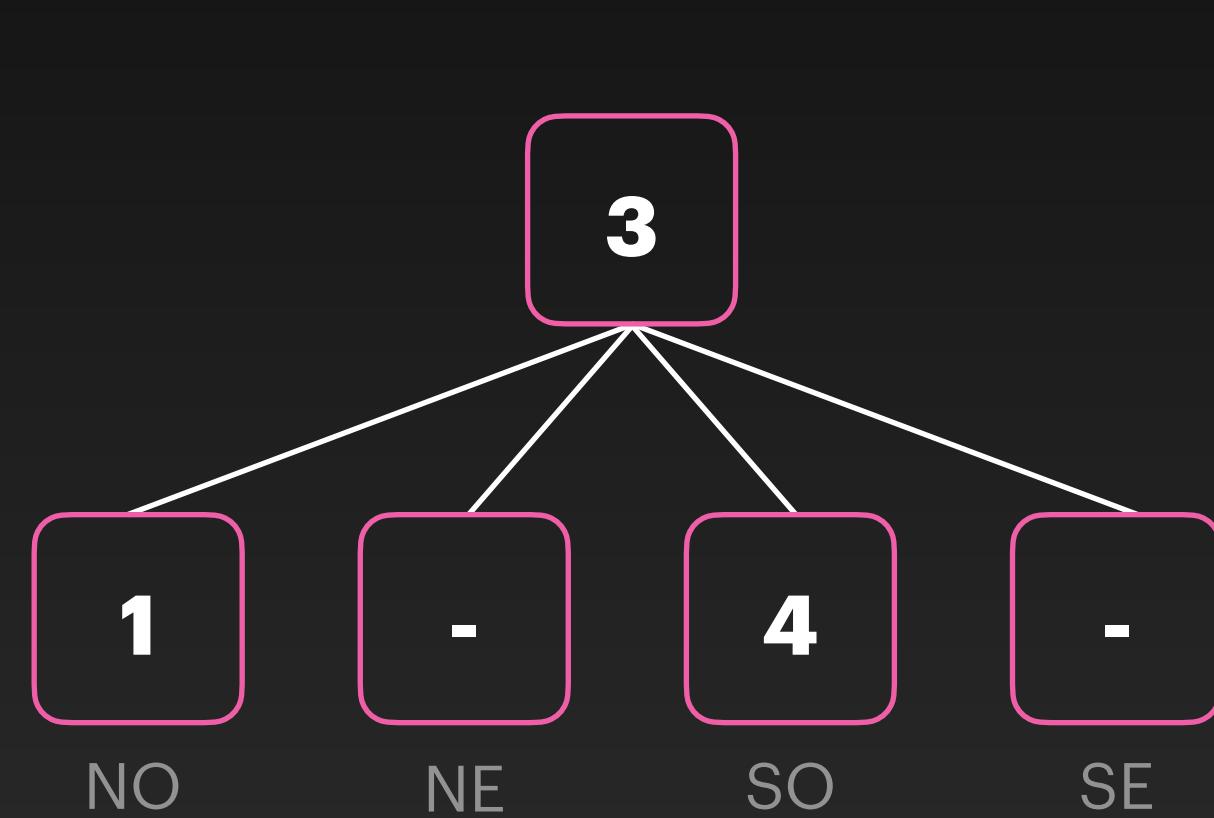
Point QuadTree

Exemplo



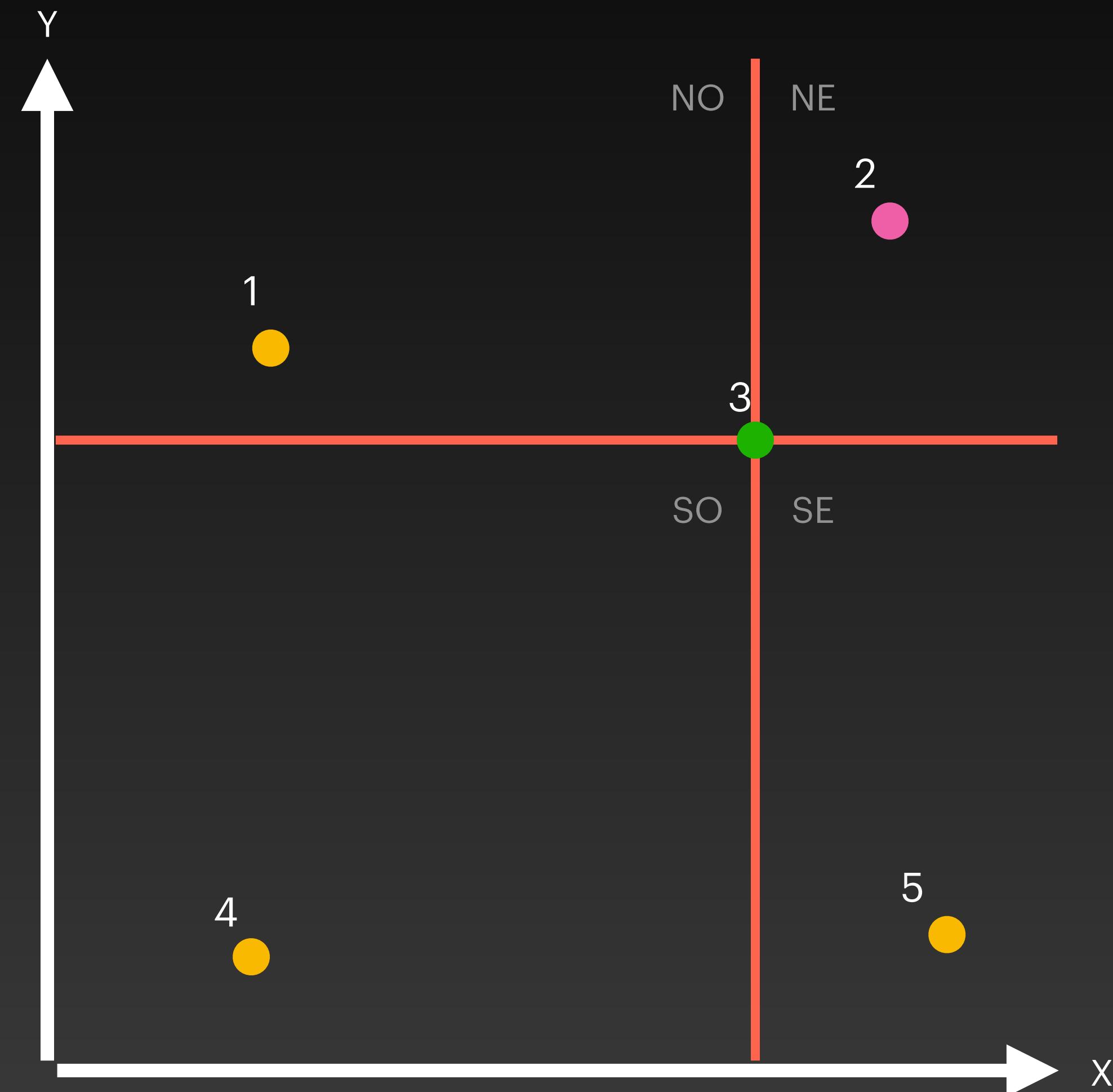
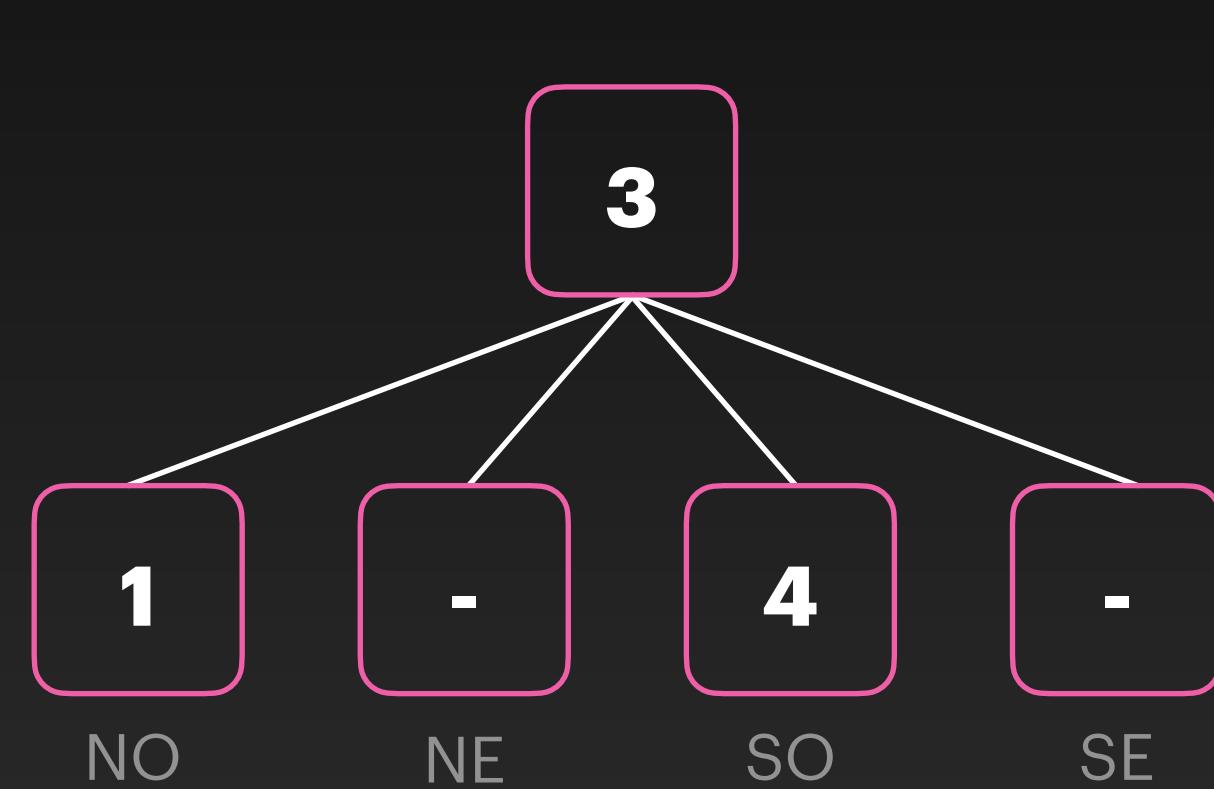
Point QuadTree

Exemplo



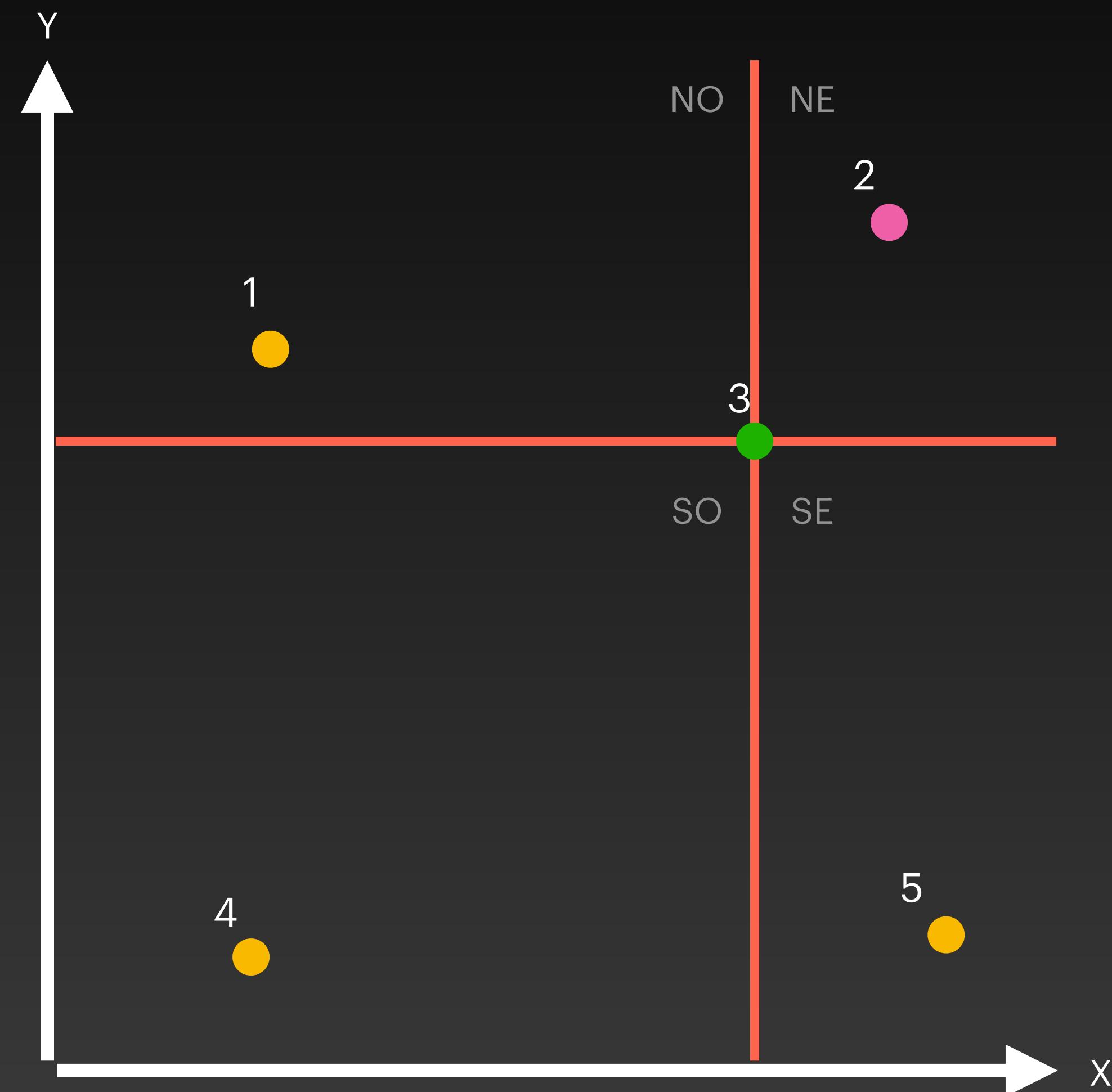
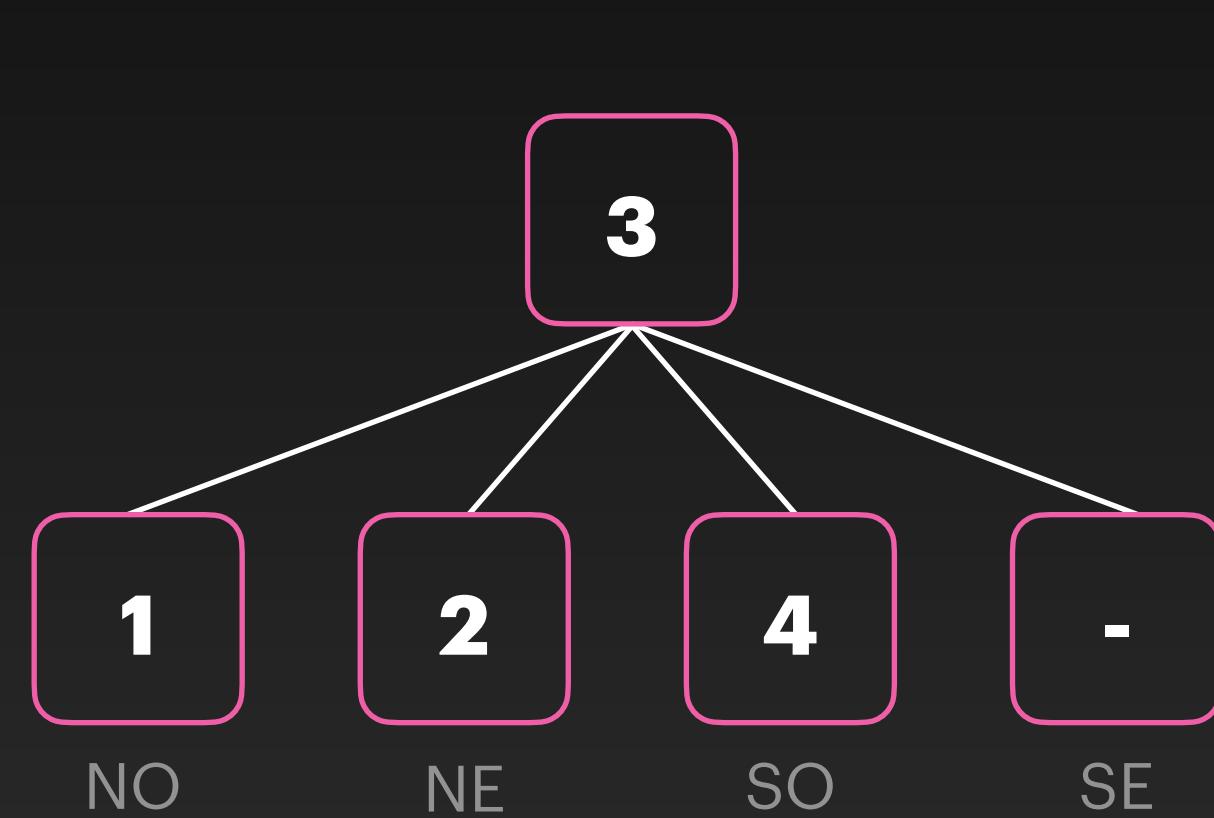
Point QuadTree

Exemplo



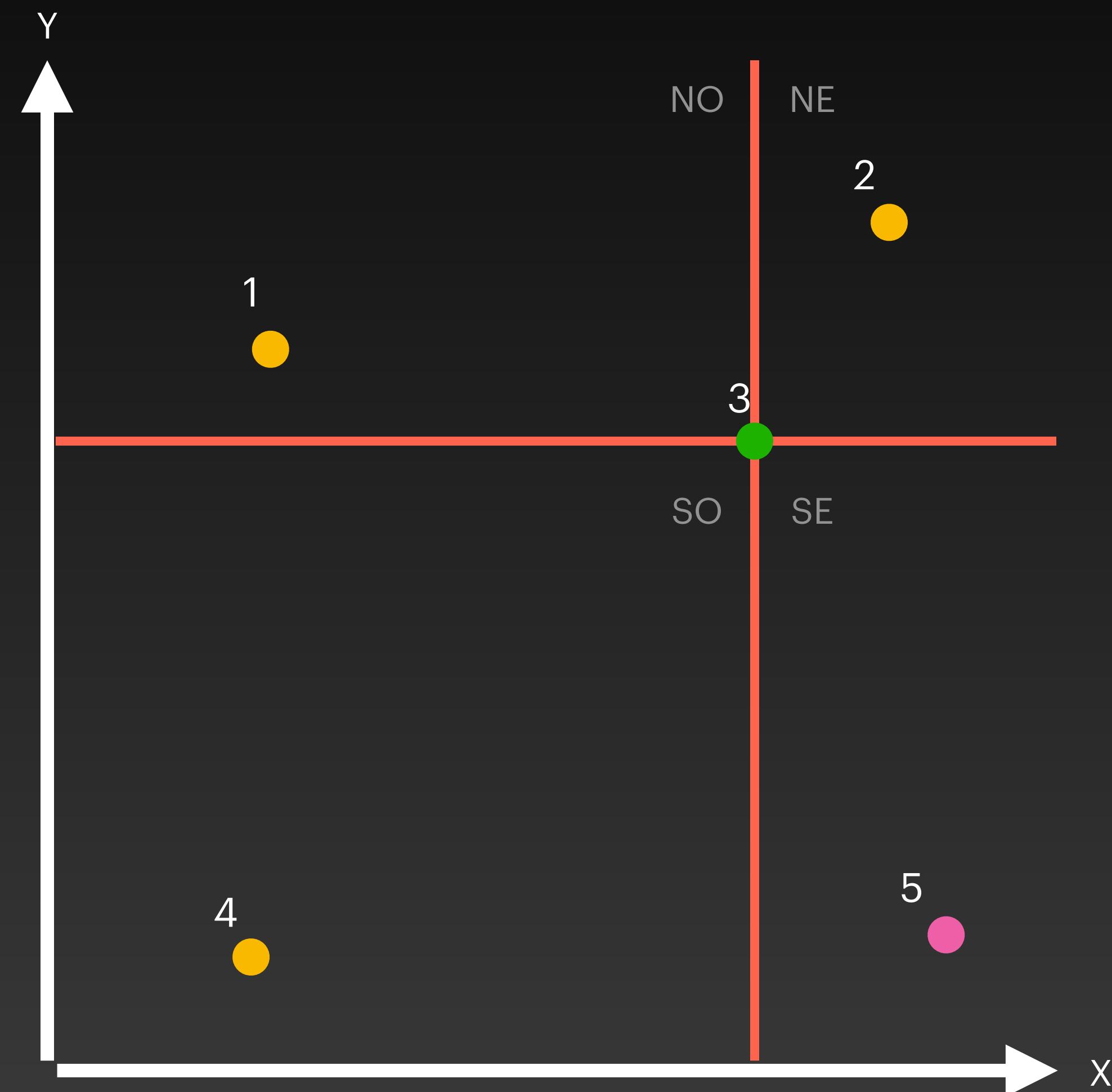
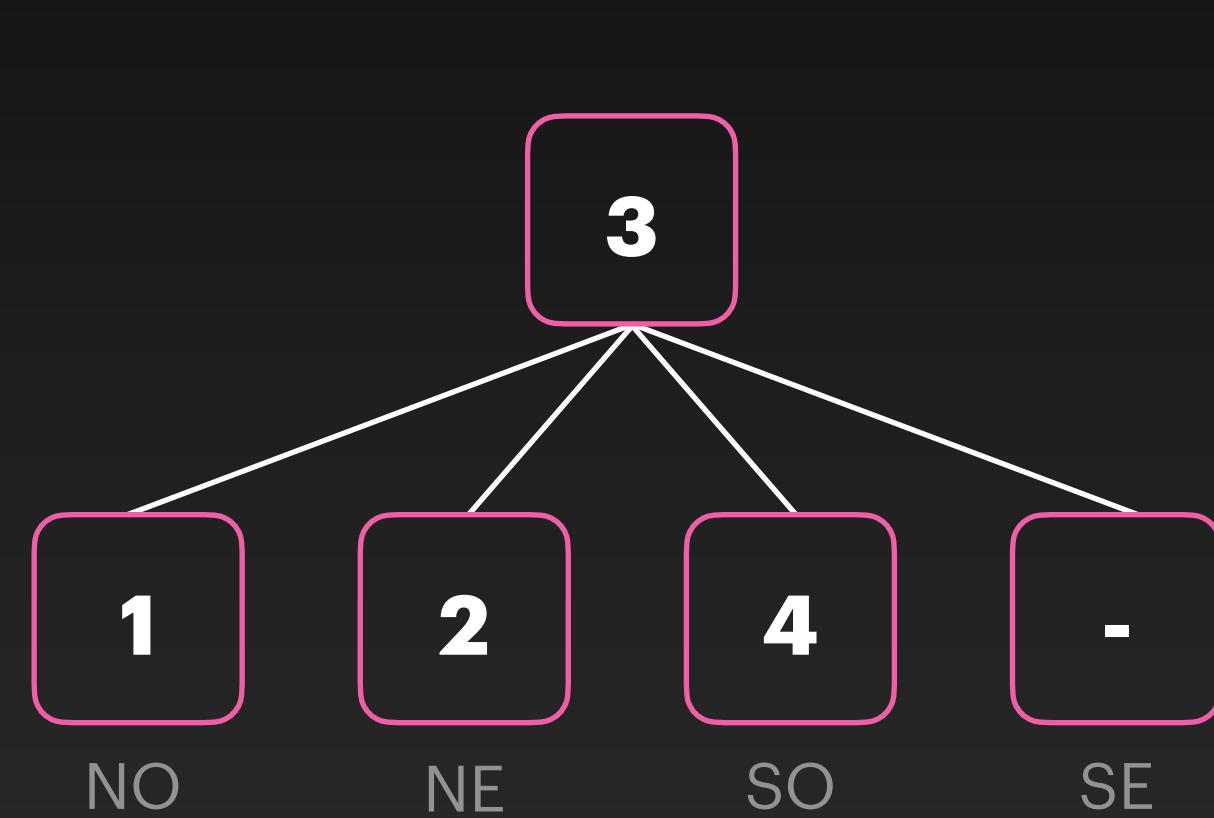
Point QuadTree

Exemplo



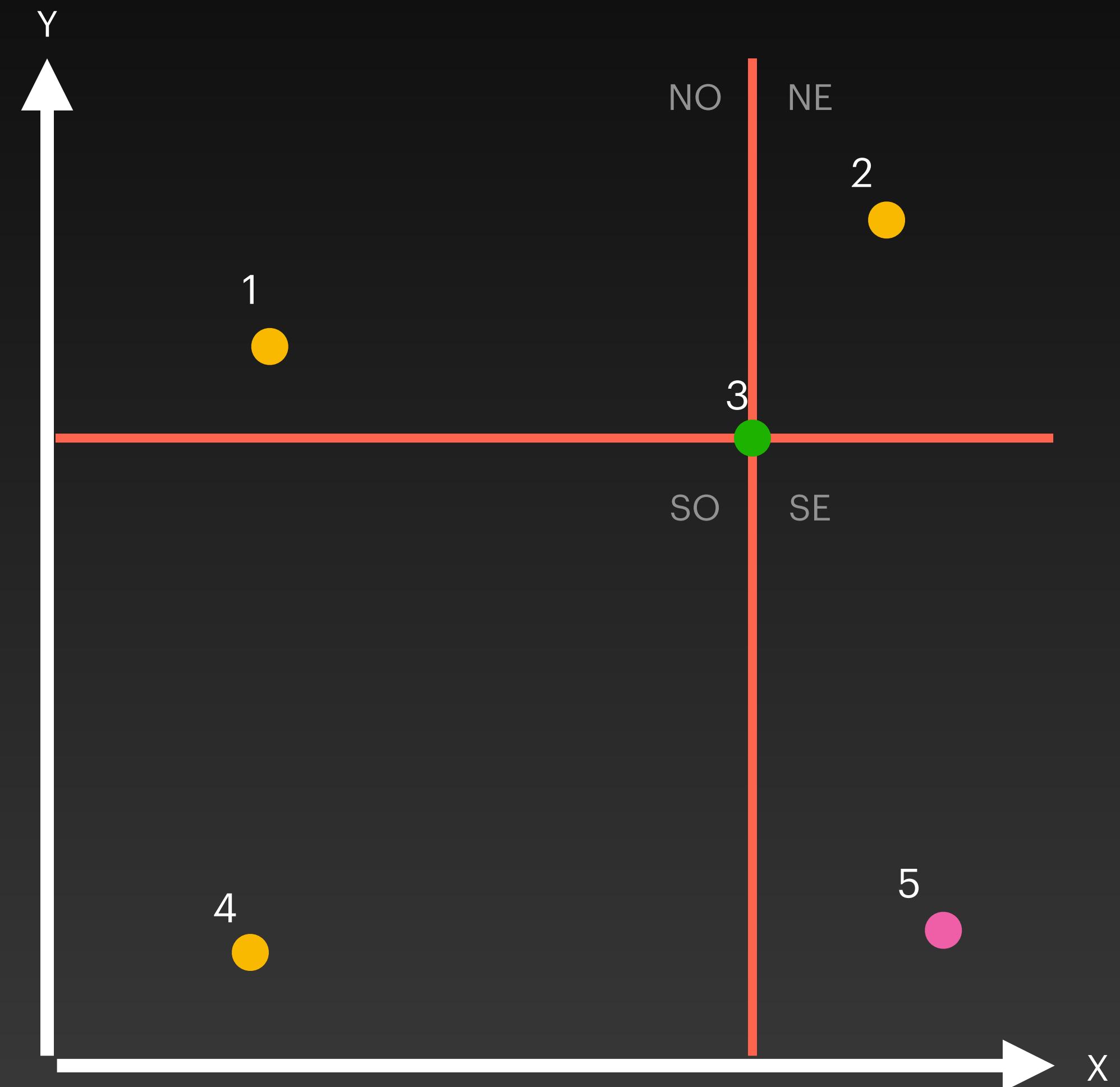
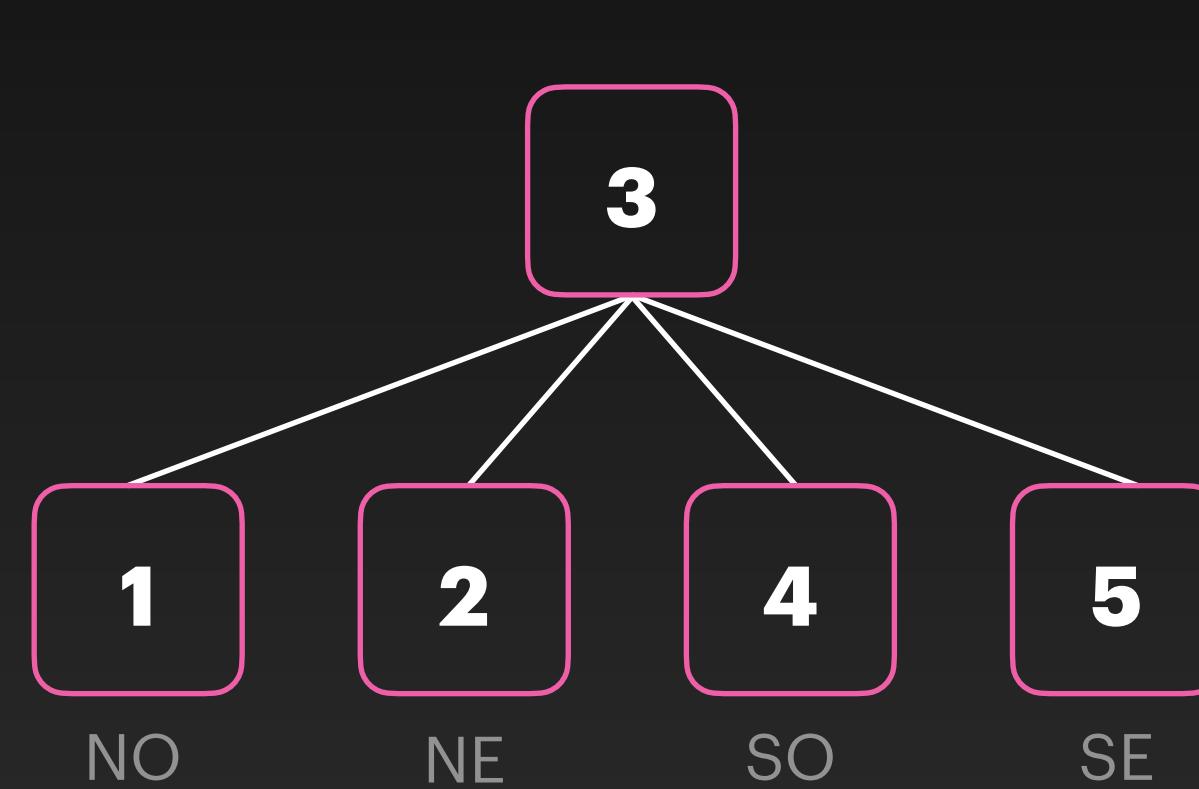
Point QuadTree

Exemplo



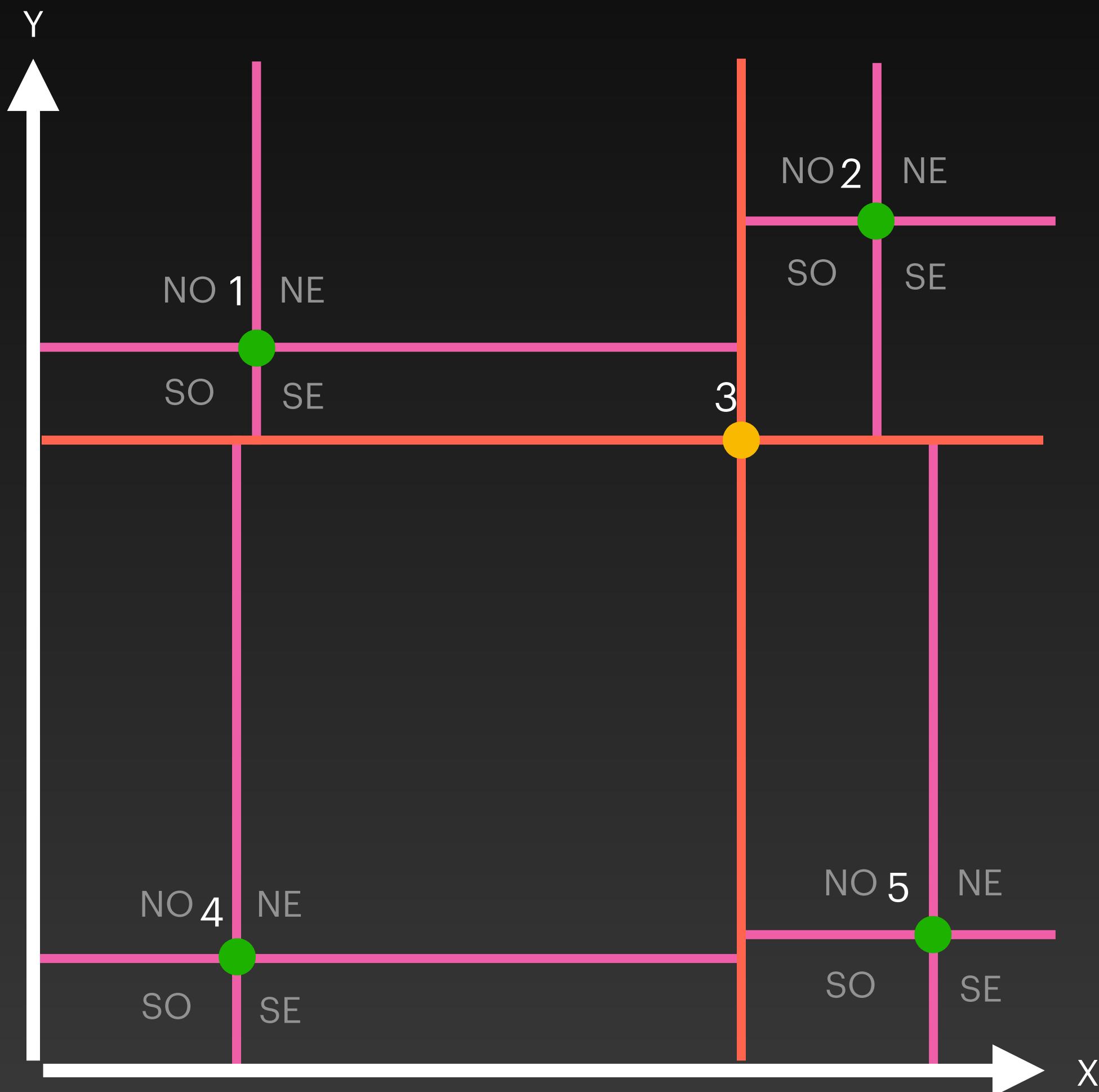
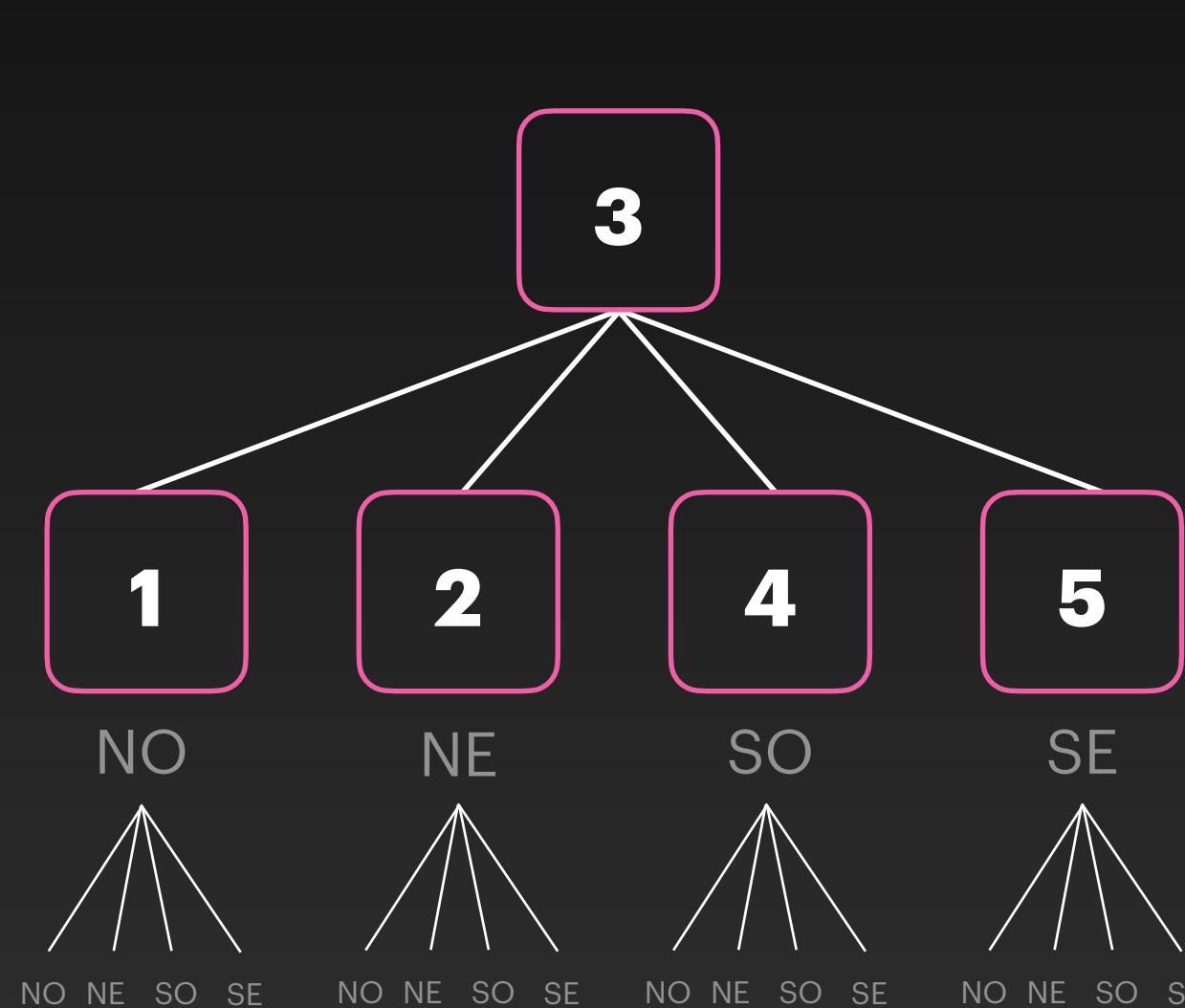
Point QuadTree

Exemplo



Point QuadTree

Exemplo



Point QuadTree

Inserção

- Em uma quadtree, a ordem de inserção dos elementos é que garante o balanceamento da árvore.
- A dica é sempre começar com o ponto mais no centro do plano, de forma a sempre dividir igualmente o espaço.

Point QuadTree

Inserção



```
InsereQt(qt, ponto, pInfo)
começo
    Quadrante quadrante;
    No pai;

    se (qt == null) então
        qt = CriaNo();
        qt.ponto = ponto;
        qt.info = pInfo;
    senão
        No qtAux = qt;
        enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto)))
            pai = qtAux;
            quadrante = GetQuadrante(ponto, qtAux.ponto);
            qtAux = GetFilho(qtAux, quadrante);
        fim-enquanto
        se qtAux == null então
            filho = CriaNo();
            filho.ponto = ponto;
            filho.info = pInfo;
            SetFilho(pai, quadrante);
        fim-se
    fim-se
fim
```

Point QuadTree

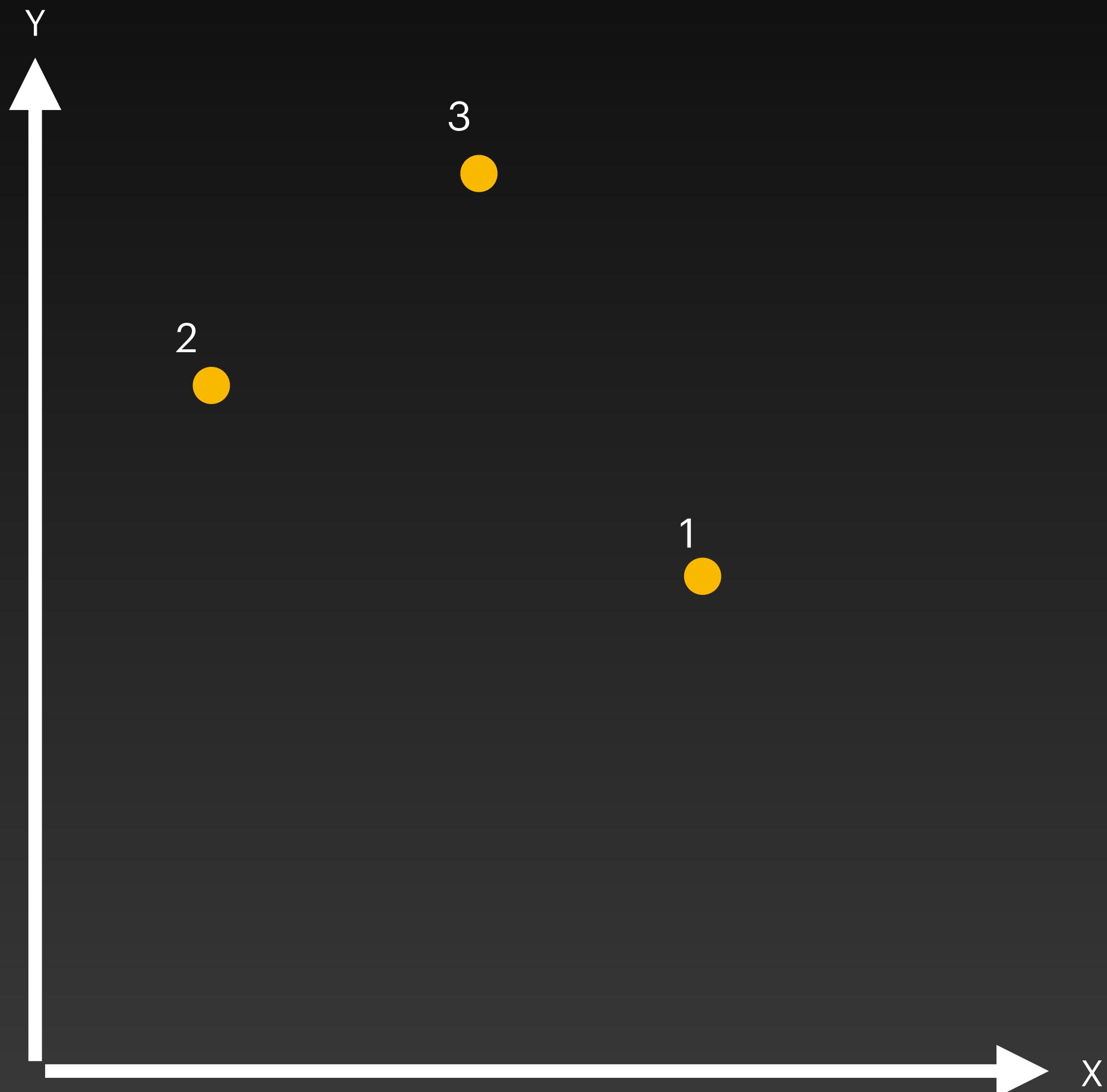
Inserção



```
GetQuadrante(ponto, pontoRaiz){  
    se ponto.x < pontoRaiz.x então  
        se ponto.y < pontoRaiz.y então  
            retorne SW  
        senão  
            retorne NW  
        fim-se  
    senão  
        se ponto.y < pontoRaiz.y então  
            retorne NW  
        senão  
            retorne NE  
        fim-se  
    }  
}
```

Point QuadTree

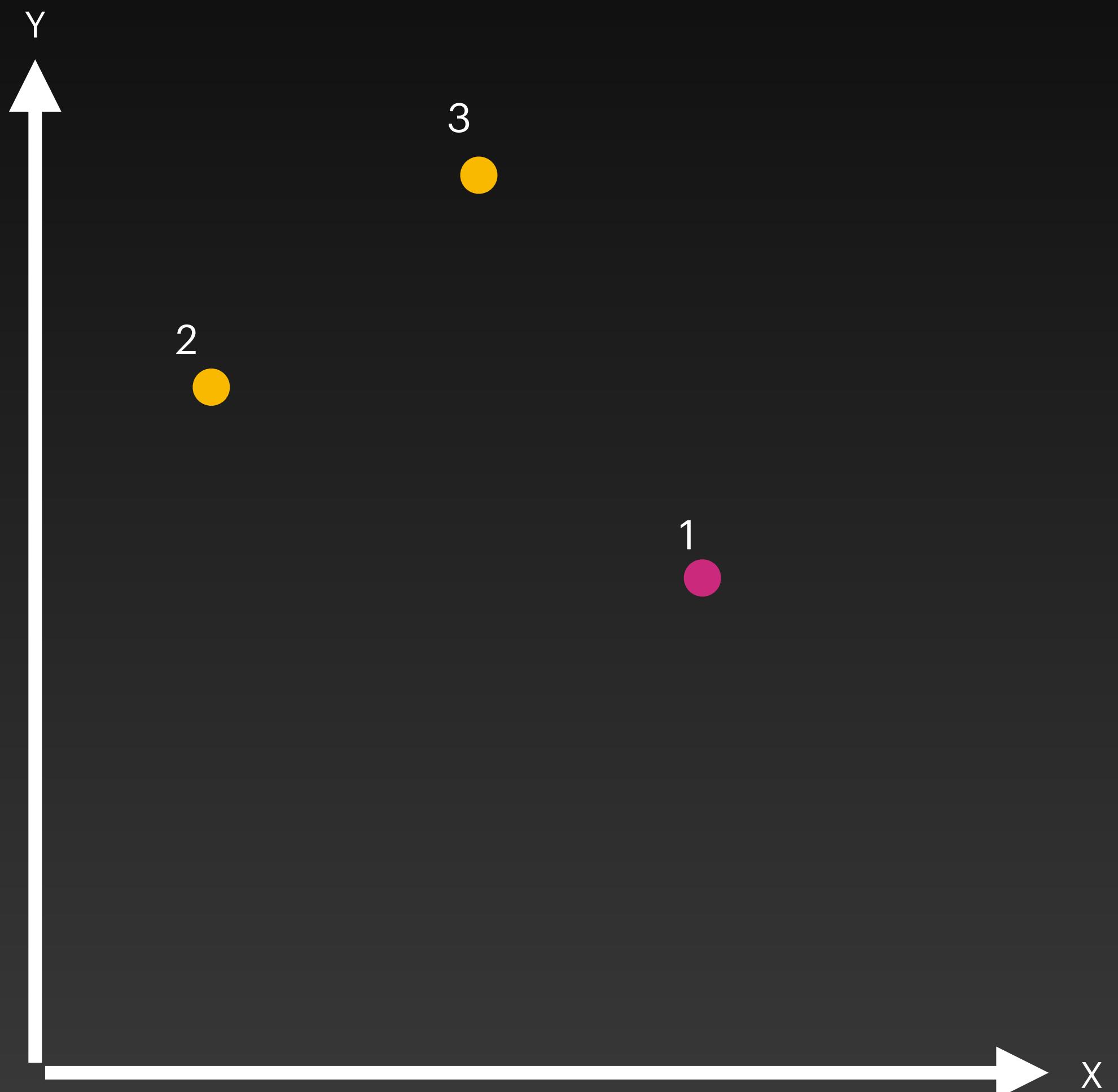
Inserção



```
var qt  
      ↓  
NULL
```

Point QuadTree

Inserção



```
InsereQt(qt, ponto, pInfo)
começo
    Quadrante quadrante;
    No pai;
    se (qt == null) então
        qt = CriaNo();
        qt.ponto = ponto;
        qt.info = pInfo;
```

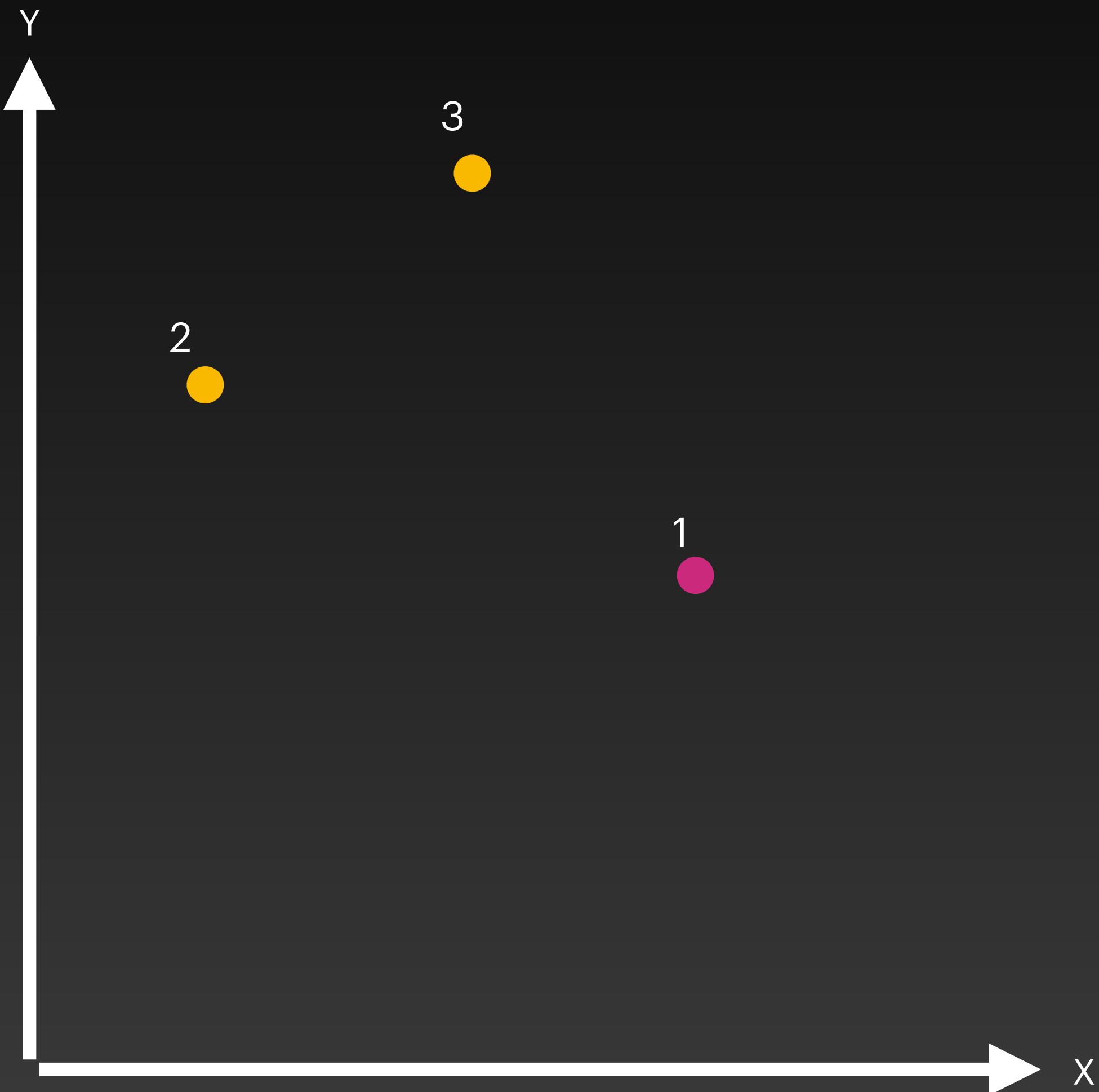
var qt



NULL

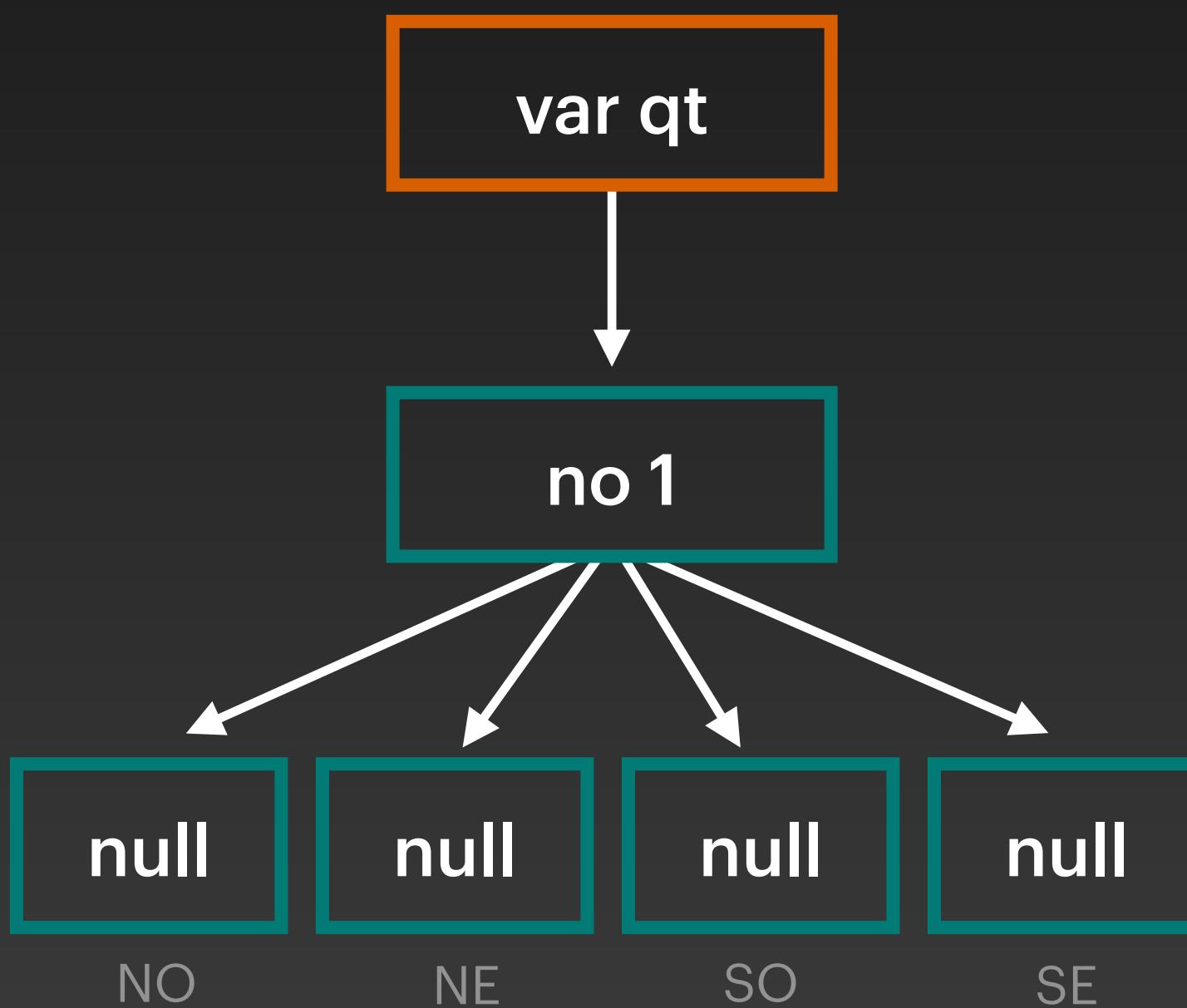
Point QuadTree

Inserção



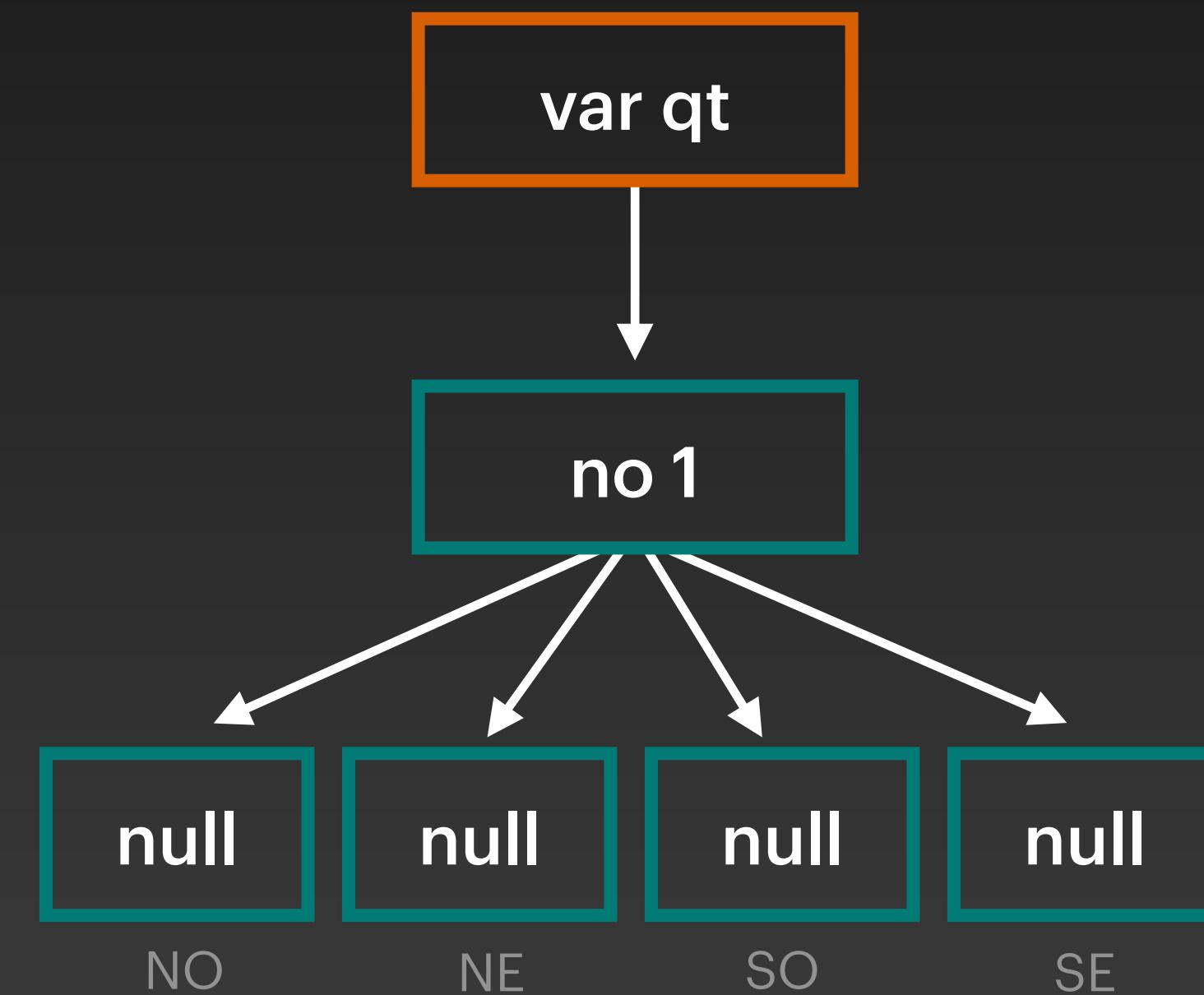
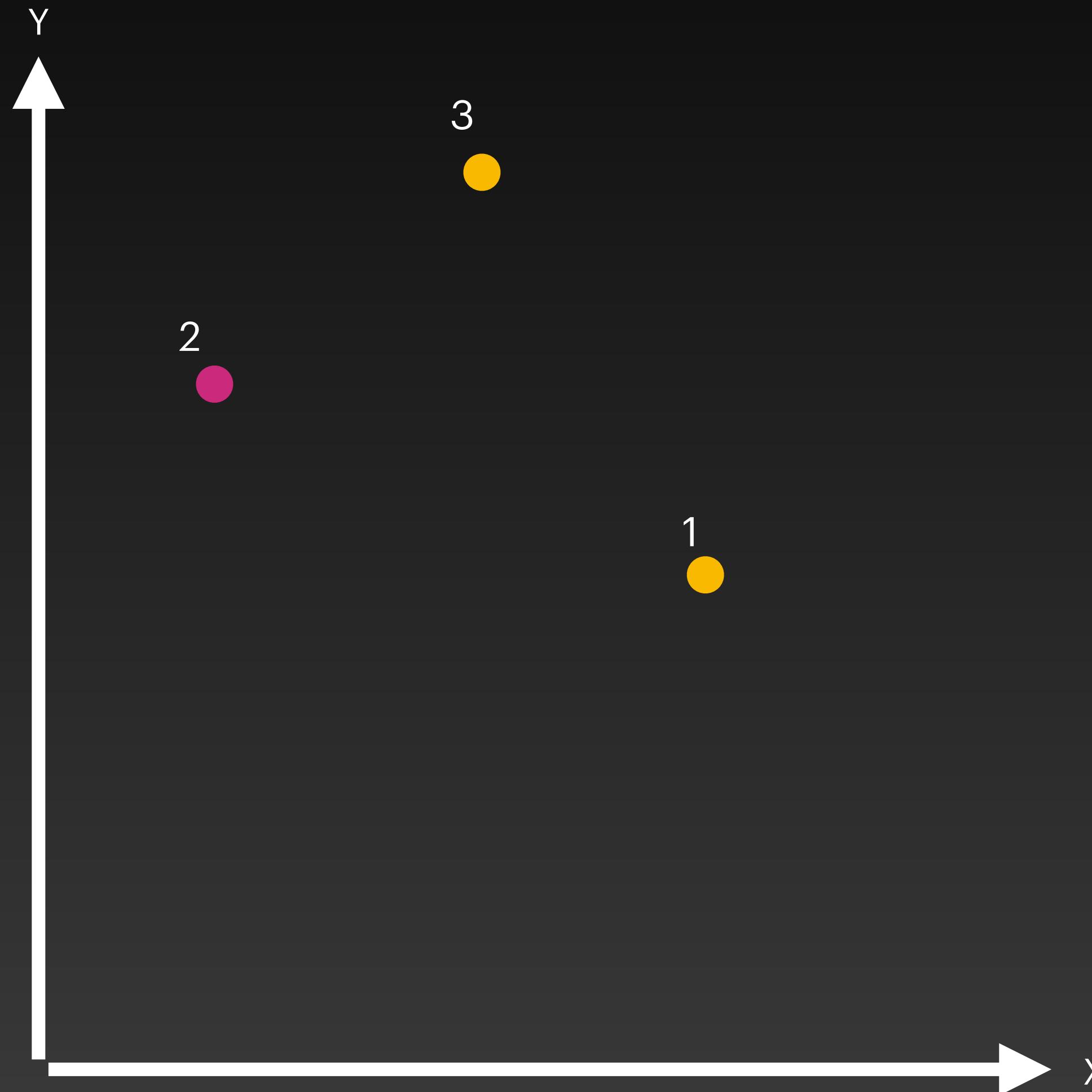
```
InsereQt(qt, ponto, pInfo)
começo
    Quadrante quadrante;
    No pai;

    se (qt == null) então
        qt = CriaNo();
        qt.ponto = ponto;
        qt.info = pInfo;
```



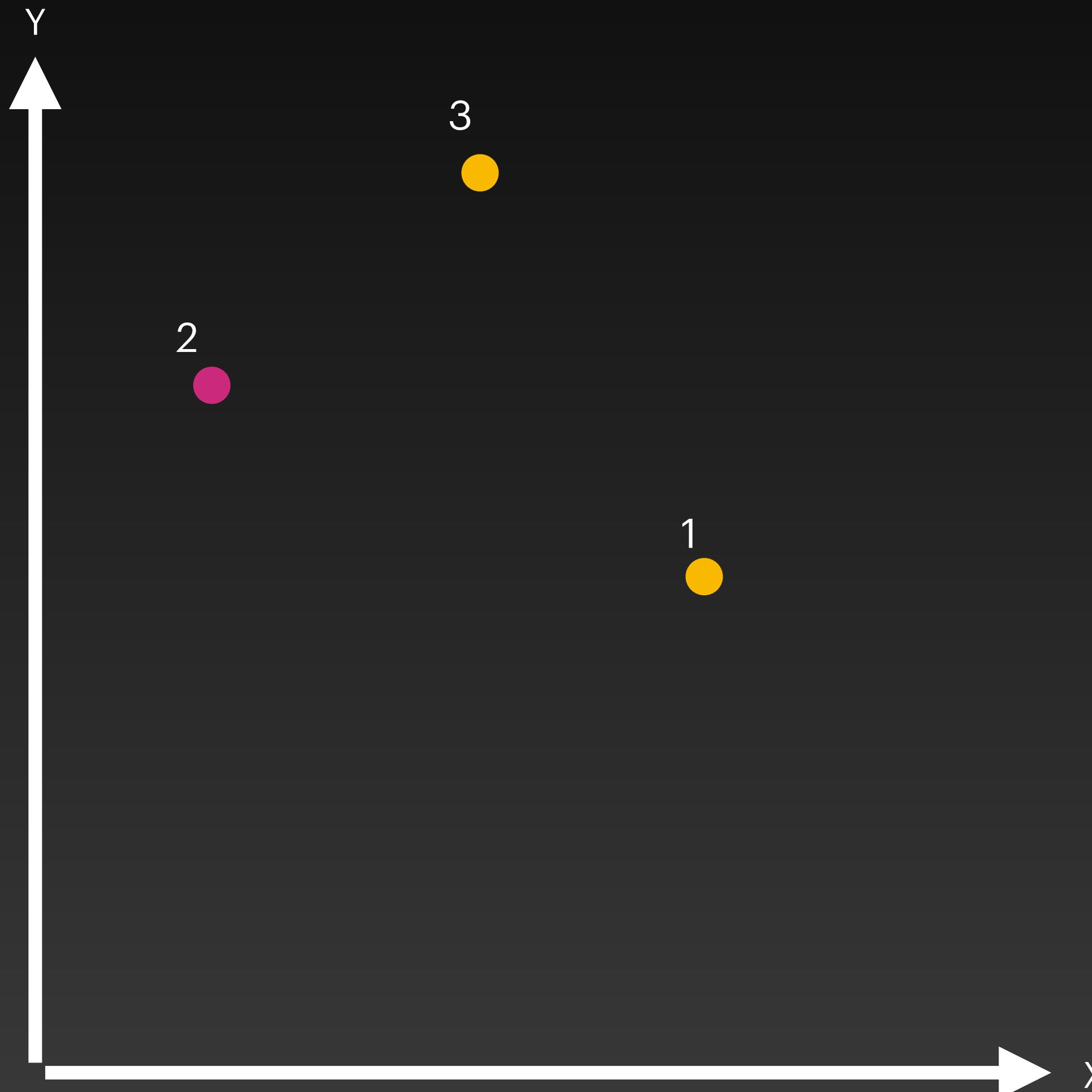
Point QuadTree

Inserção

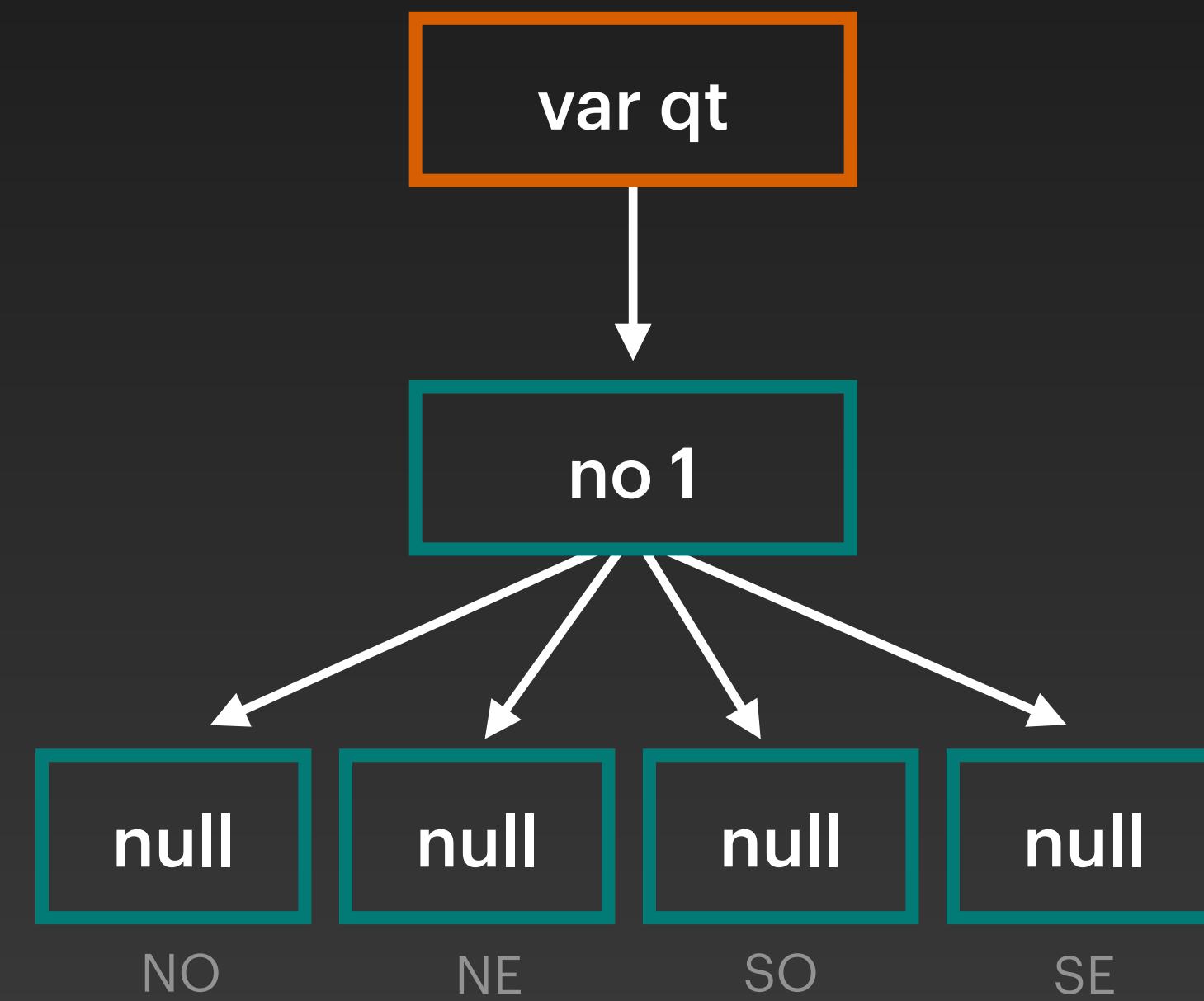


Point QuadTree

Inserção

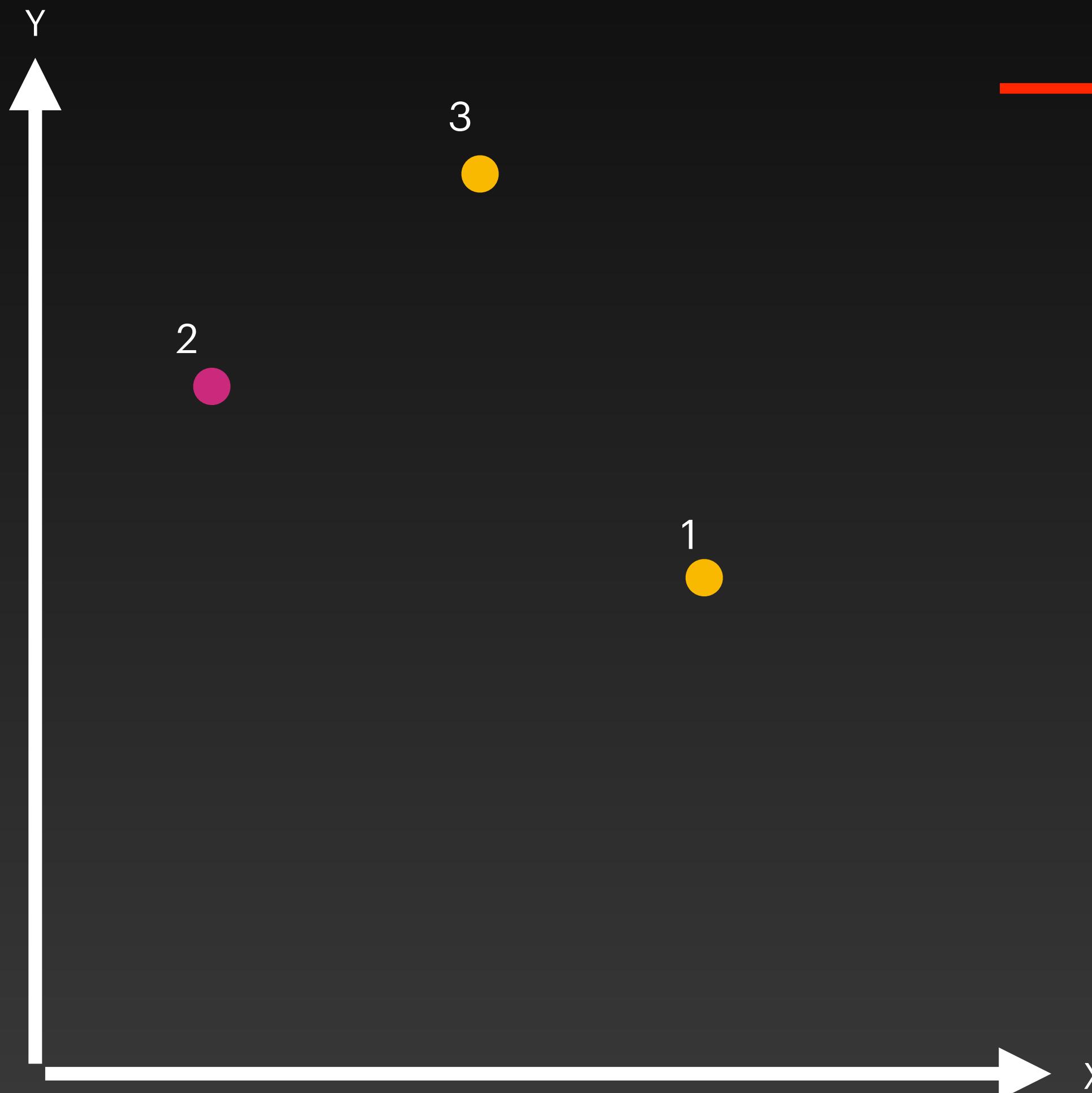


```
senão
    No qtAux = qt;
    enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
        pai = qtAux;
        quadrante = GetQuadrante(ponto, qtAux.ponto);
        qtAux = GetFilho(qtAux, quadrante);
    fim-enquanto
```

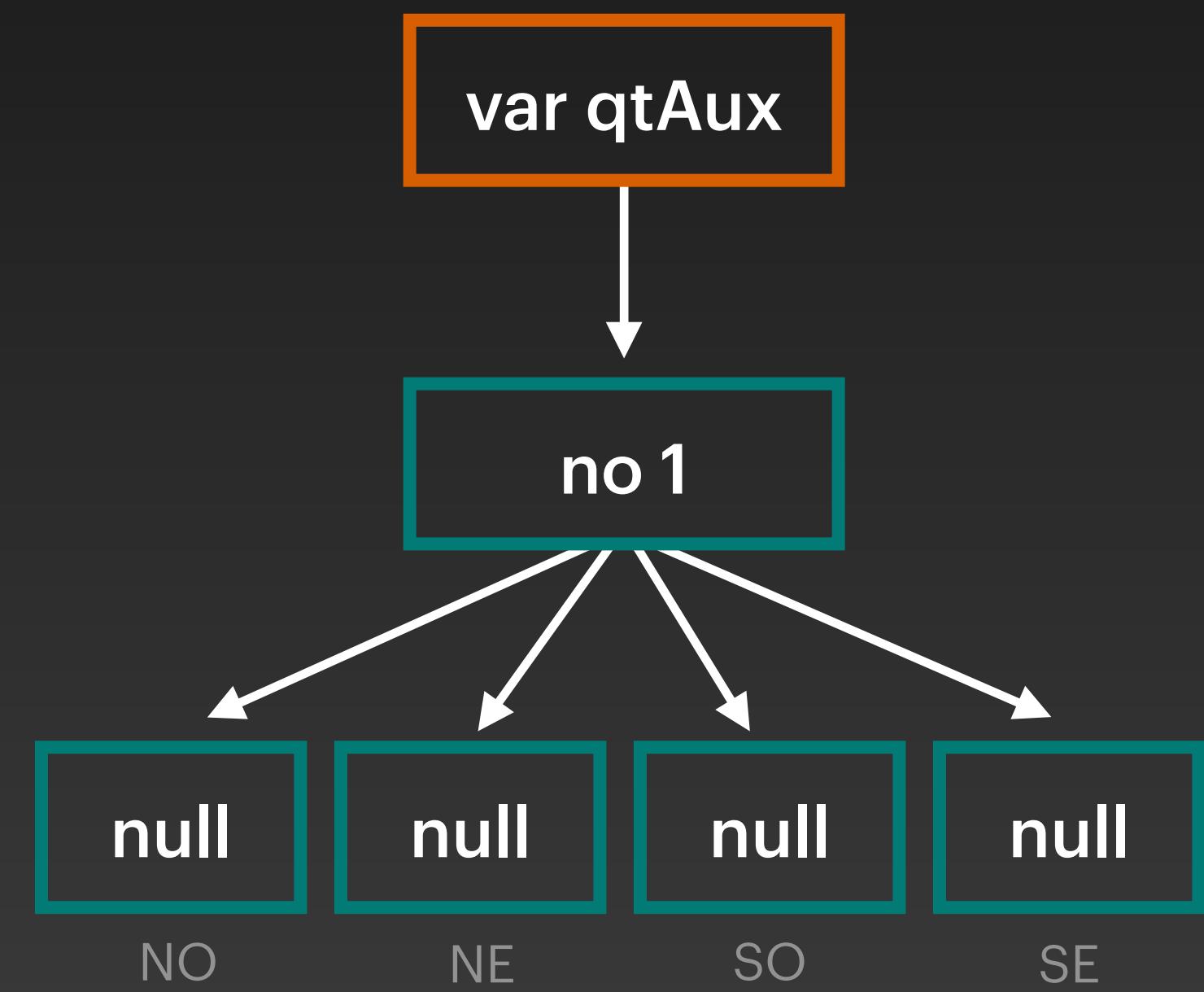


Point QuadTree

Inserção

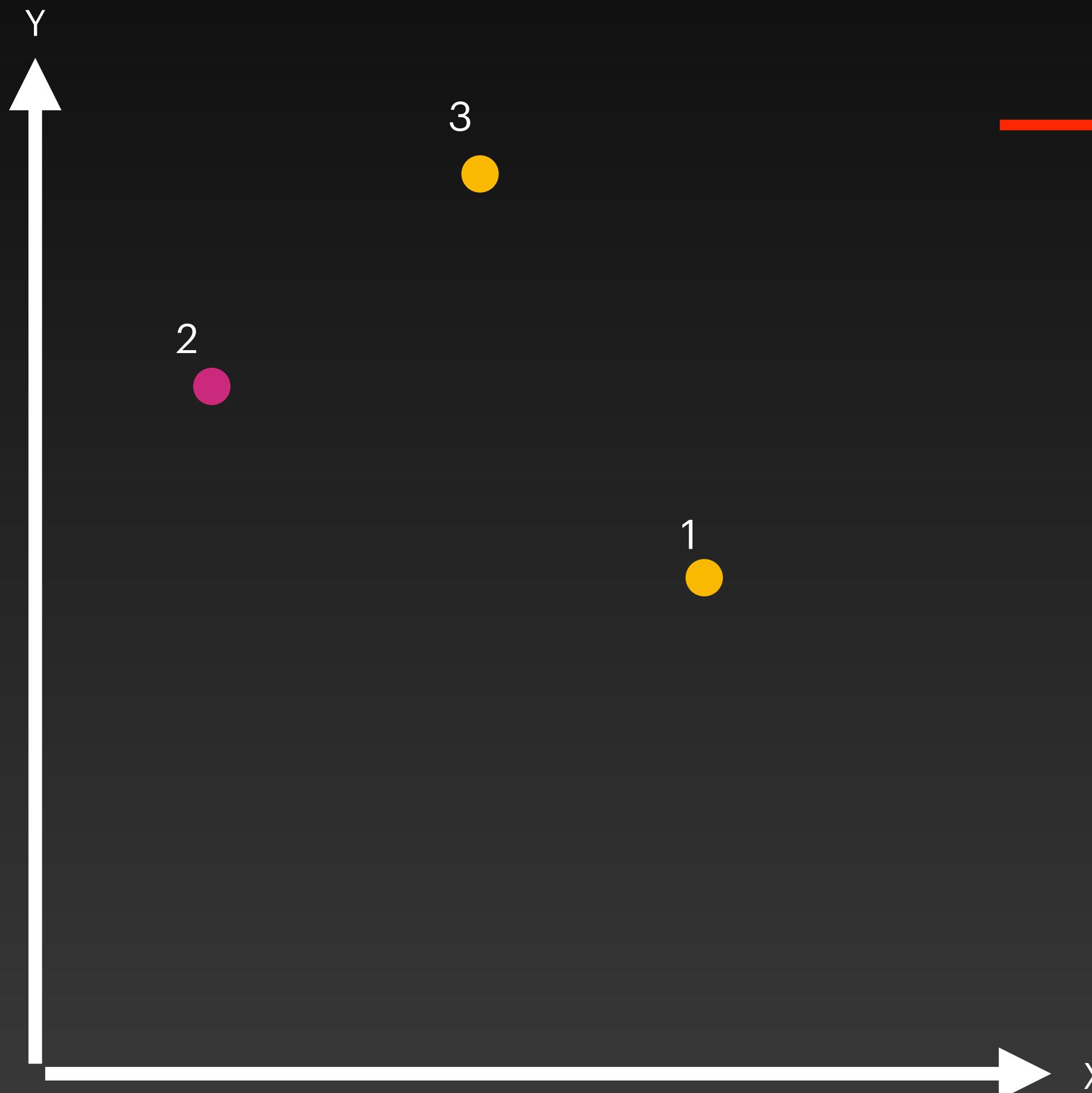


```
senão
    No qtAux = qt;
    enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
        pai = qtAux;
        quadrante = GetQuadrante(ponto, qtAux.ponto);
        qtAux = GetFilho(qtAux, quadrante);
    fim-enquanto
```

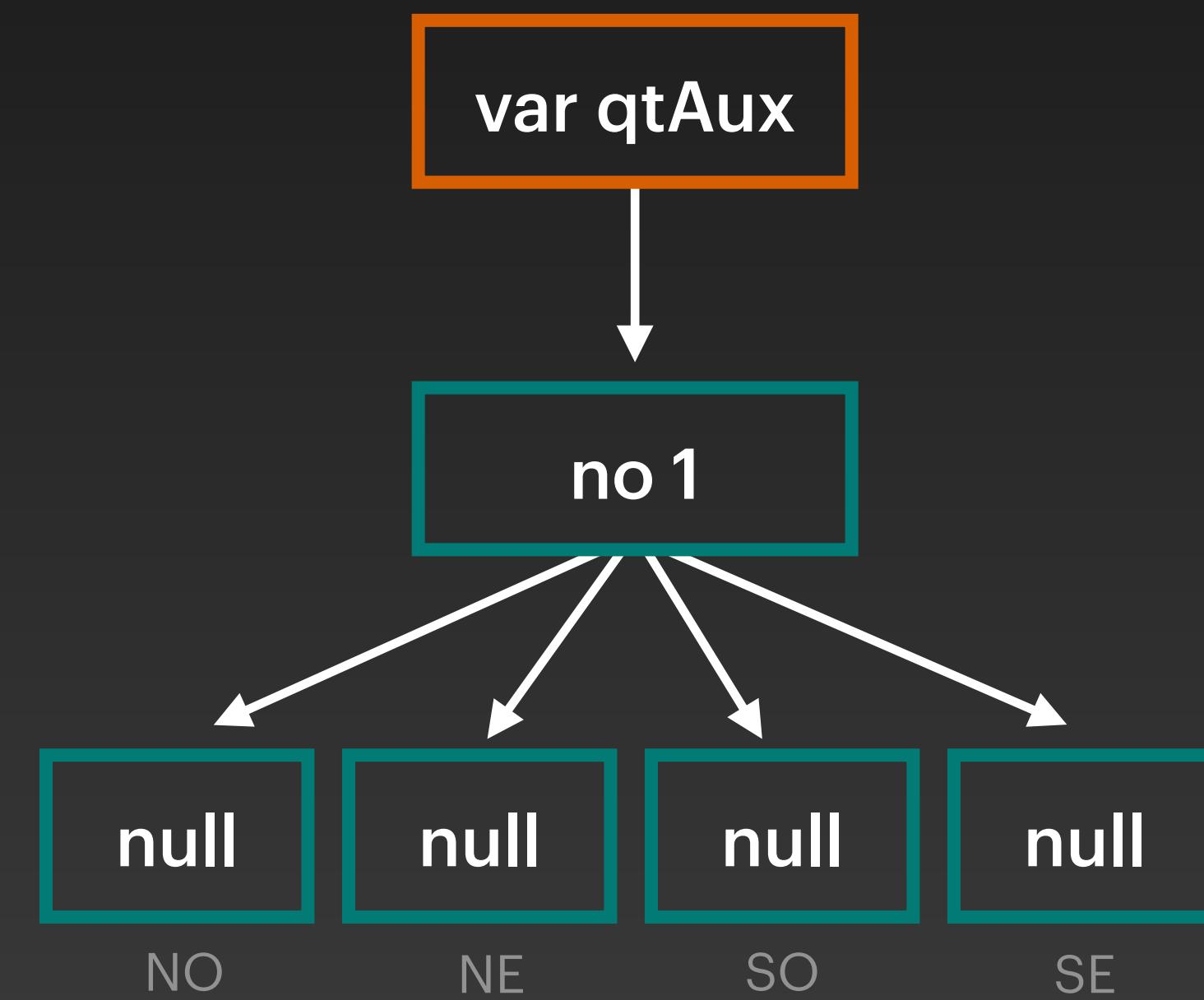


Point QuadTree

Inserção

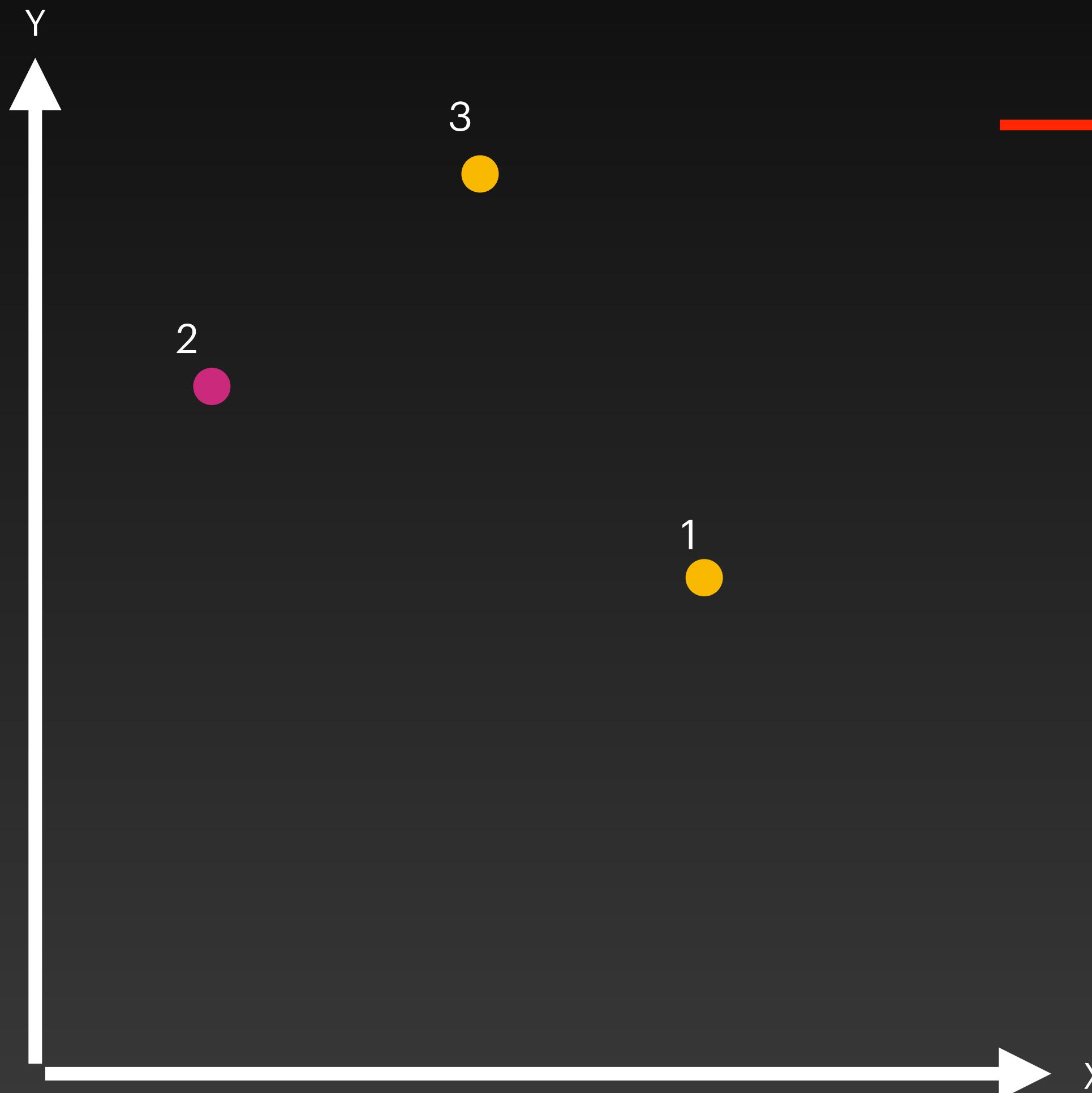


```
senão
    No qtAux = qt;
    enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
        pai = qtAux;
        quadrante = GetQuadrante(ponto, qtAux.ponto);
        qtAux = GetFilho(qtAux, quadrante);
    fim-enquanto
```

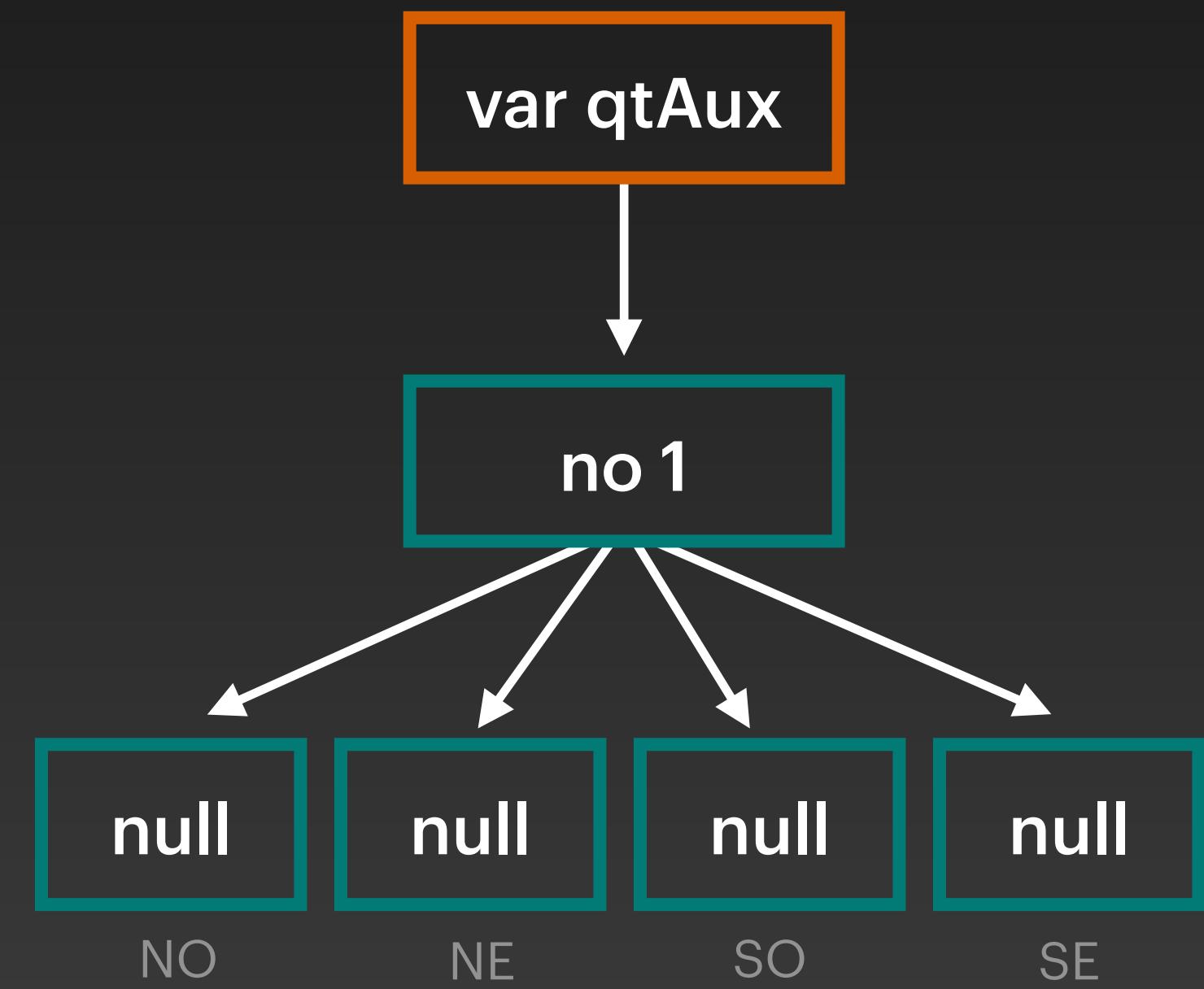


Point QuadTree

Inserção

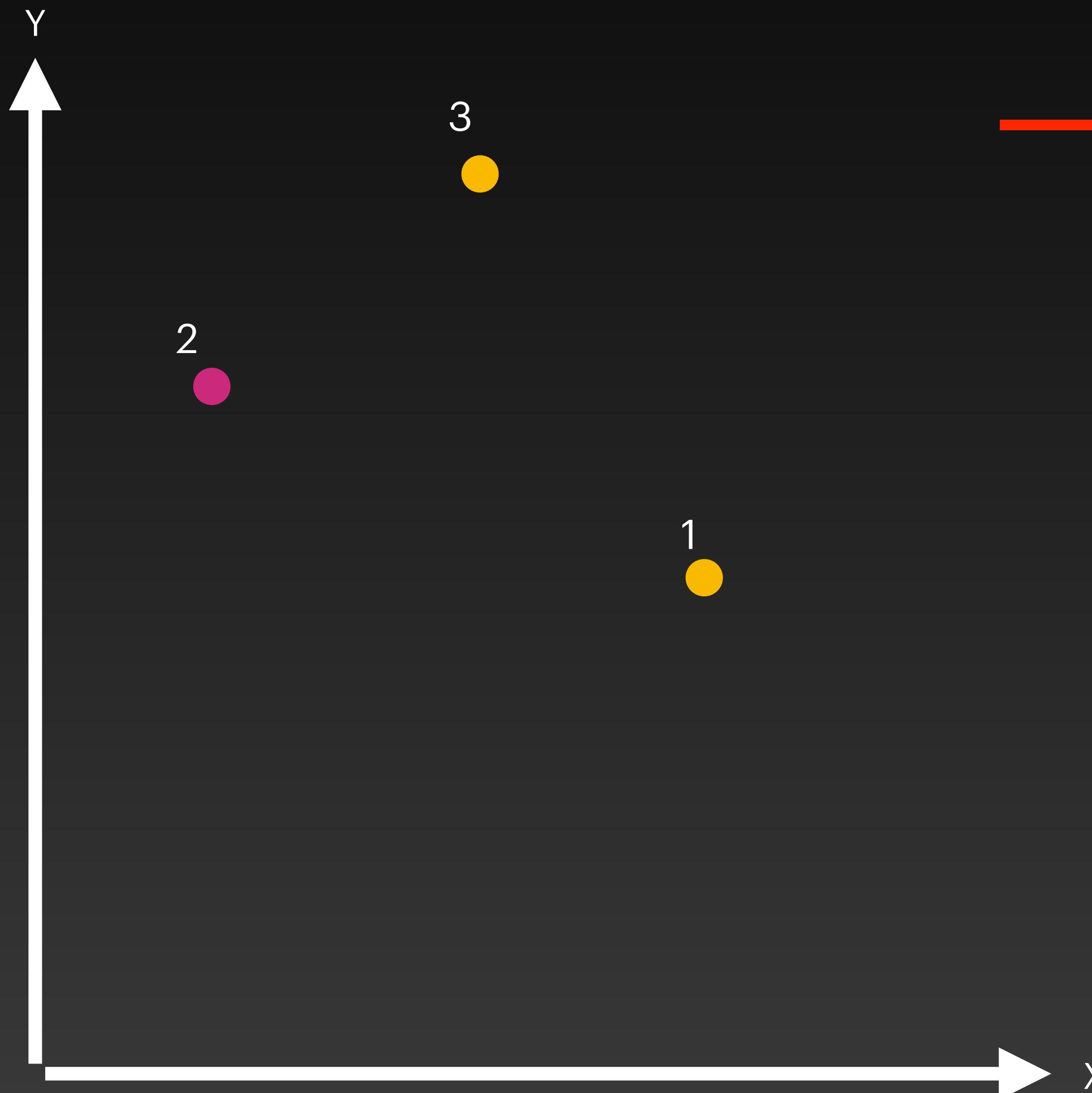


```
senão  
    No qtAux = qt;     X  
    enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))  
        pai = qtAux;  
        quadrante = GetQuadrante(ponto, qtAux.ponto);  
        qtAux = GetFilho(qtAux, quadrante);  
    fim-enquanto
```

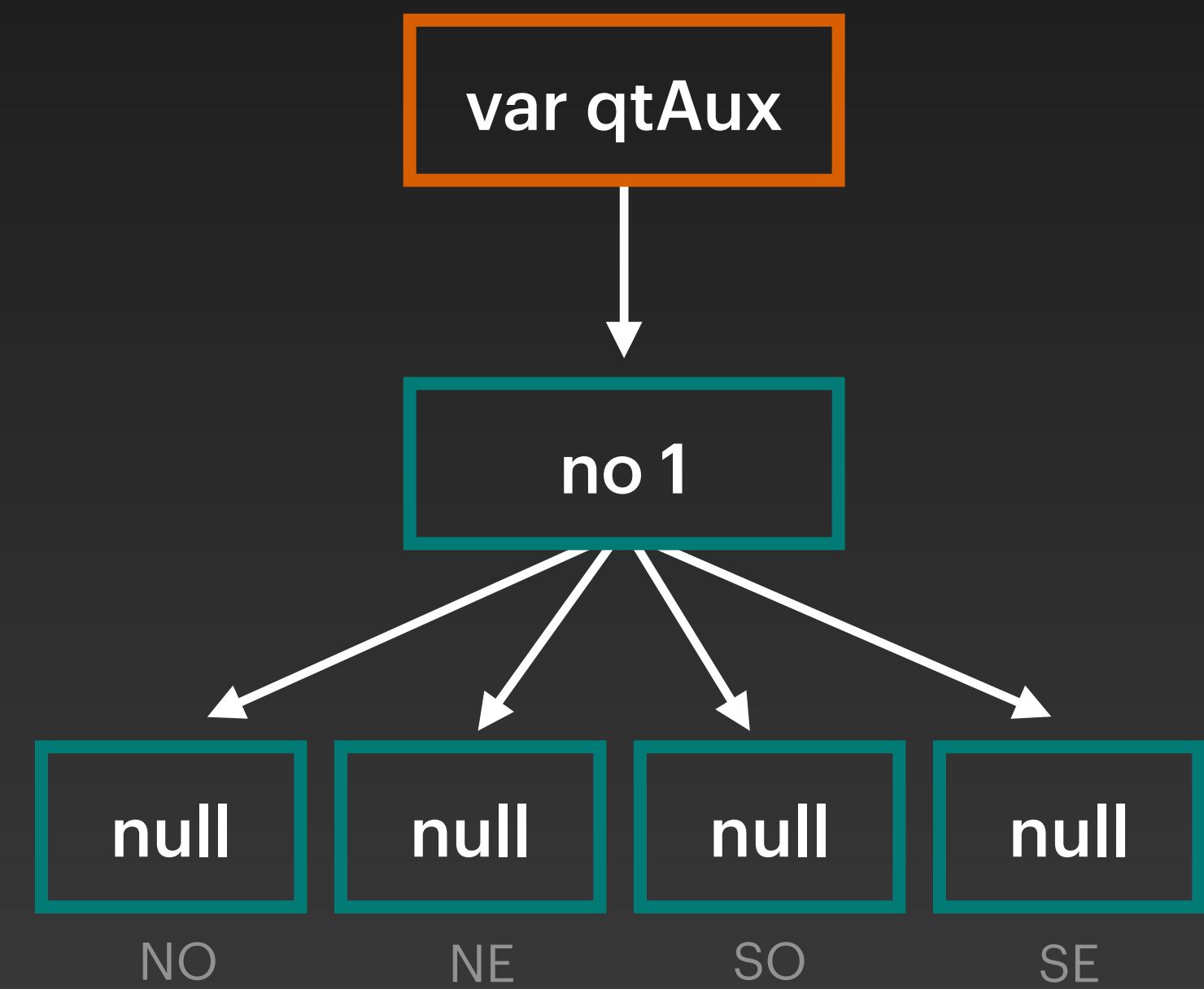


Point QuadTree

Inserção

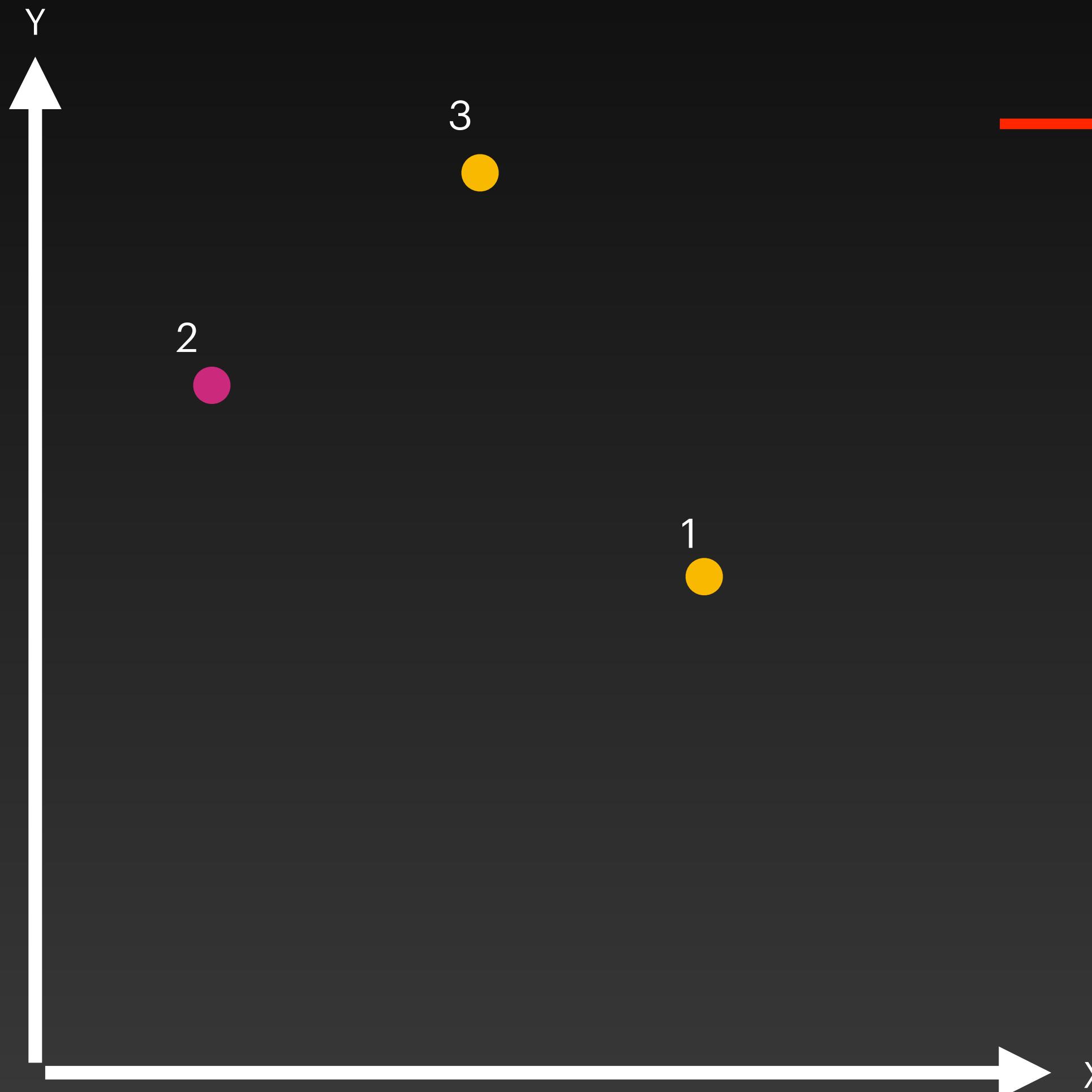


```
senão
    No qtAux = qt;     X
    enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
        pai = qtAux;
        quadrante = GetQuadrante(ponto, qtAux.ponto);
        qtAux = GetFilho(qtAux, quadrante);
    fim-enquanto
```

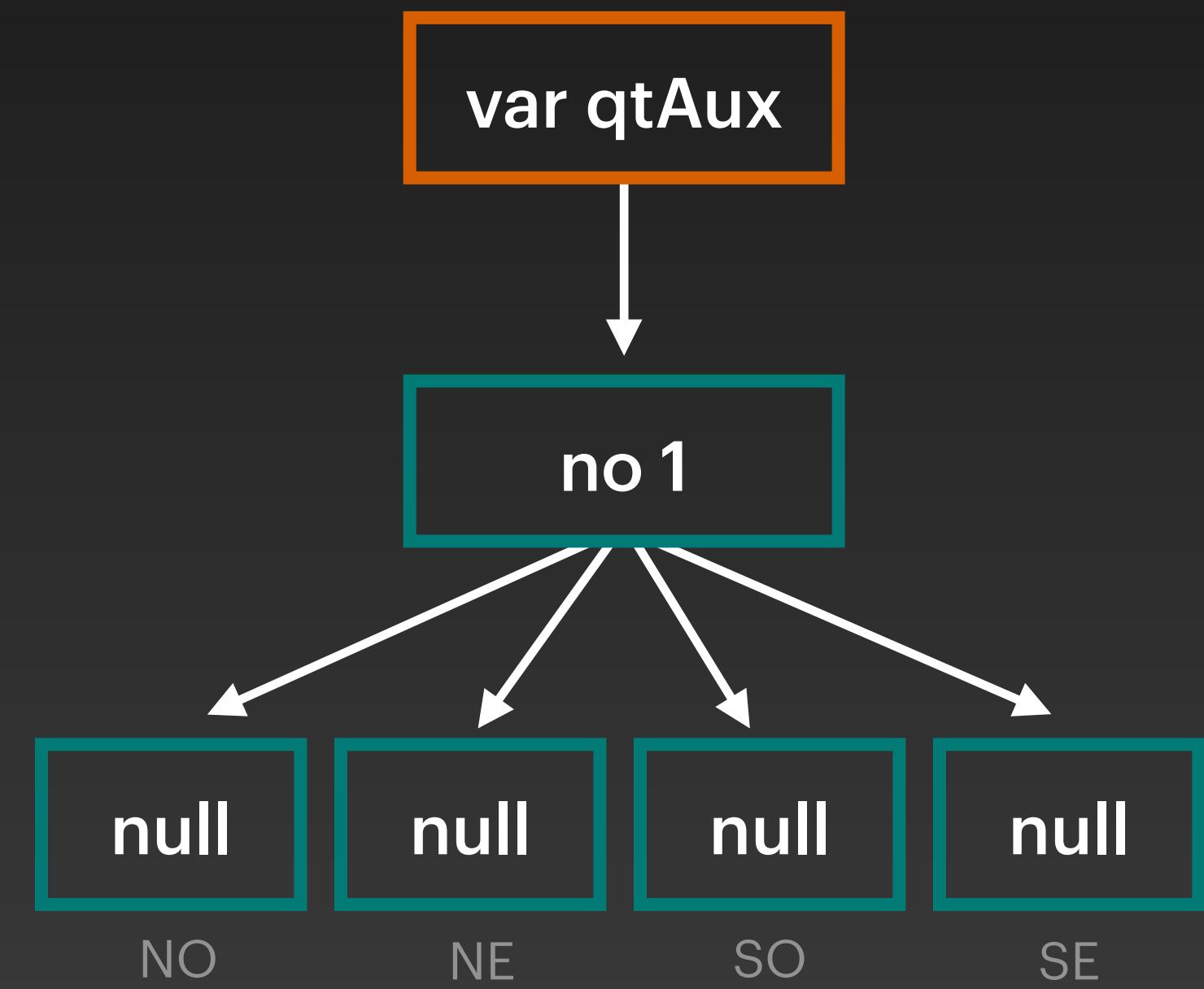


Point QuadTree

Inserção

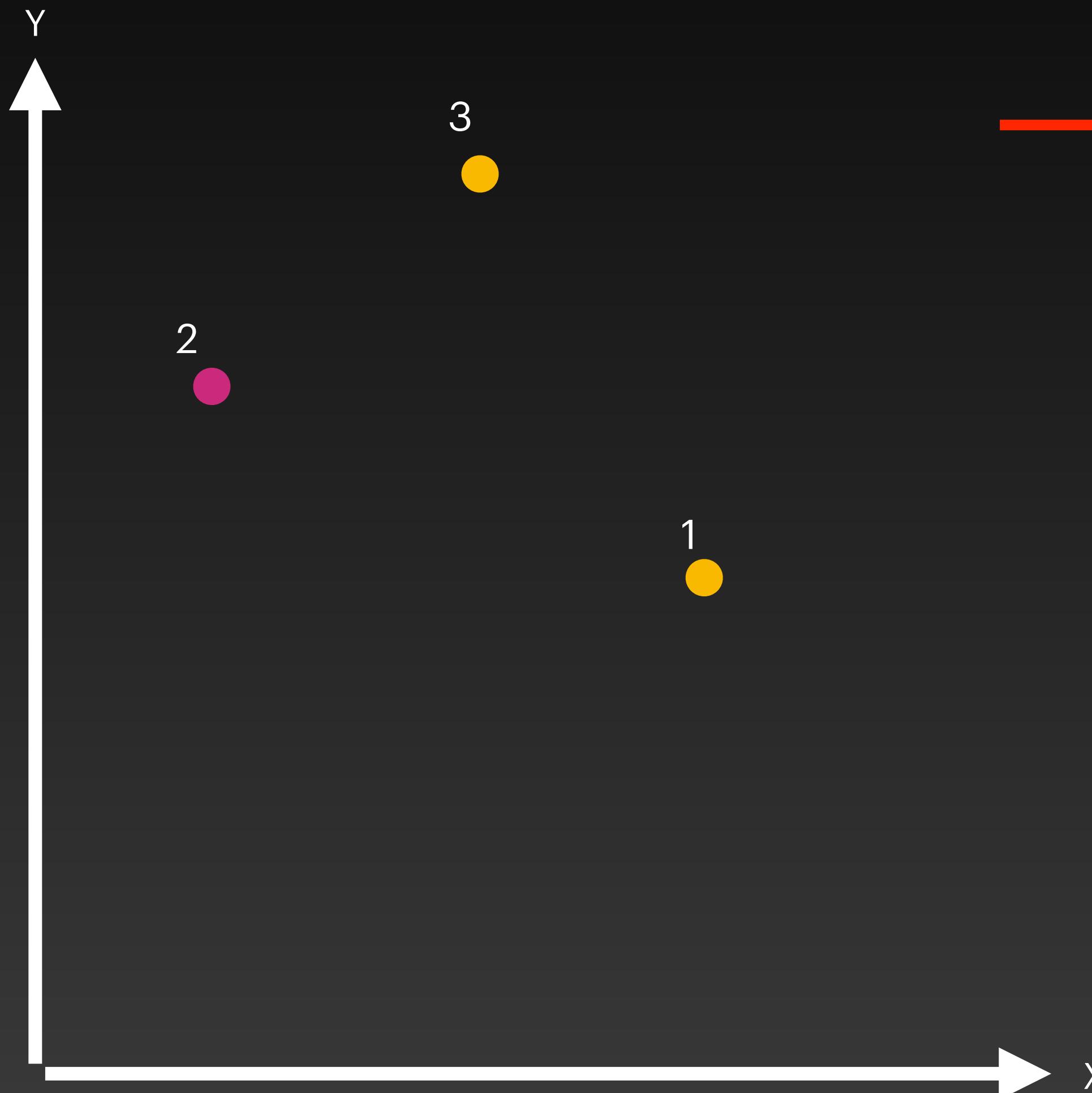


```
senão
    No qtAux = qt; X ✓
    enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
        pai = qtAux;
        quadrante = GetQuadrante(ponto, qtAux.ponto);
        qtAux = GetFilho(qtAux, quadrante);
    fim-enquanto
```

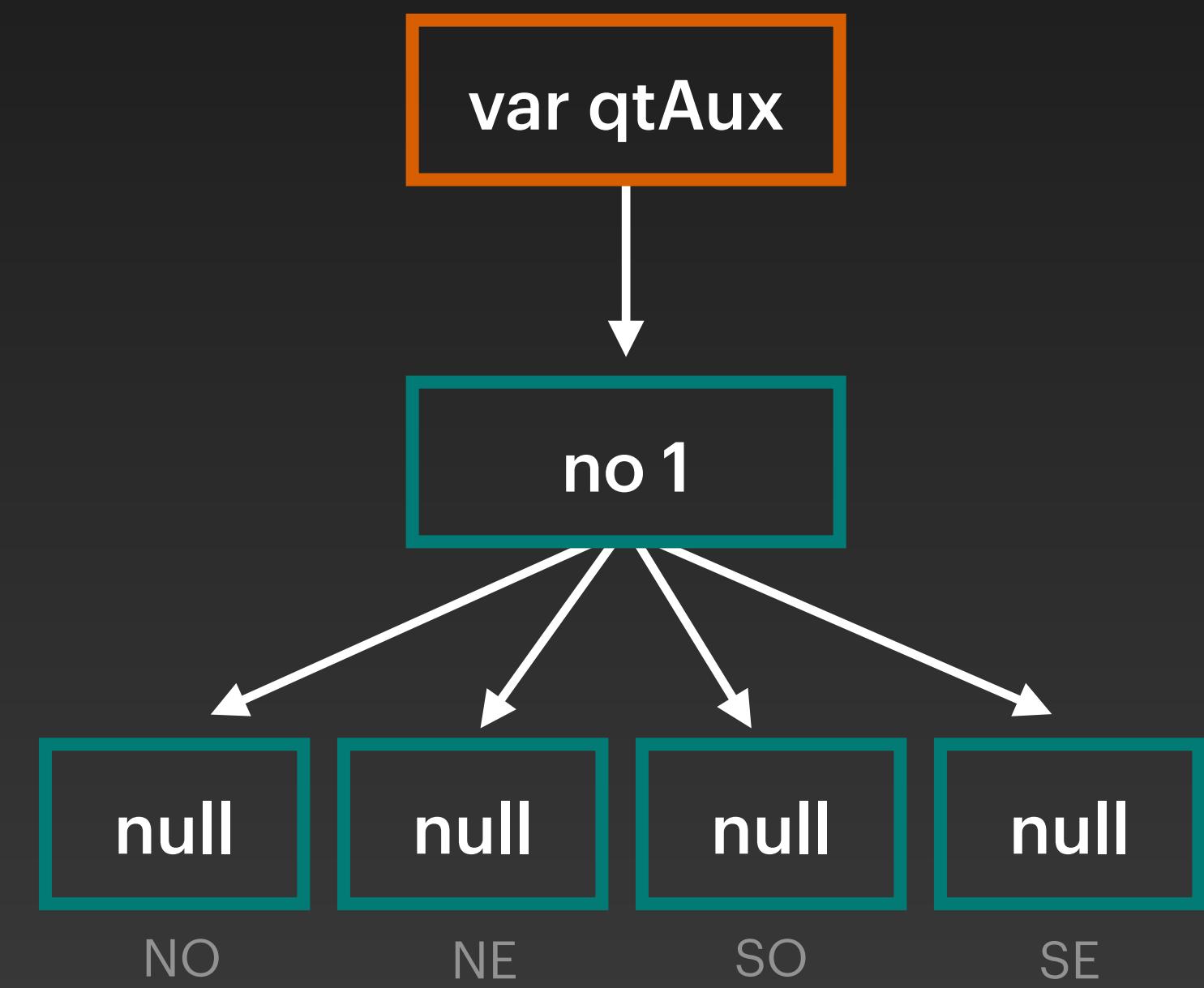


Point QuadTree

Inserção

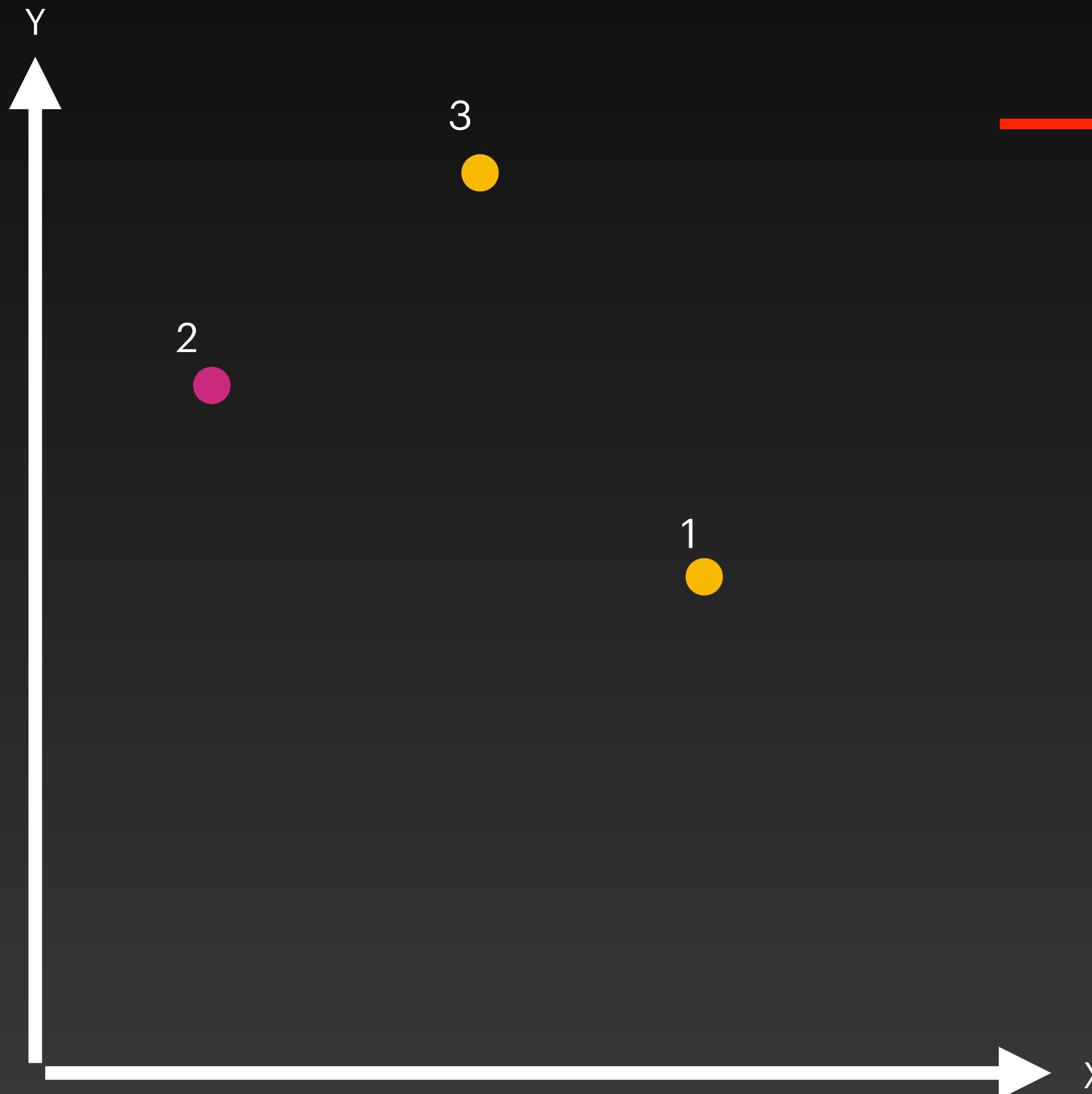


```
senão
    No qtAux = qt; X + ✓ = X
    enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
        pai = qtAux;
        quadrante = GetQuadrante(ponto, qtAux.ponto);
        qtAux = GetFilho(qtAux, quadrante);
    fim-enquanto
```

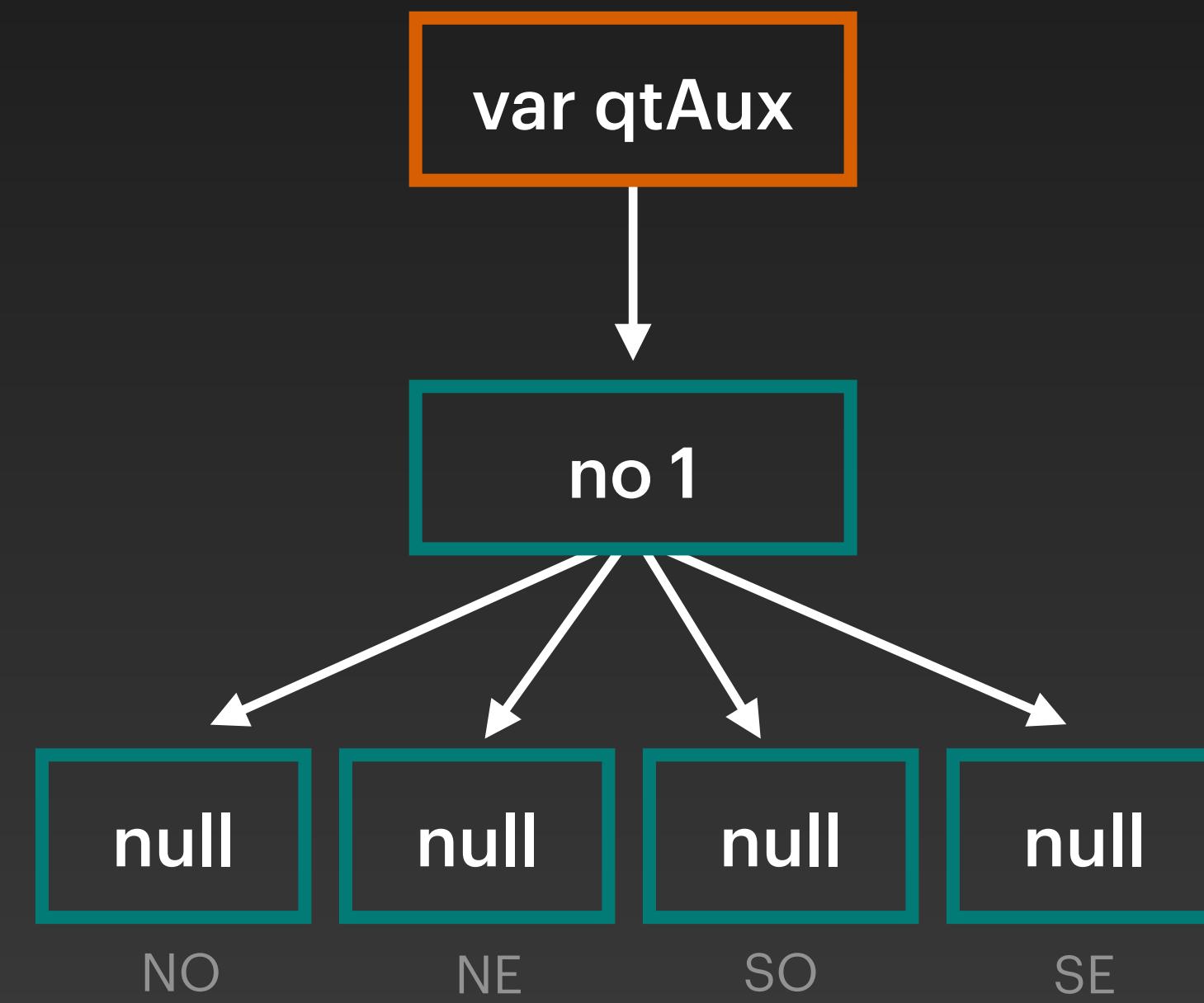


Point QuadTree

Inserção

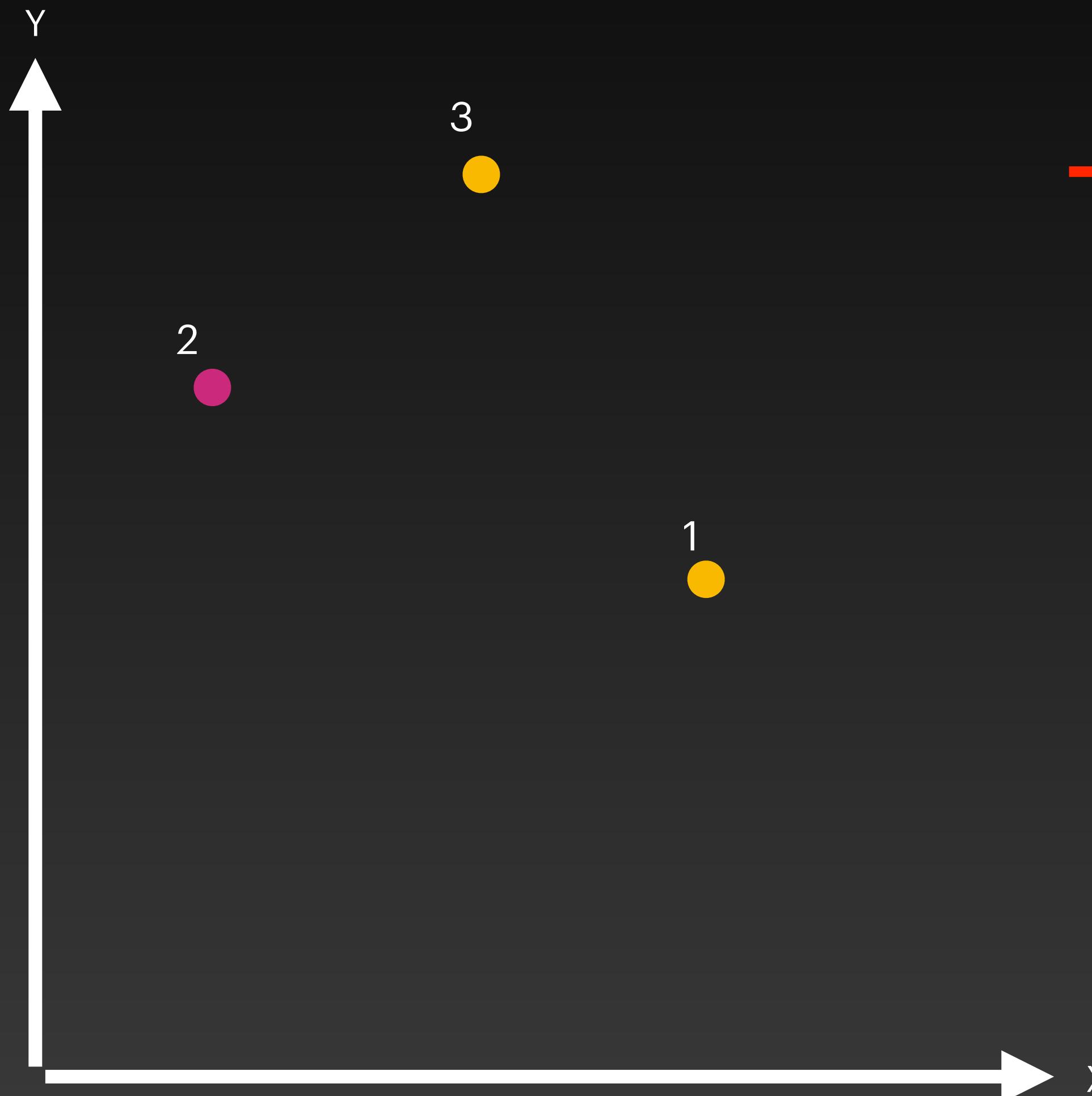


senão
 No qtAux = qt; ~~X~~ + ~~✓~~ = ~~X~~ -> ✓
 enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
 pai = qtAux;
 quadrante = GetQuadrante(ponto, qtAux.ponto);
 qtAux = GetFilho(qtAux, quadrante);
 fim-enquanto

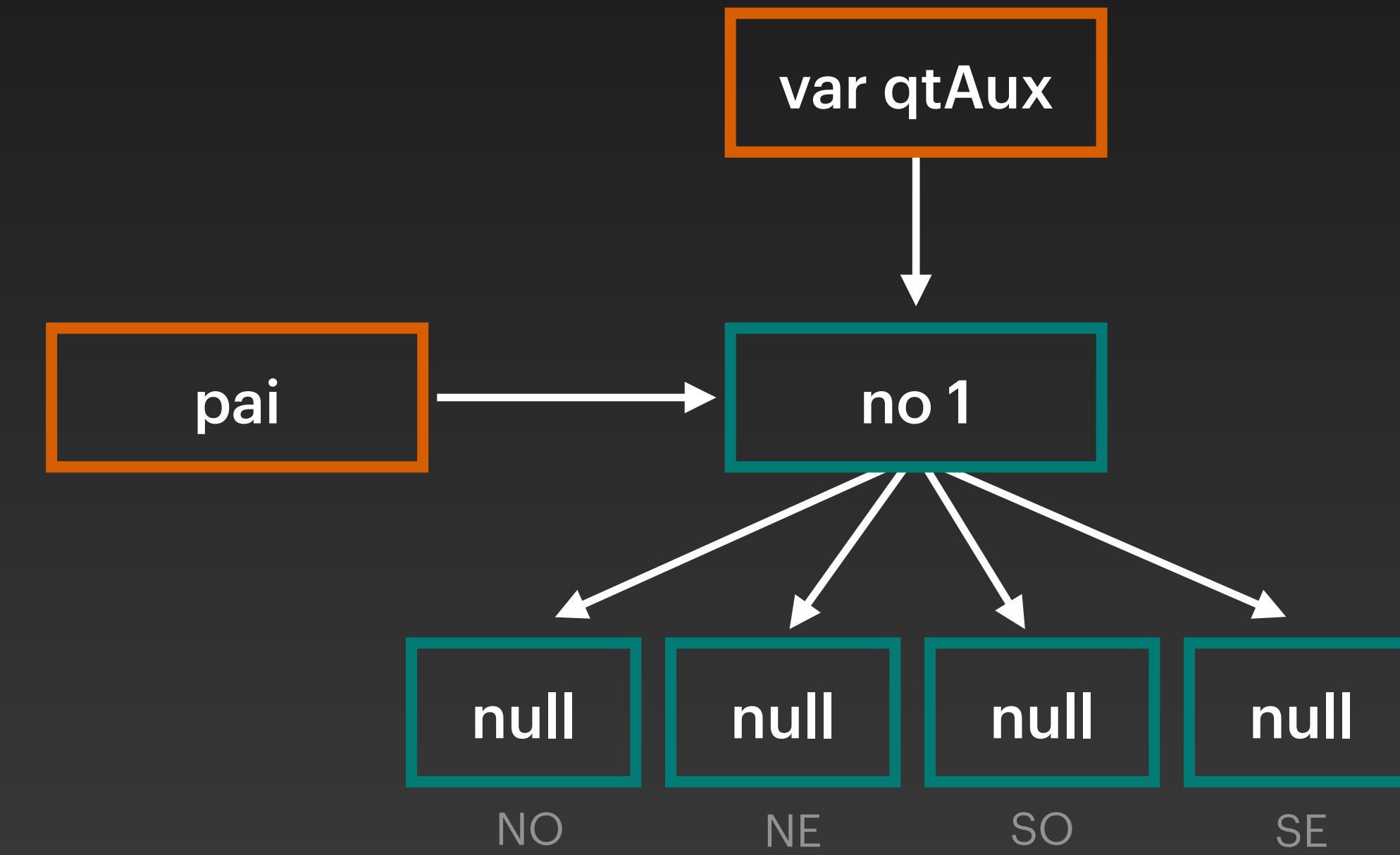


Point QuadTree

Inserção

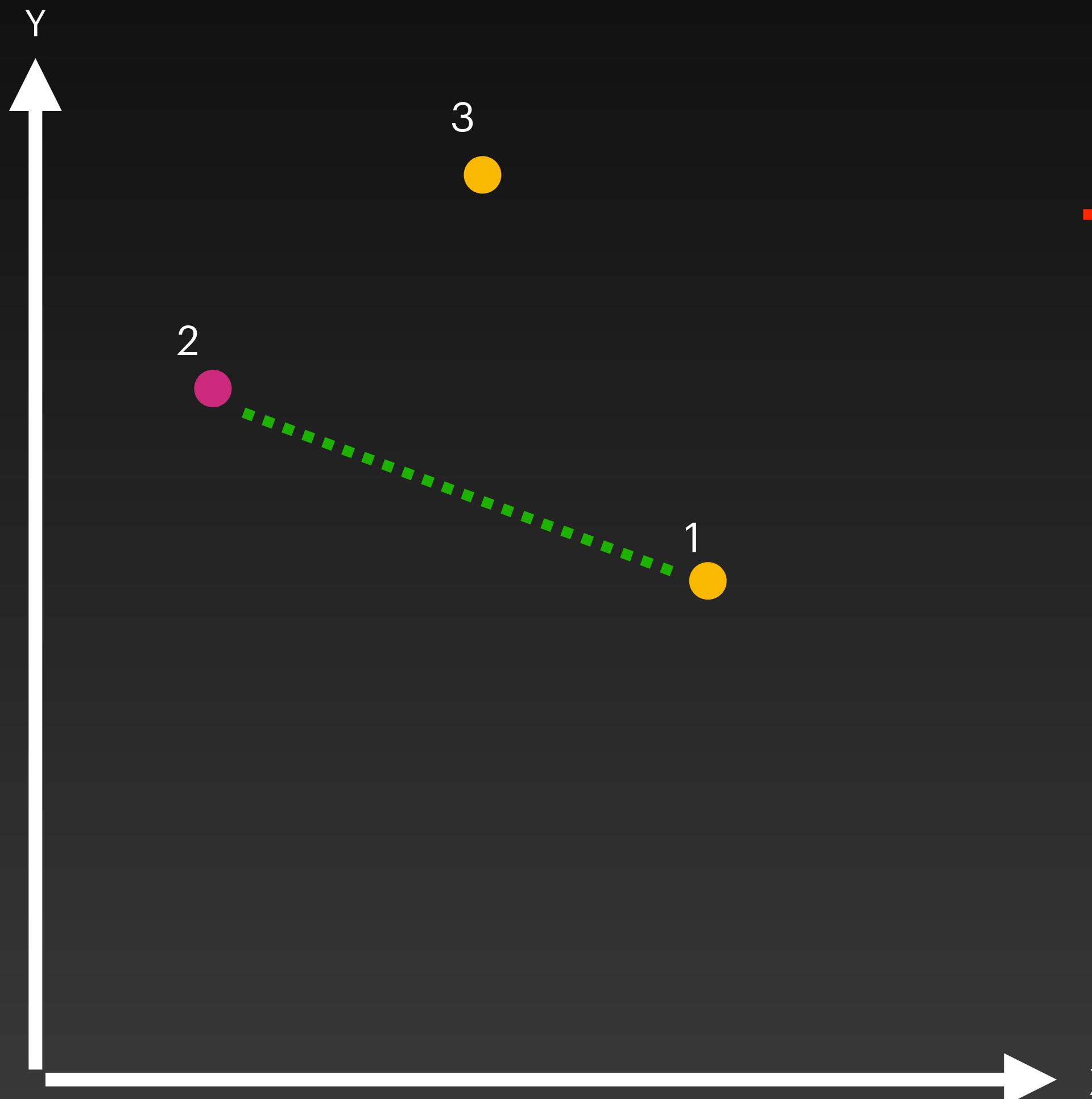


```
senão
    No qtAux = qt;
    enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
        pai = qtAux;
        quadrante = GetQuadrante(ponto, qtAux.ponto);
        qtAux = GetFilho(qtAux, quadrante);
    fim-enquanto
```

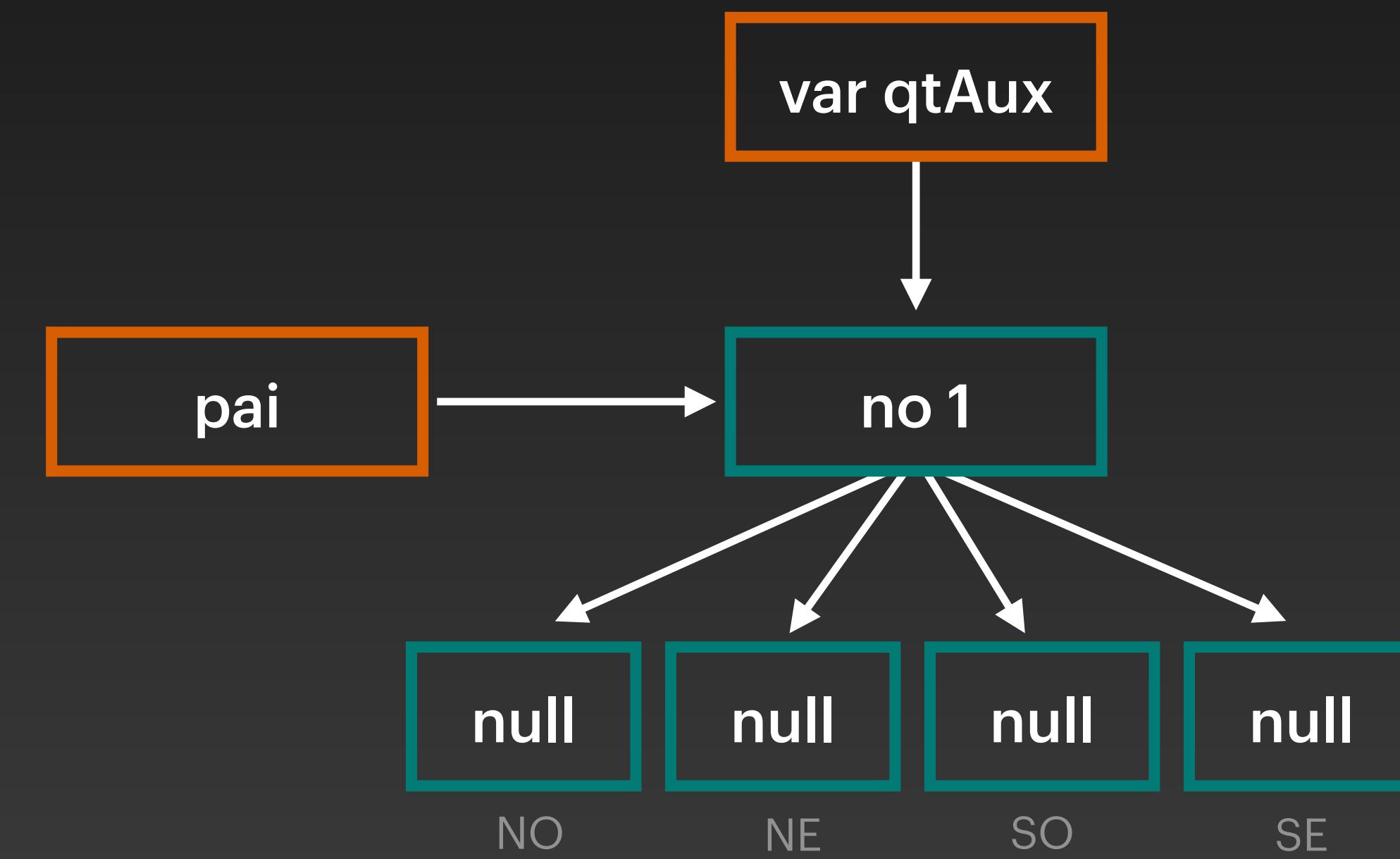


Point QuadTree

Inserção

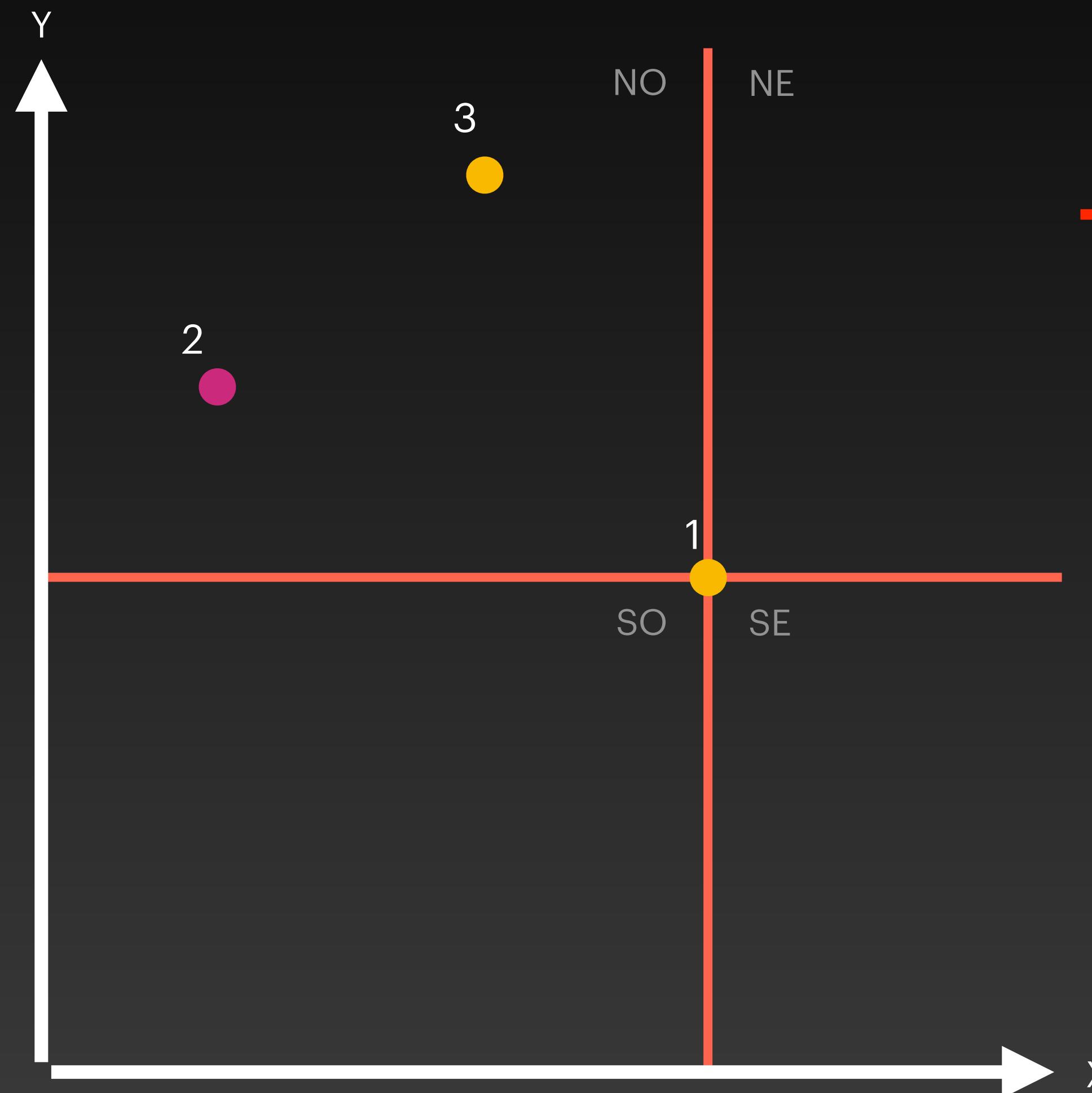


```
senão
  No qtAux = qt;
  enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
    pai = qtAux;
    quadrante = GetQuadrante(ponto, qtAux.ponto);
    qtAux = GetFilho(qtAux, quadrante);
  fim-enquanto
```

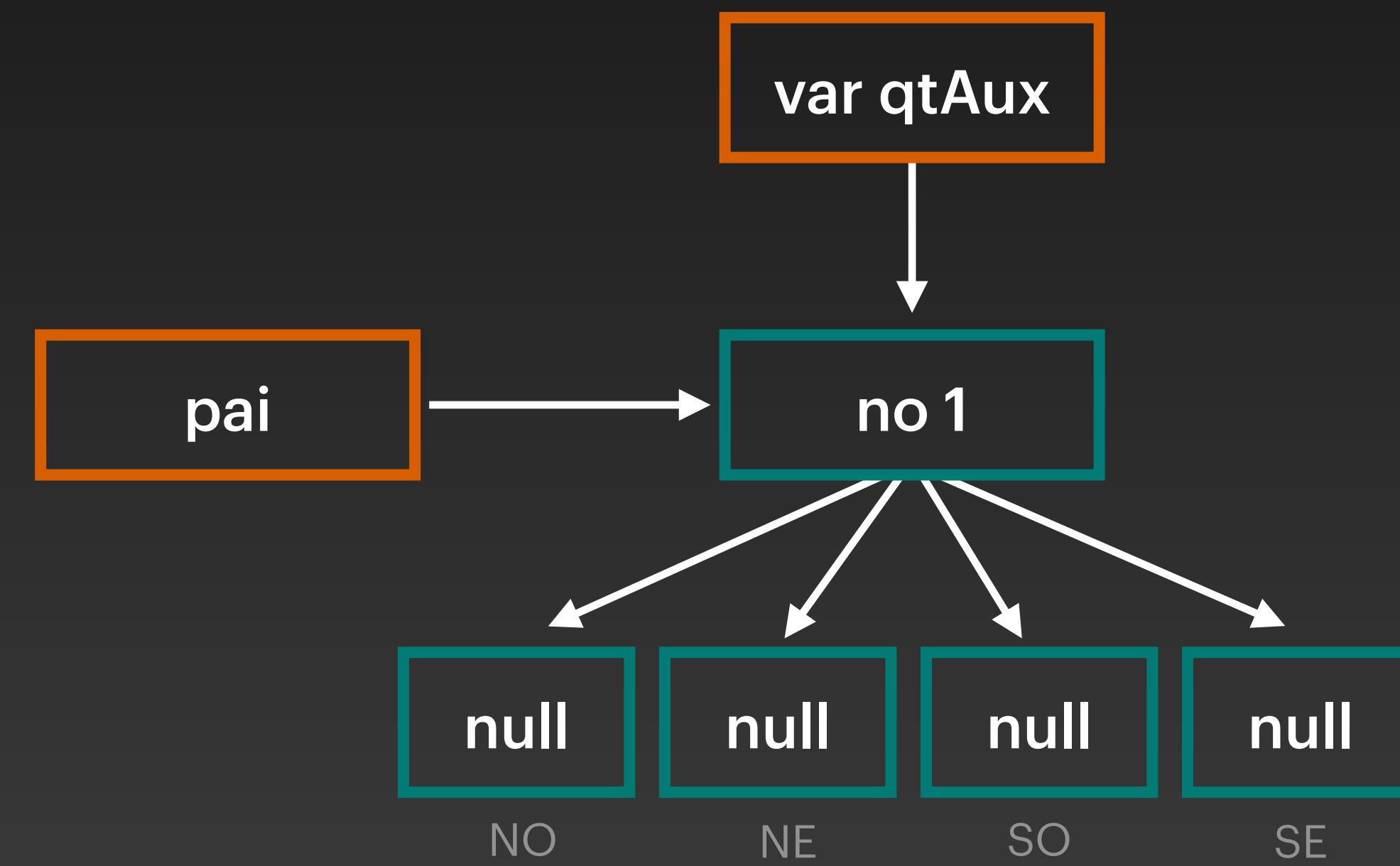


Point QuadTree

Inserção

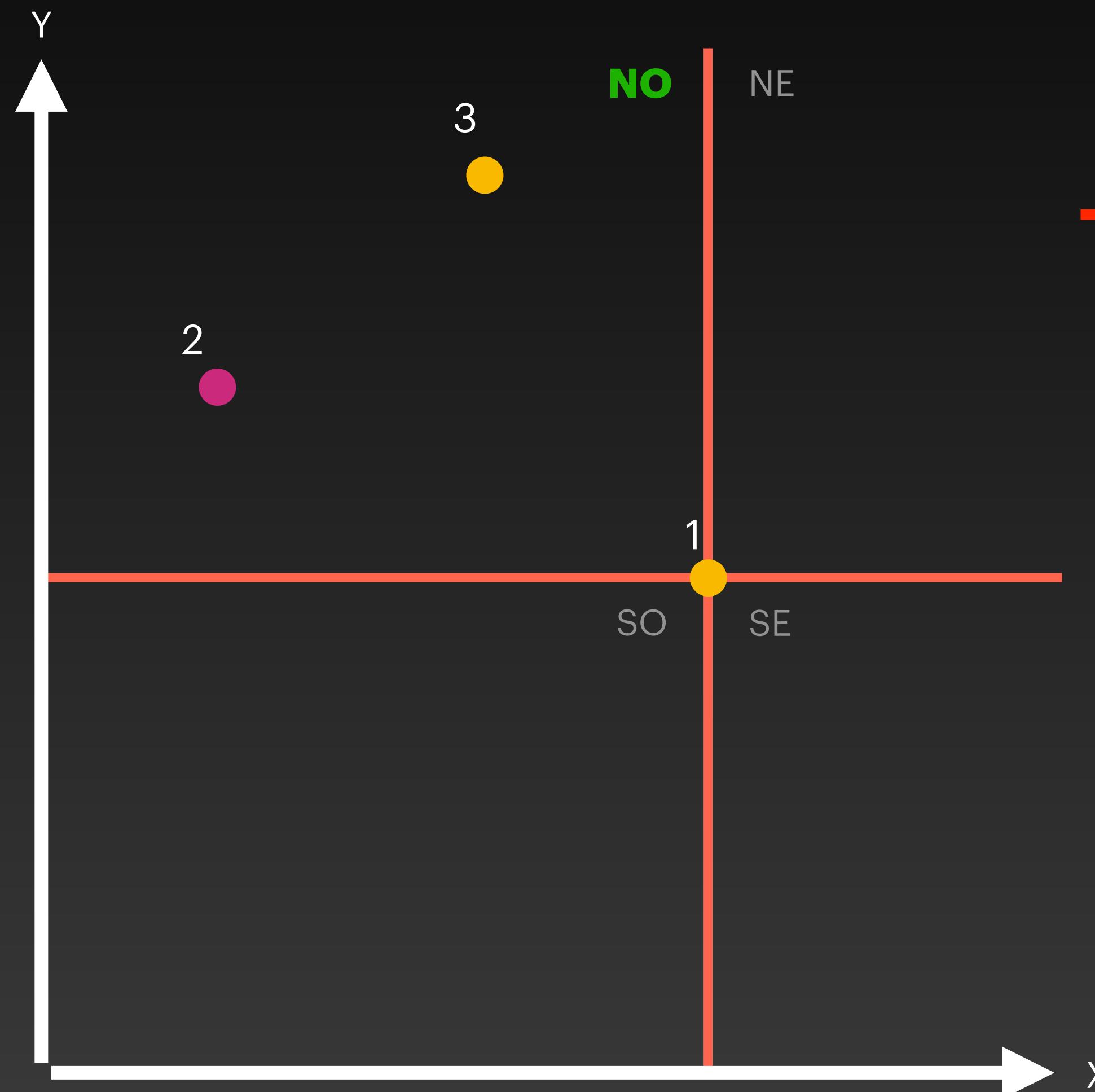


```
senão
  No qtAux = qt;
  enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
    pai = qtAux;
    quadrante = GetQuadrante(ponto, qtAux.ponto);
    qtAux = GetFilho(qtAux, quadrante);
  fim-enquanto
```

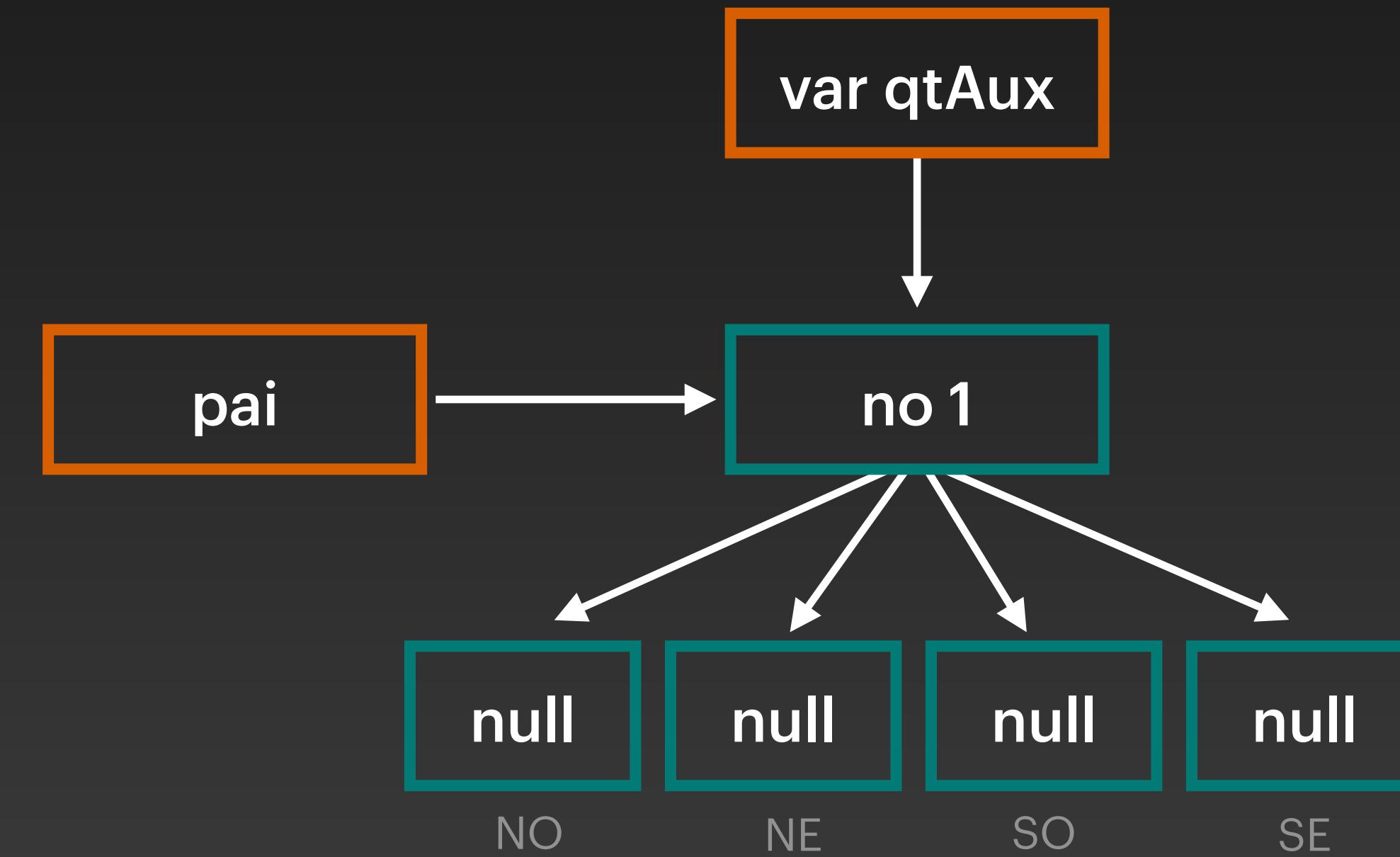


Point QuadTree

Inserção

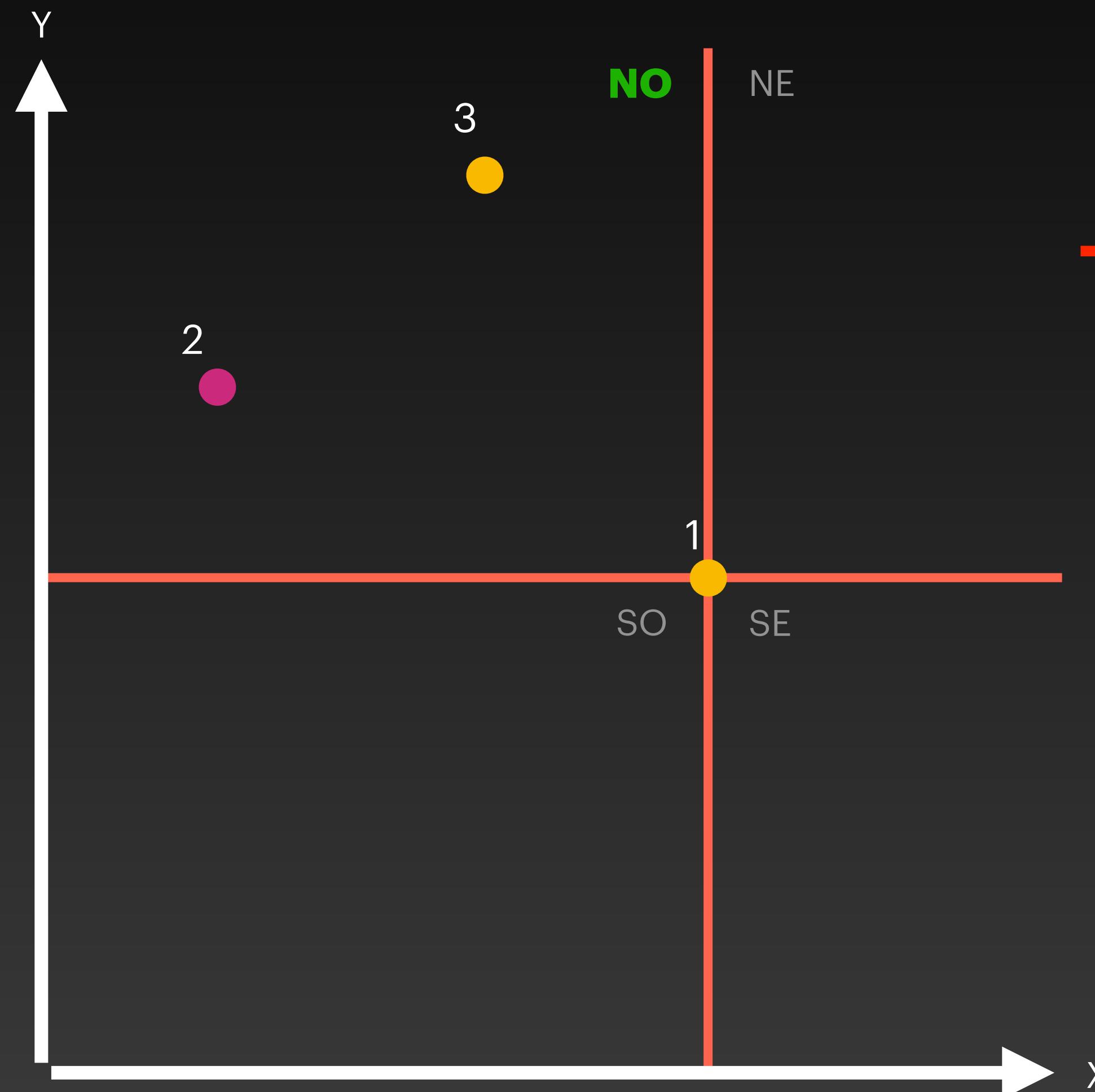


```
senão
  No qtAux = qt;
  enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
    pai = qtAux;
    quadrante = GetQuadrante(ponto, qtAux.ponto);
    qtAux = GetFilho(qtAux, quadrante);
  fim-enquanto
```

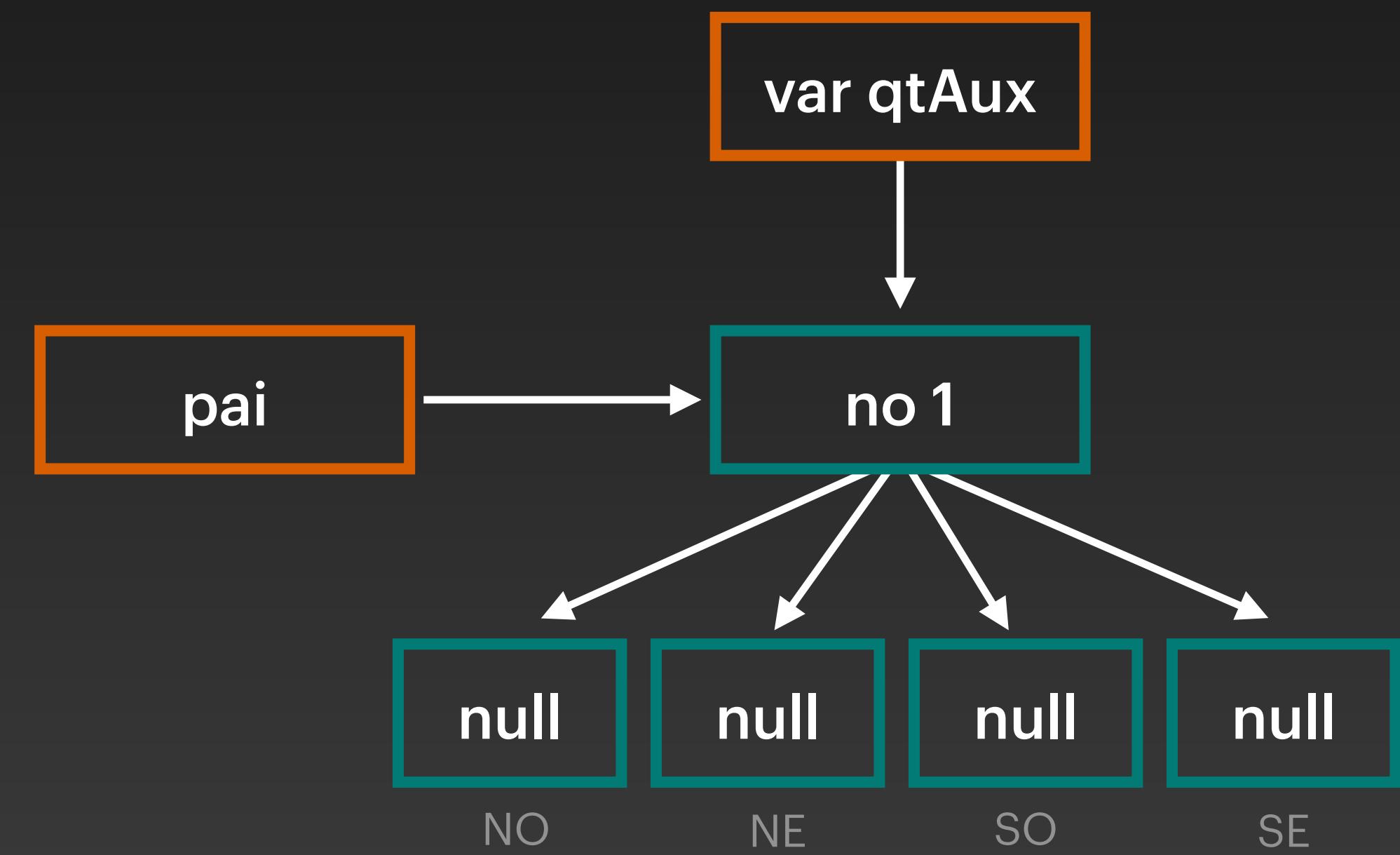


Point QuadTree

Inserção

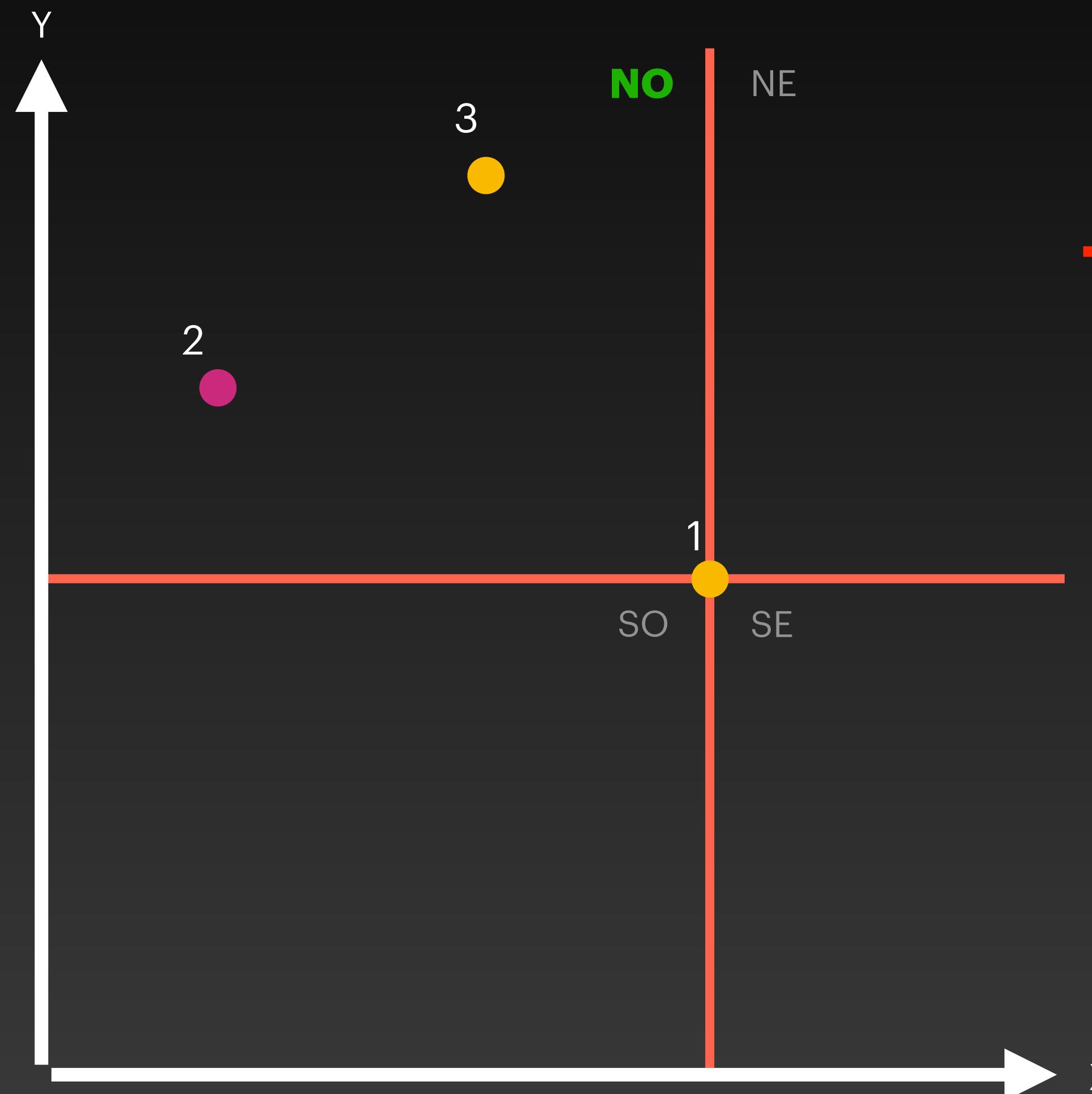


```
senão
  No qtAux = qt;
  enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
    pai = qtAux;
    quadrante = GetQuadrante(ponto, qtAux.ponto);
    qtAux = GetFilho(qtAux, quadrante);
  fim-enquanto
```

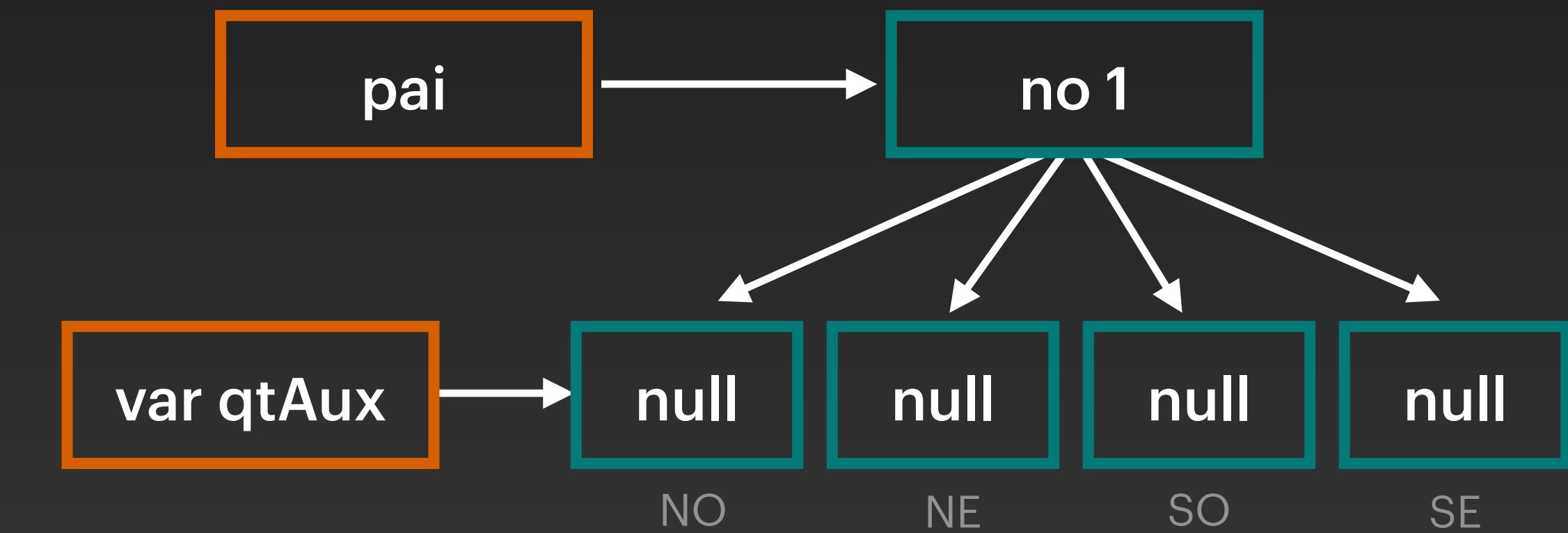


Point QuadTree

Inserção

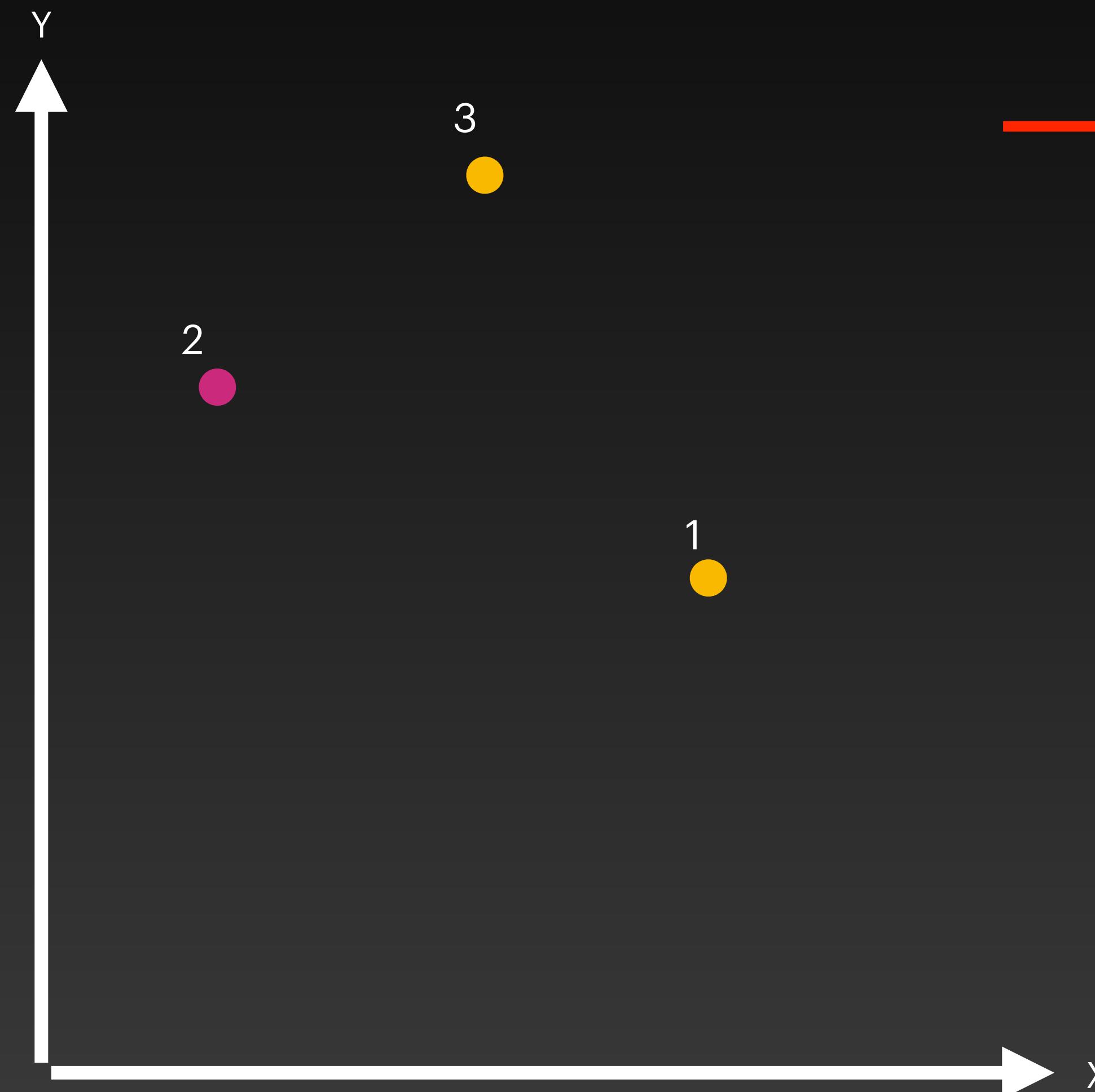


```
senão
  No qtAux = qt;
  enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
    pai = qtAux;
    quadrante = GetQuadrante(ponto, qtAux.ponto);
    qtAux = GetFilho(qtAux, quadrante);
  fim-enquanto
```

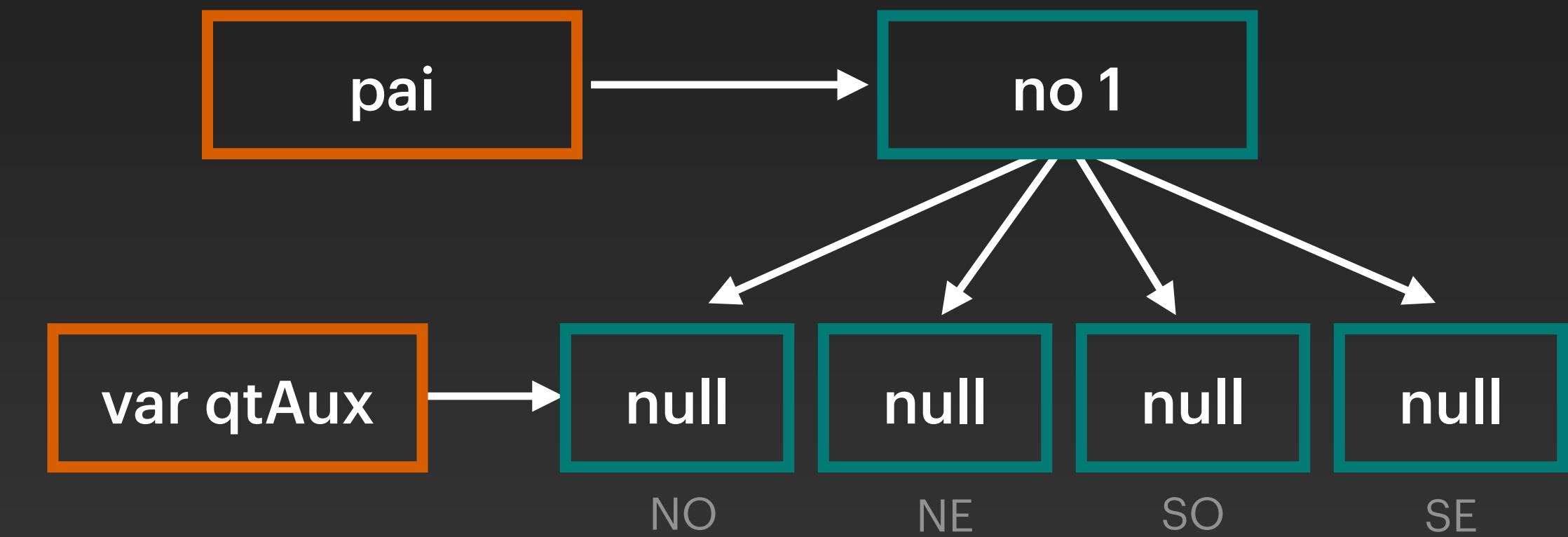


Point QuadTree

Inserção

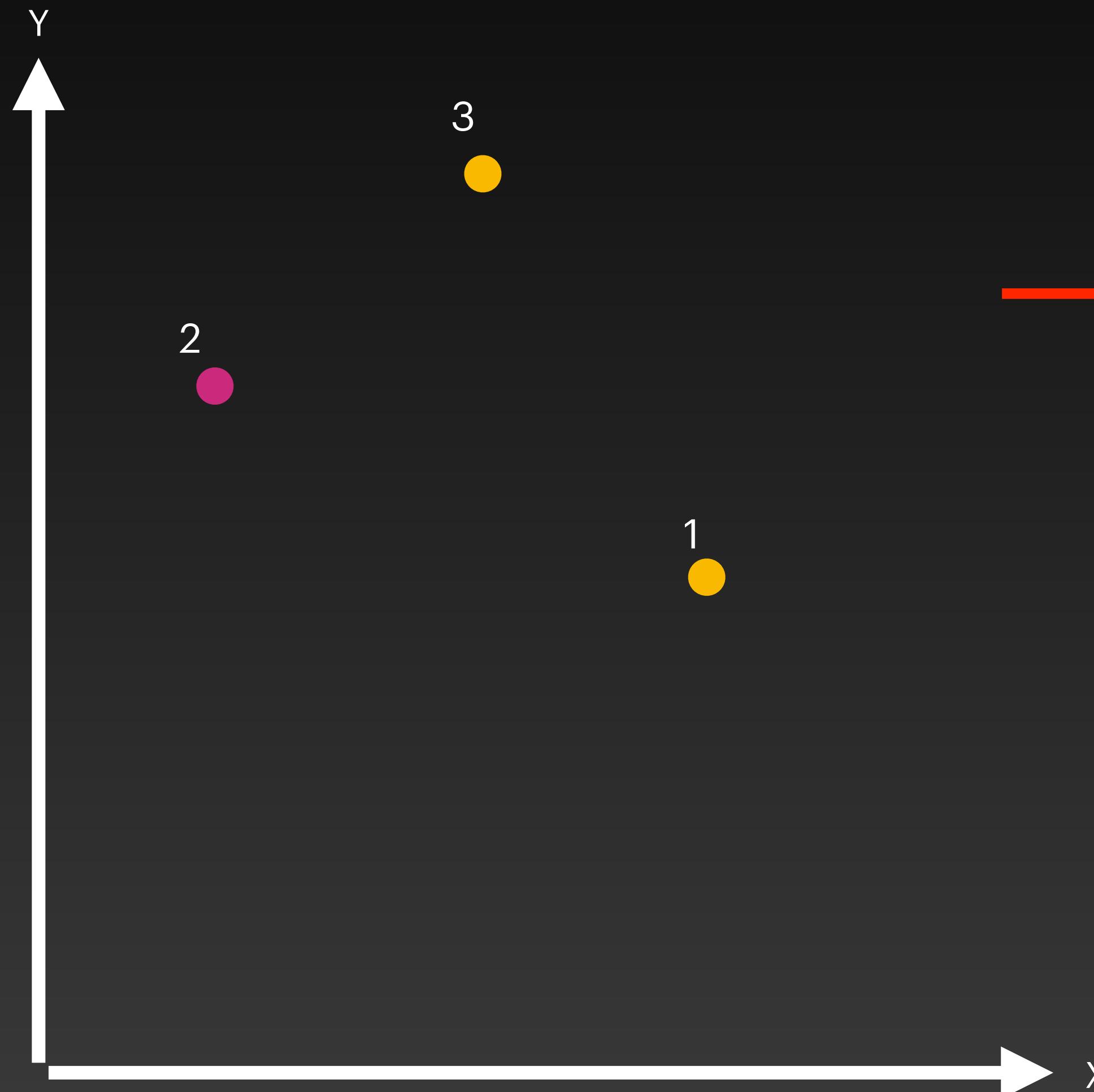


```
senão
    No qtAux = qt;      ✓ + ✓ = ✓ -> ✗
    enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
        pai = qtAux;
        quadrante = GetQuadrante(ponto, qtAux.ponto);
        qtAux = GetFilho(qtAux, quadrante);
    fim-enquanto
```

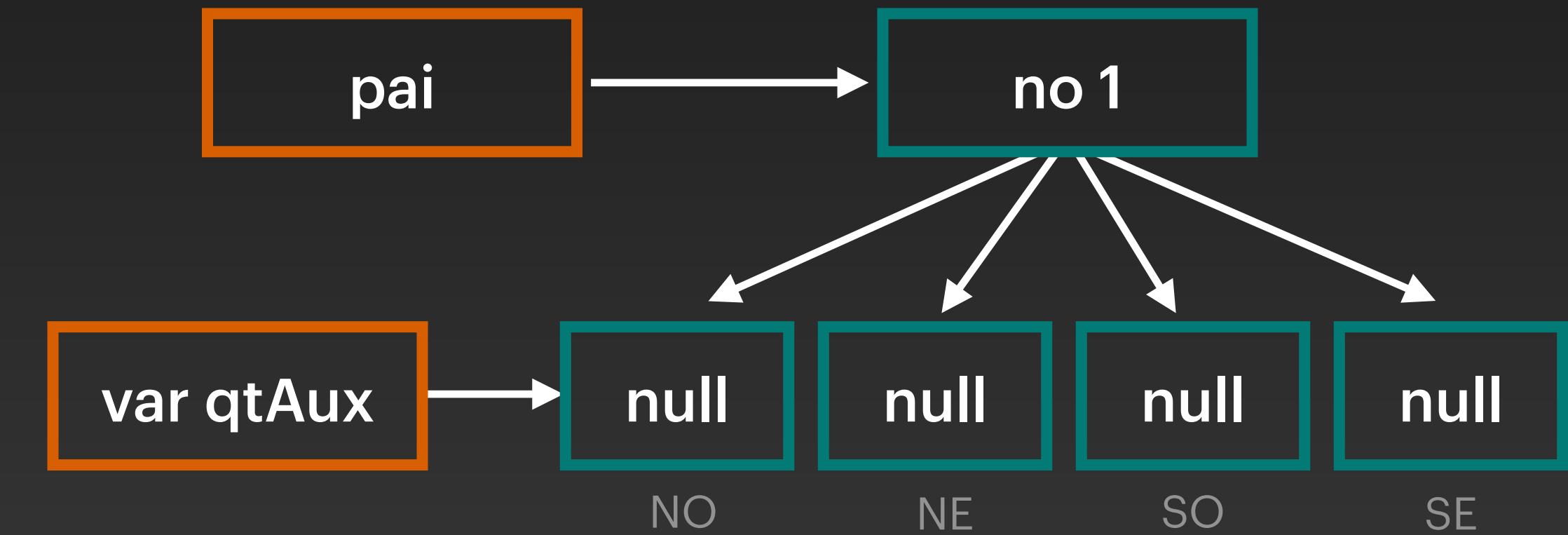


Point QuadTree

Inserção

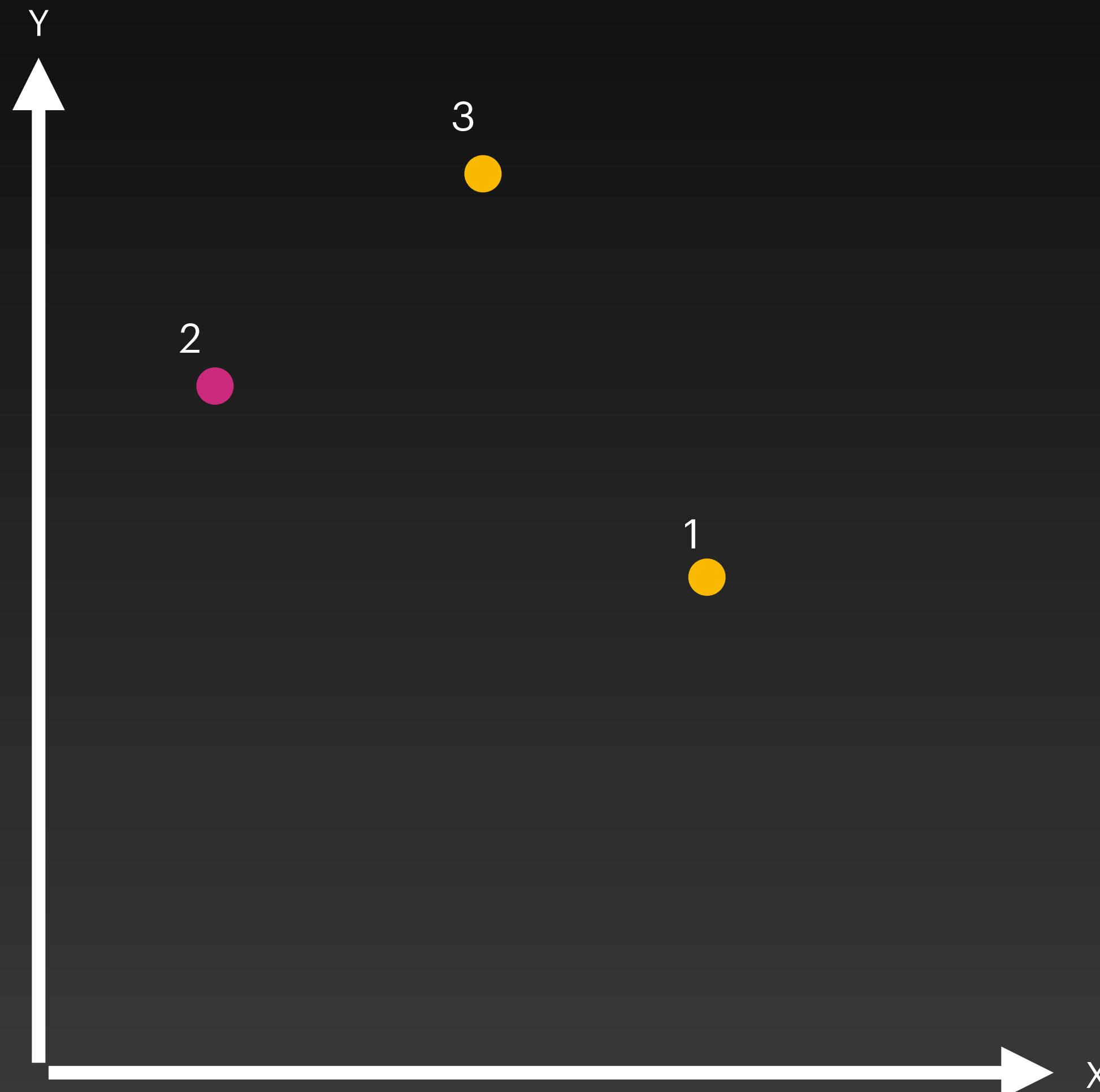


```
senão
    No qtAux = qt;      ✓ + ✓ = ✓ -> ✗
    enquanto not(qtAux == null e not(VerificaCoordenadasIguais(ponto, qtAux.ponto))
        pai = qtAux;
        quadrante = GetQuadrante(ponto, qtAux.ponto);
        qtAux = GetFilho(qtAux, quadrante);
    fim-enquanto
```

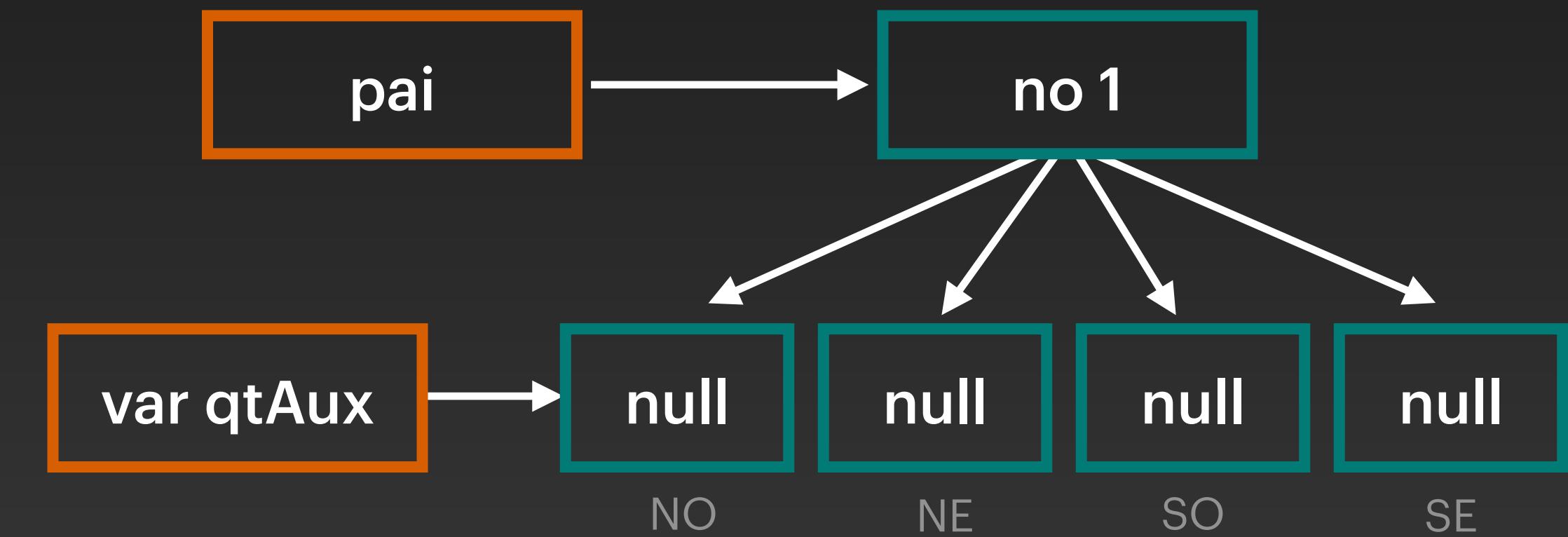


Point QuadTree

Inserção

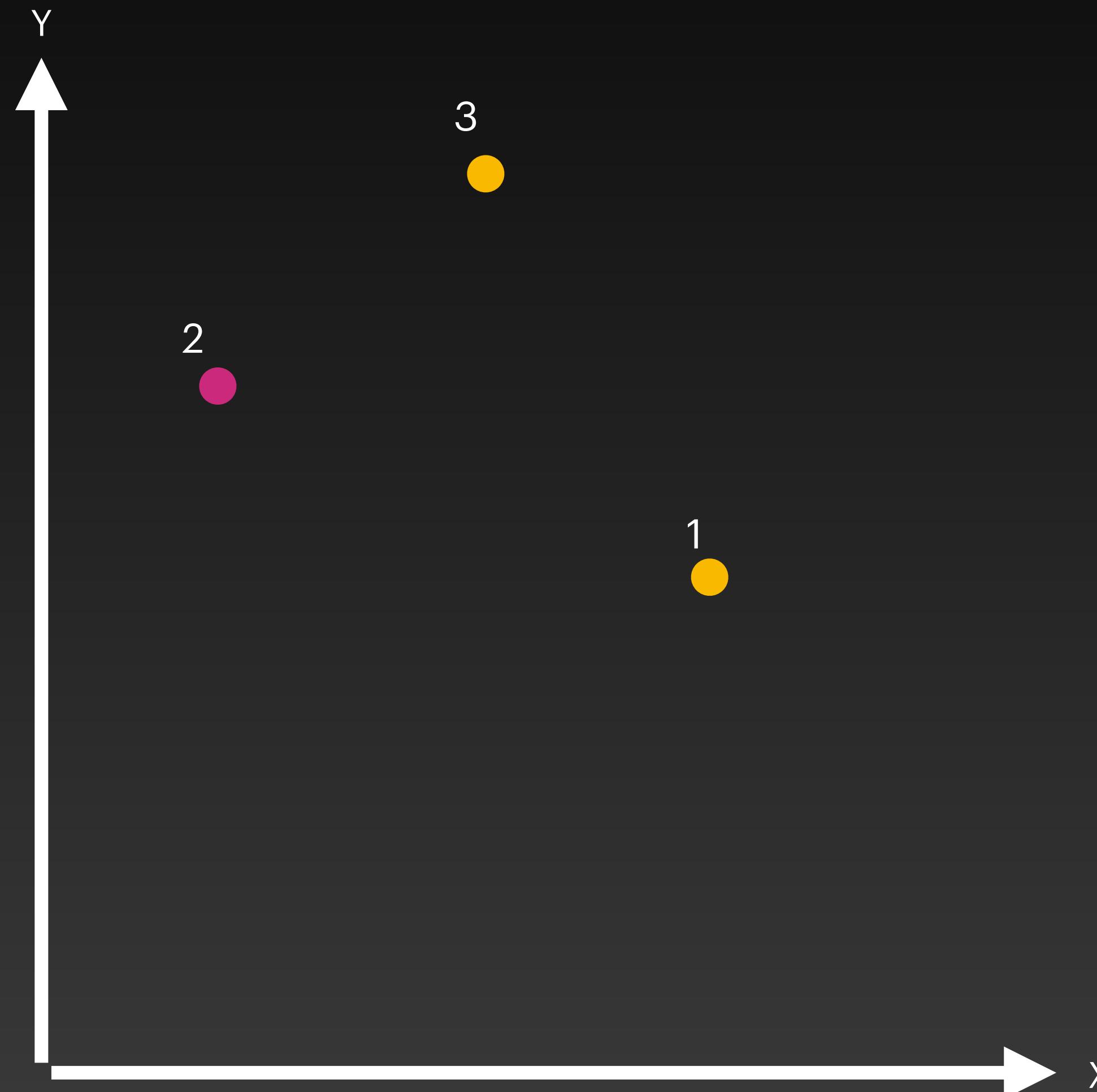


```
se qtAux == null então
    filho = CriaNo();
    filho.ponto = ponto;
    filho.info = pInfo;
    SetFilho(pai, quadrante);
fim-se
```

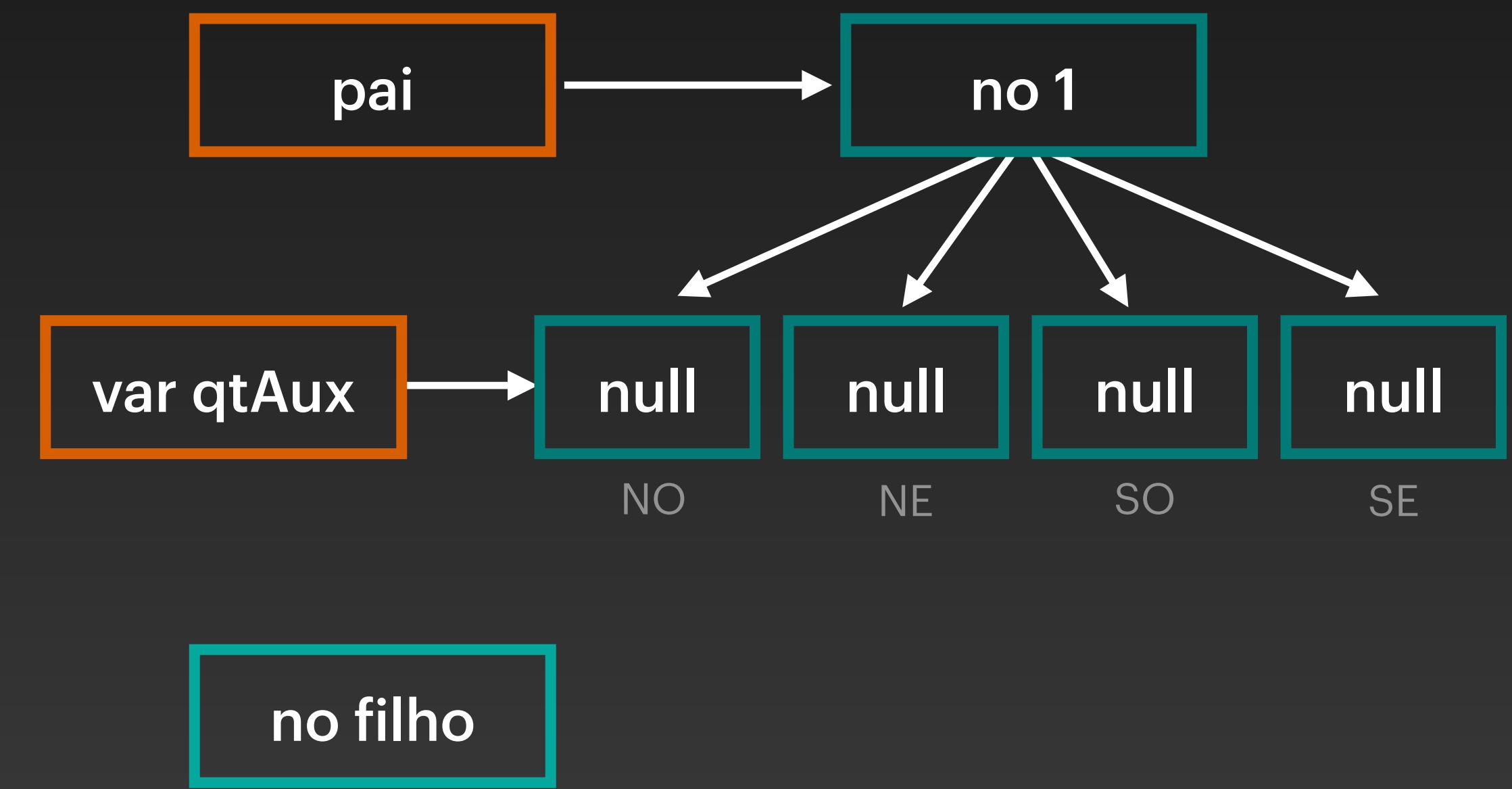


Point QuadTree

Inserção

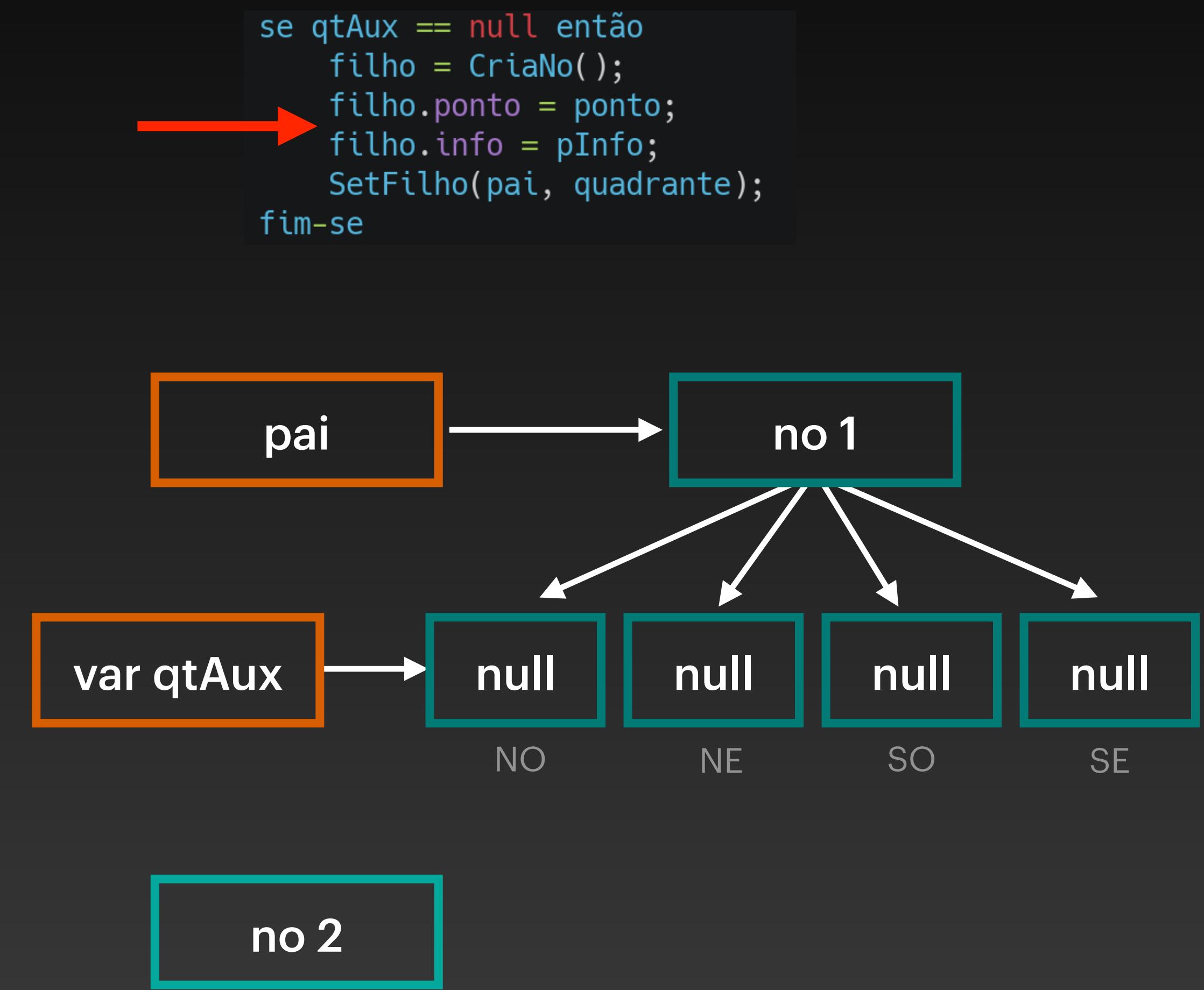
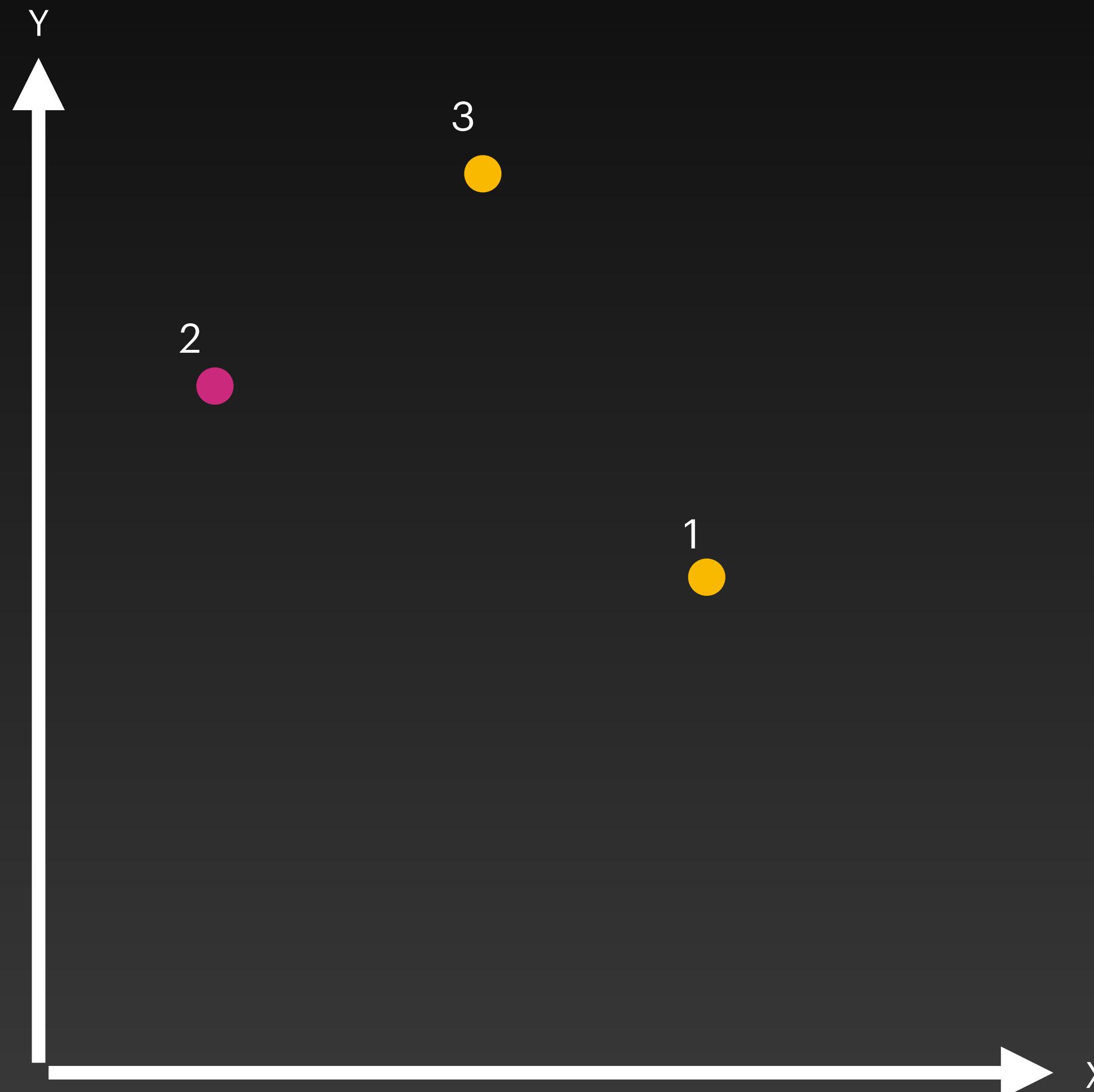


```
se qtAux == null então
    filho = CriaNo();
    filho.ponto = ponto;
    filho.info = pInfo;
    SetFilho(pai, quadrante);
fim-se
```



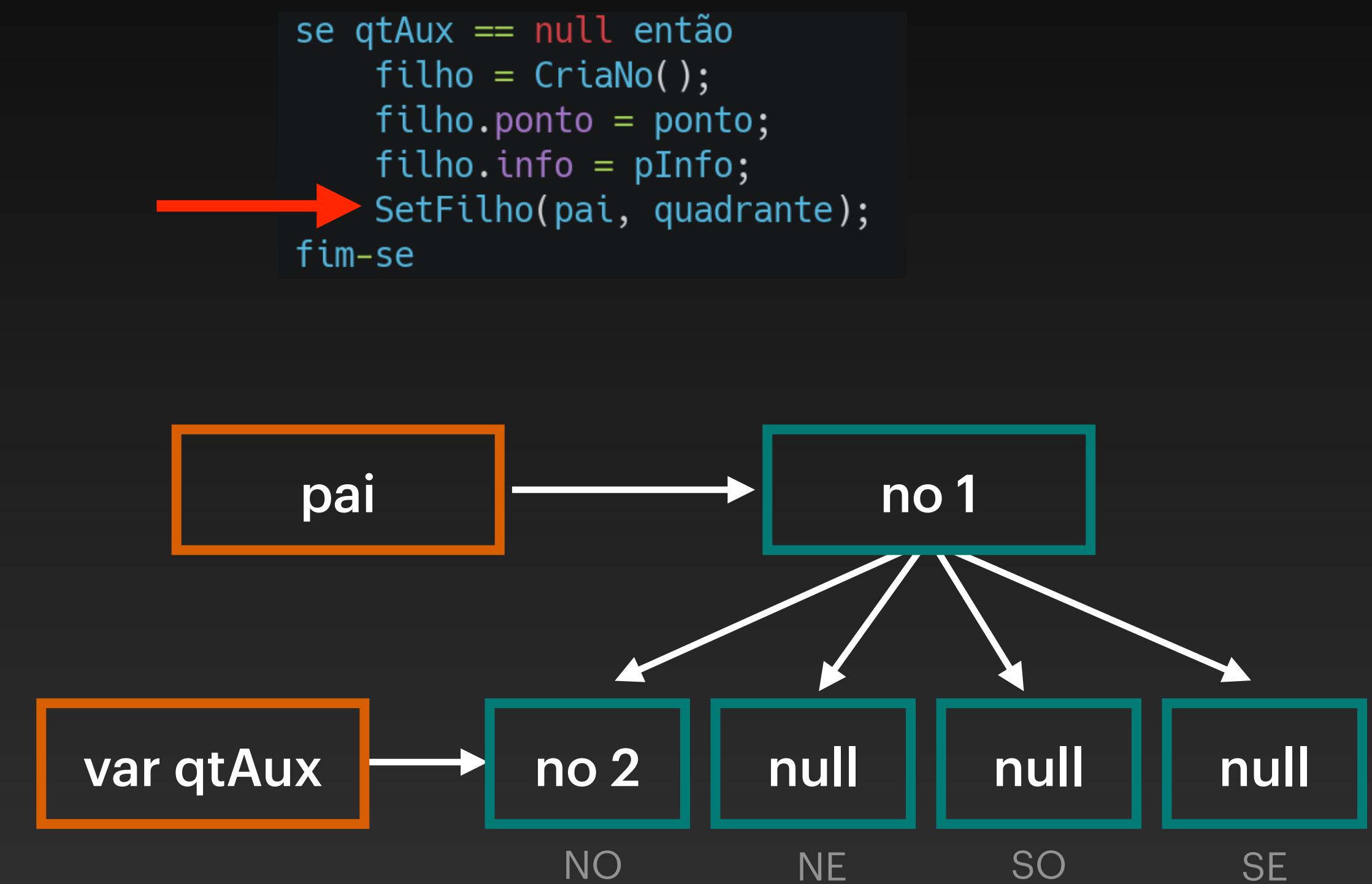
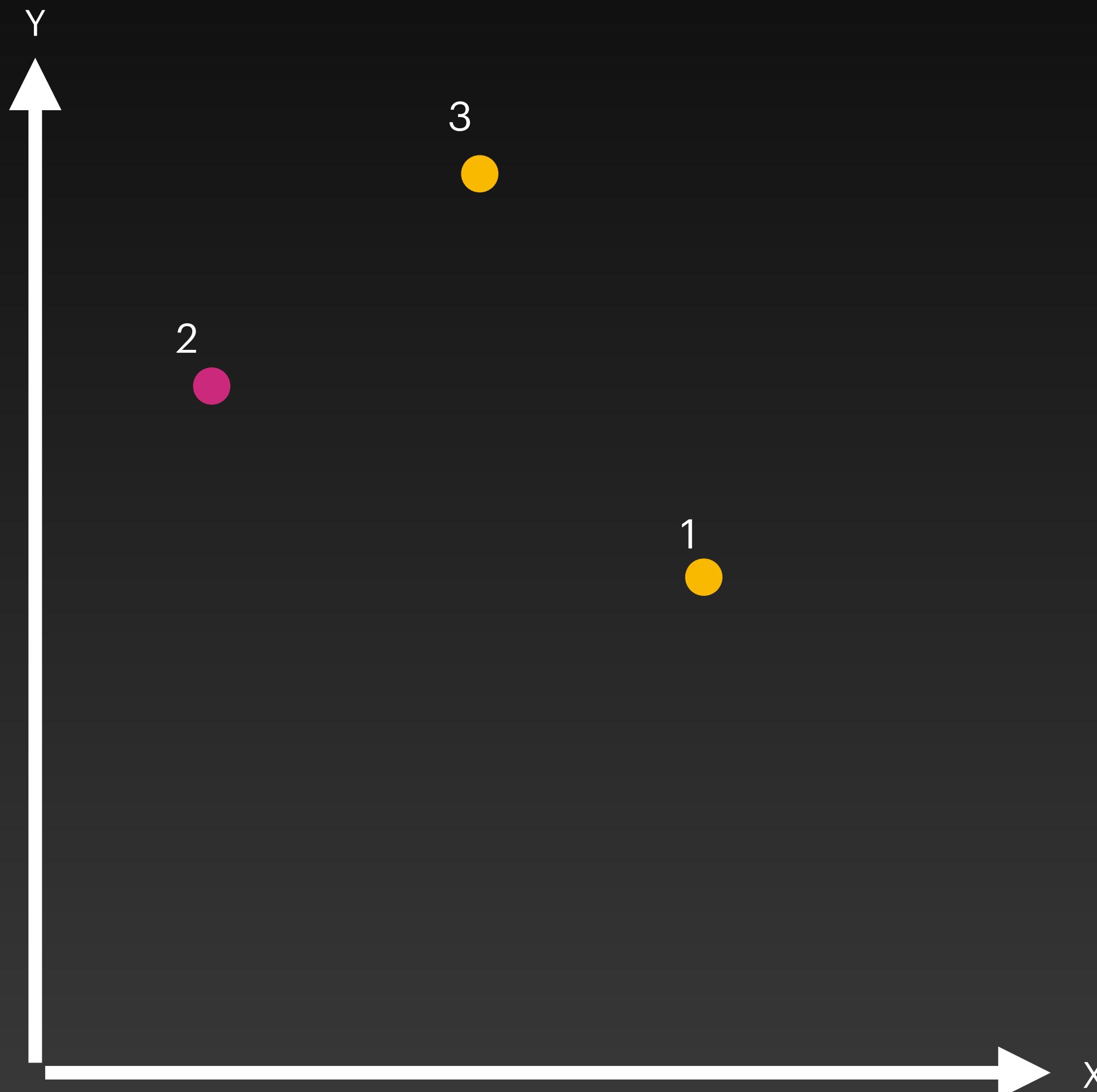
Point QuadTree

Inserção



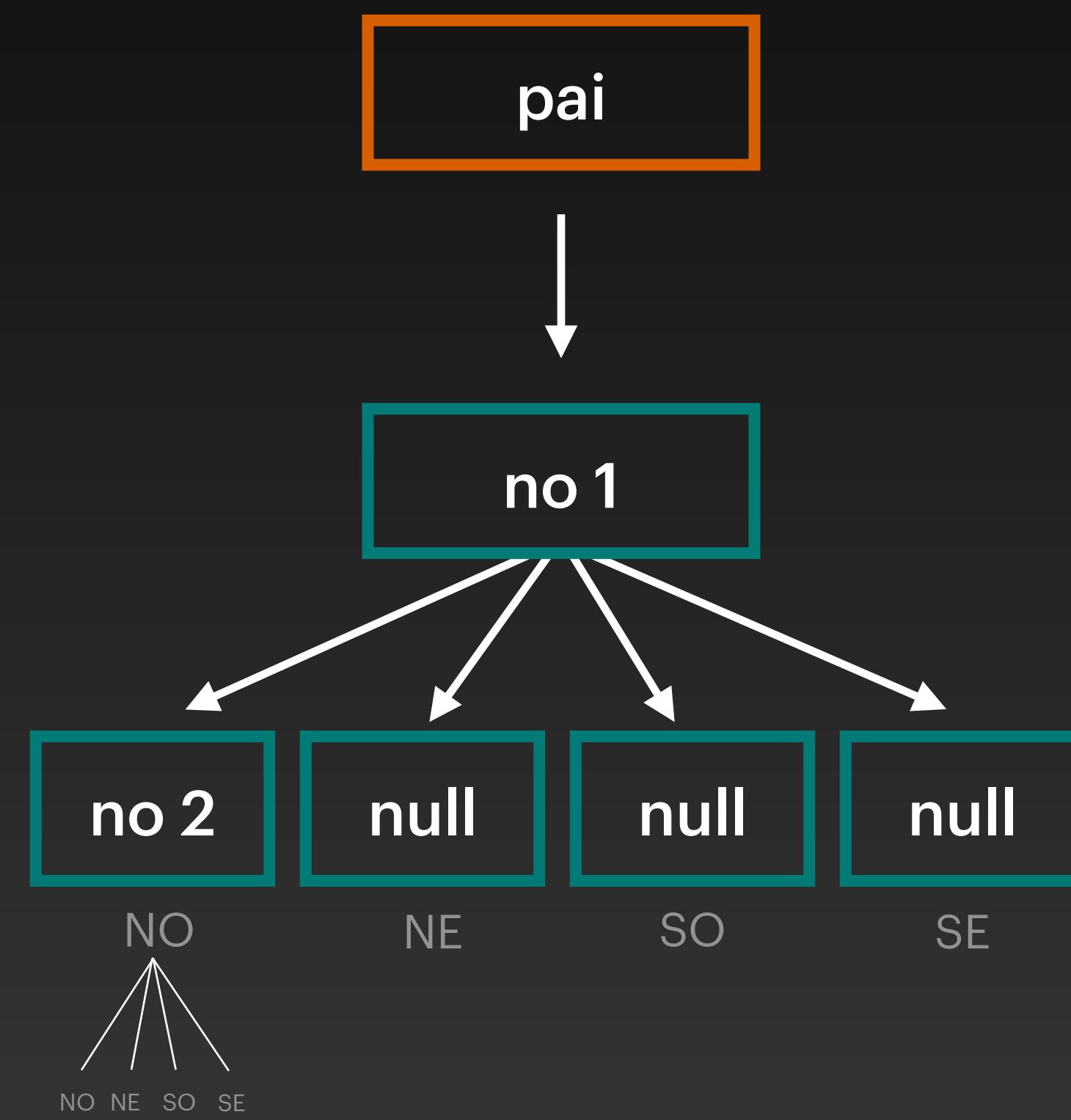
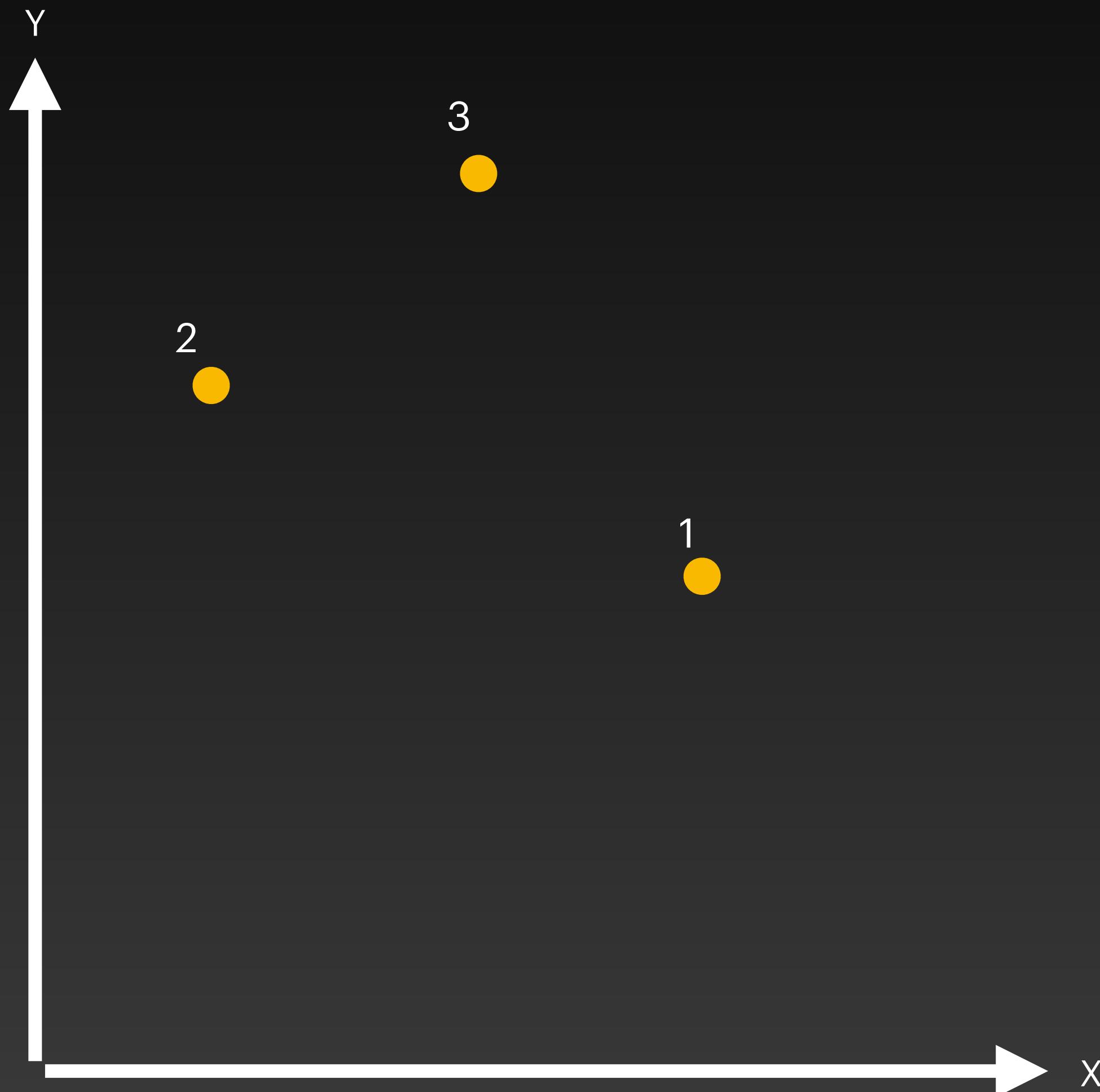
Point QuadTree

Inserção



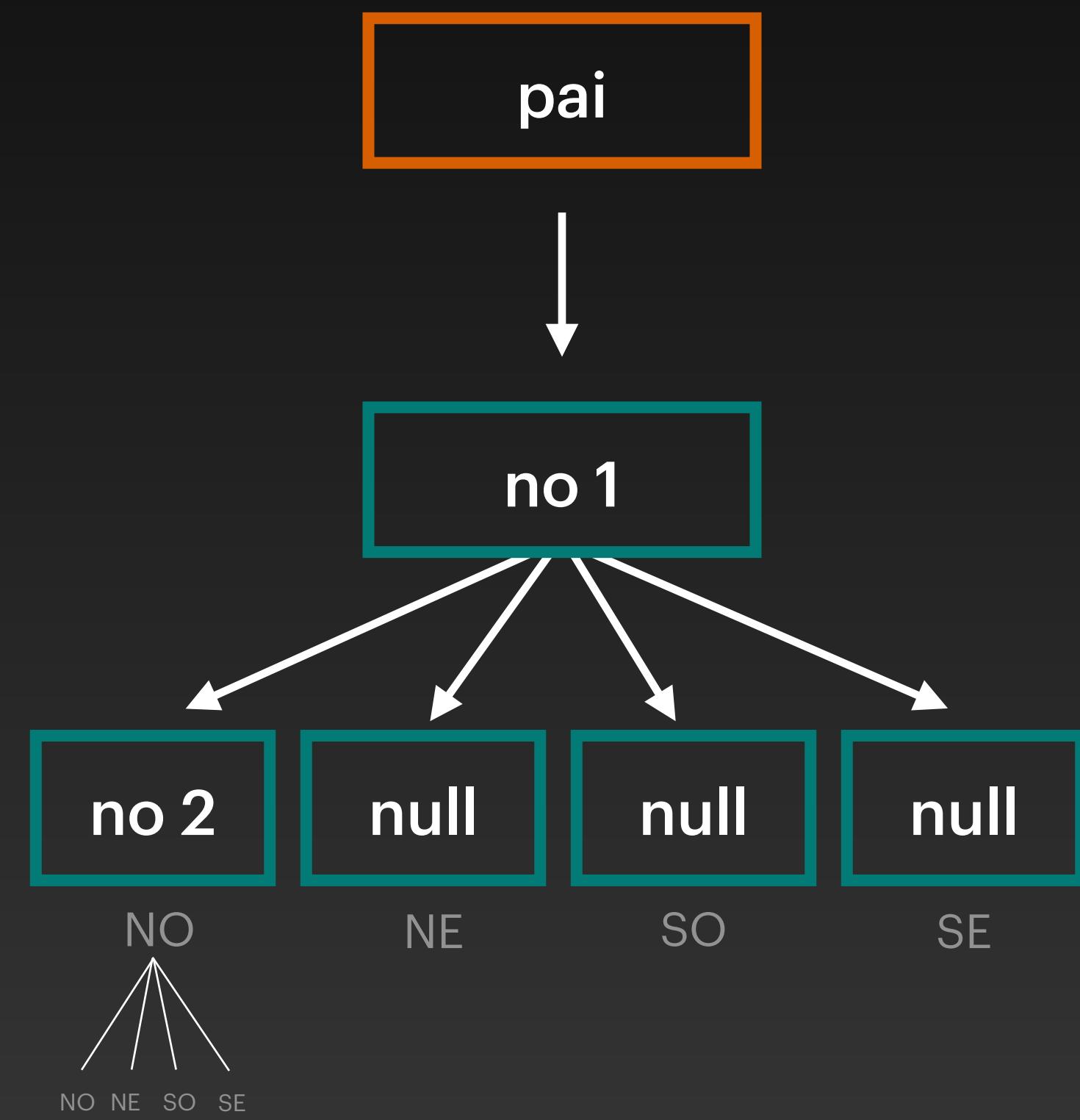
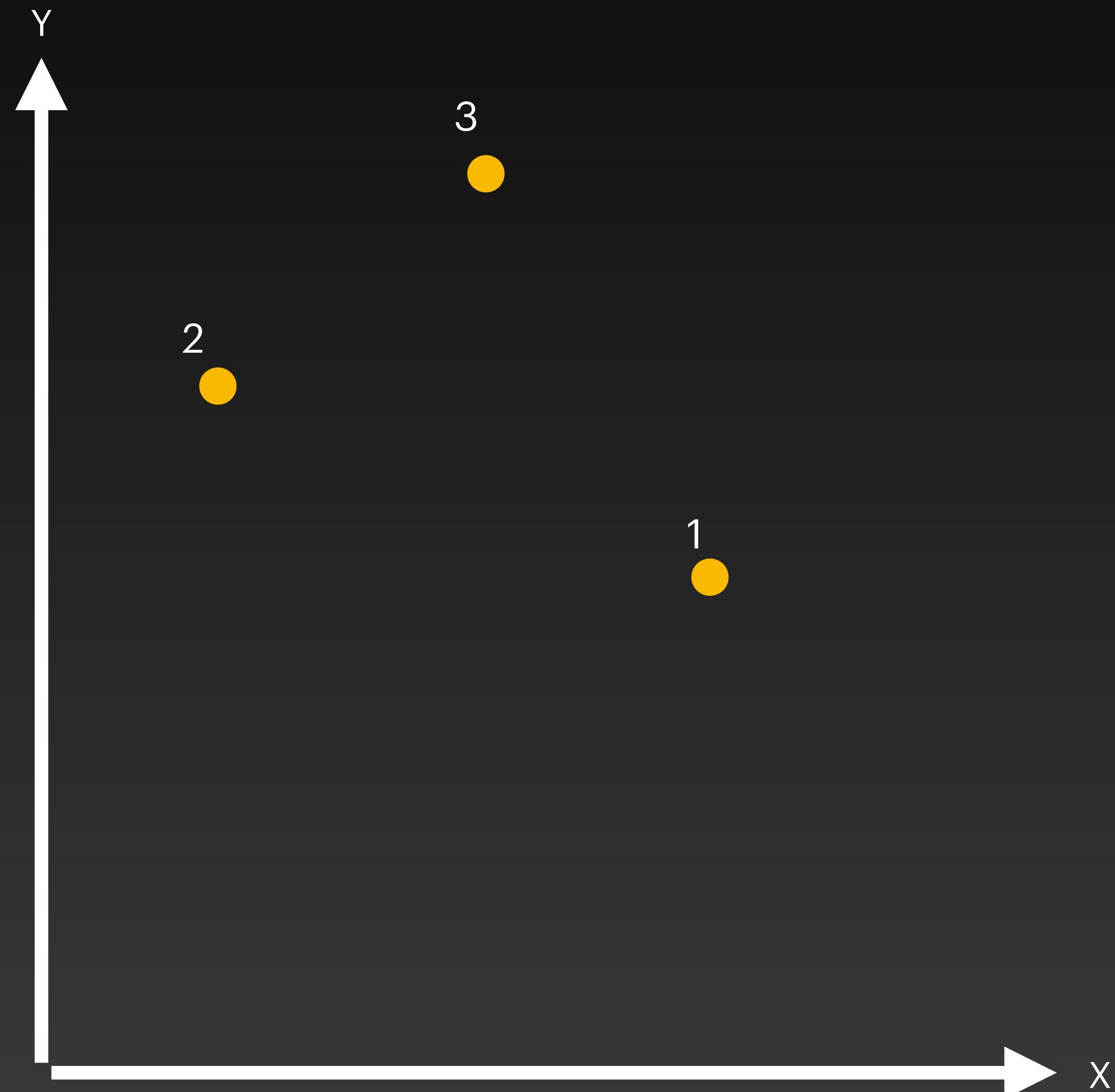
Point QuadTree

Inserção



Point QuadTree

Inserção



Para treinar: faça o passo a passo para inserir o nó 3

Point QuadTree

Busca

- Usa o conceito de **busca por profundidade**, através da recursão.
- Minimiza o tempo de busca ao buscar o nó relativo a sua posição (x,y).

Point QuadTree

Busca



```
GetNoQt(qt, x, y)
começo

    se qt == null
        retorne null;
    senão
        se qt.ponto.x == x e qt.ponto.y == y então
            retorne qt.info;
        senão
            Quadrante quadrante;
            Ponto ponto;
            No filho;

            ponto.x = x;
            ponto.y = y;

            quadrante = GetQuadrante(ponto, qt.ponto);

            filho = GetFilho(qt, quadrante);

            retorne GetNoQt(filho, x, y);
        fim-se
    fim-se

fim
```

Point QuadTree

Remoção

- Remover um nó de uma árvore geralmente é uma das tarefas mais difíceis.
- A remoção sempre tem que manter a **integridade** da árvore, tanto em relação a informação, como também ao balanceamento.
- **O que acontece o nó de uma QuadTreee for removido?**

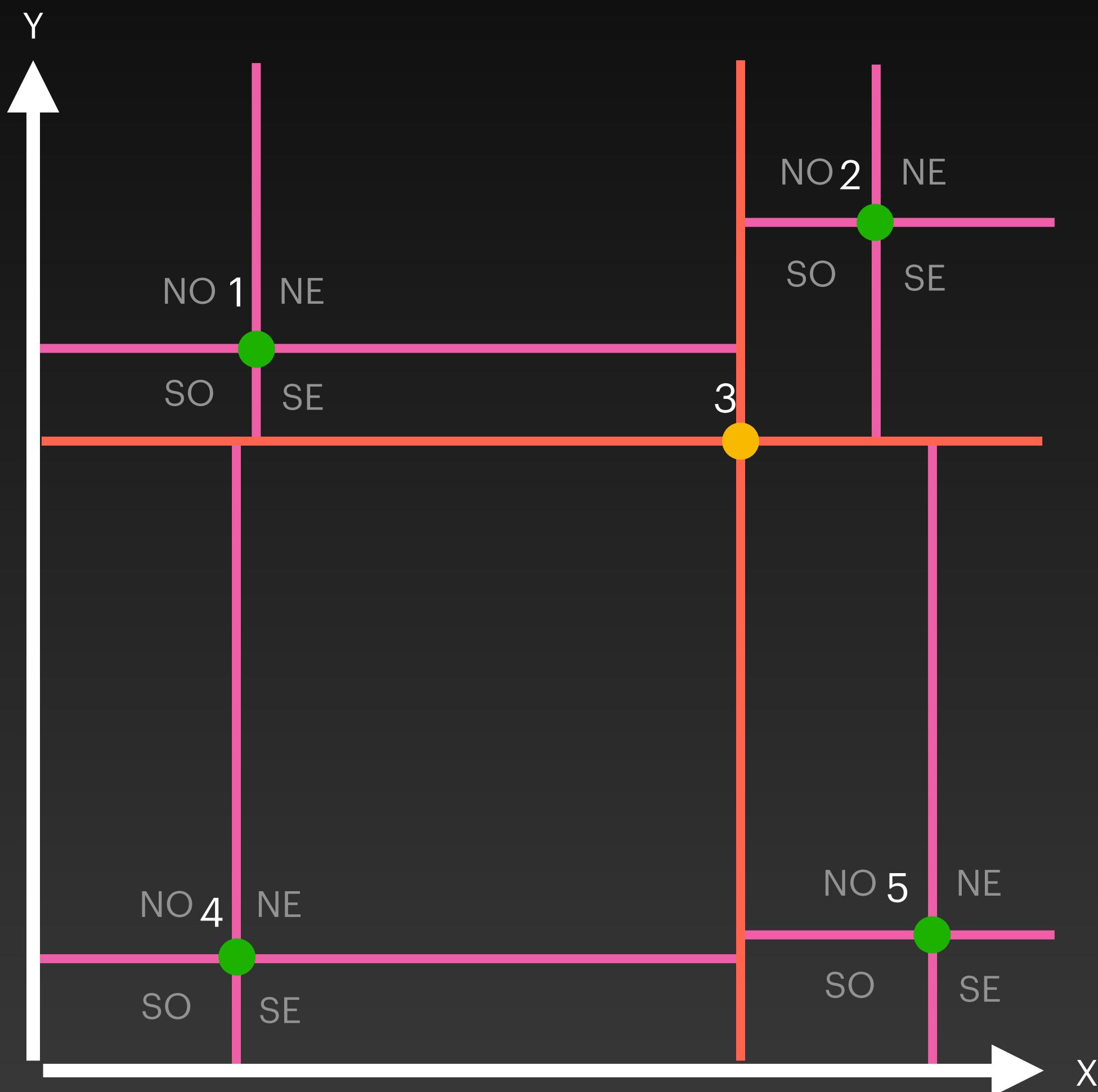
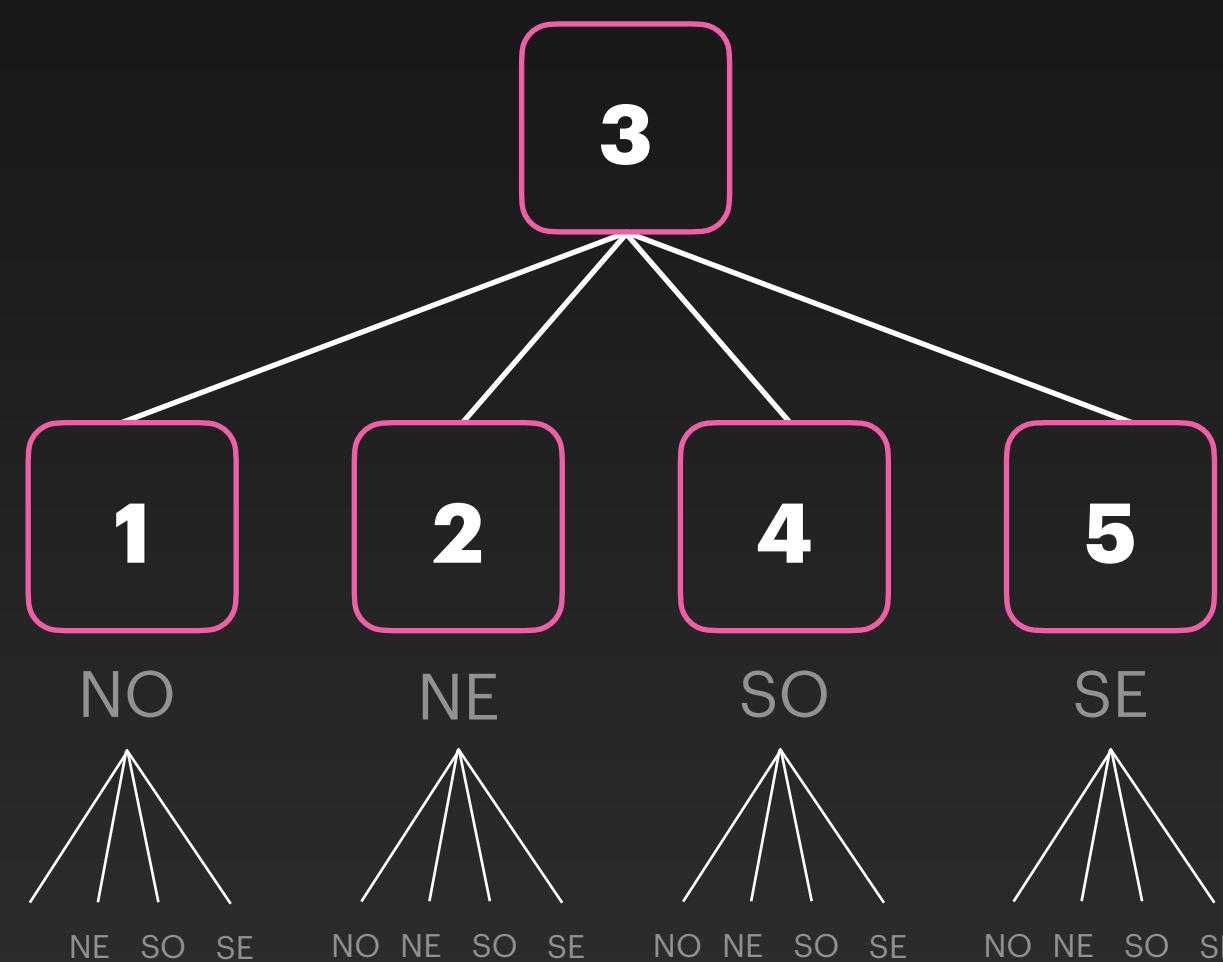
Point QuadTree

Remoção

- Remover um nó de uma árvore geralmente é uma das tarefas mais difíceis.
- A remoção sempre tem que manter a **integridade** da árvore, tanto em relação a informação, como também ao balanceamento.
- **O que acontece o nó de uma QuadTreee for removido?**

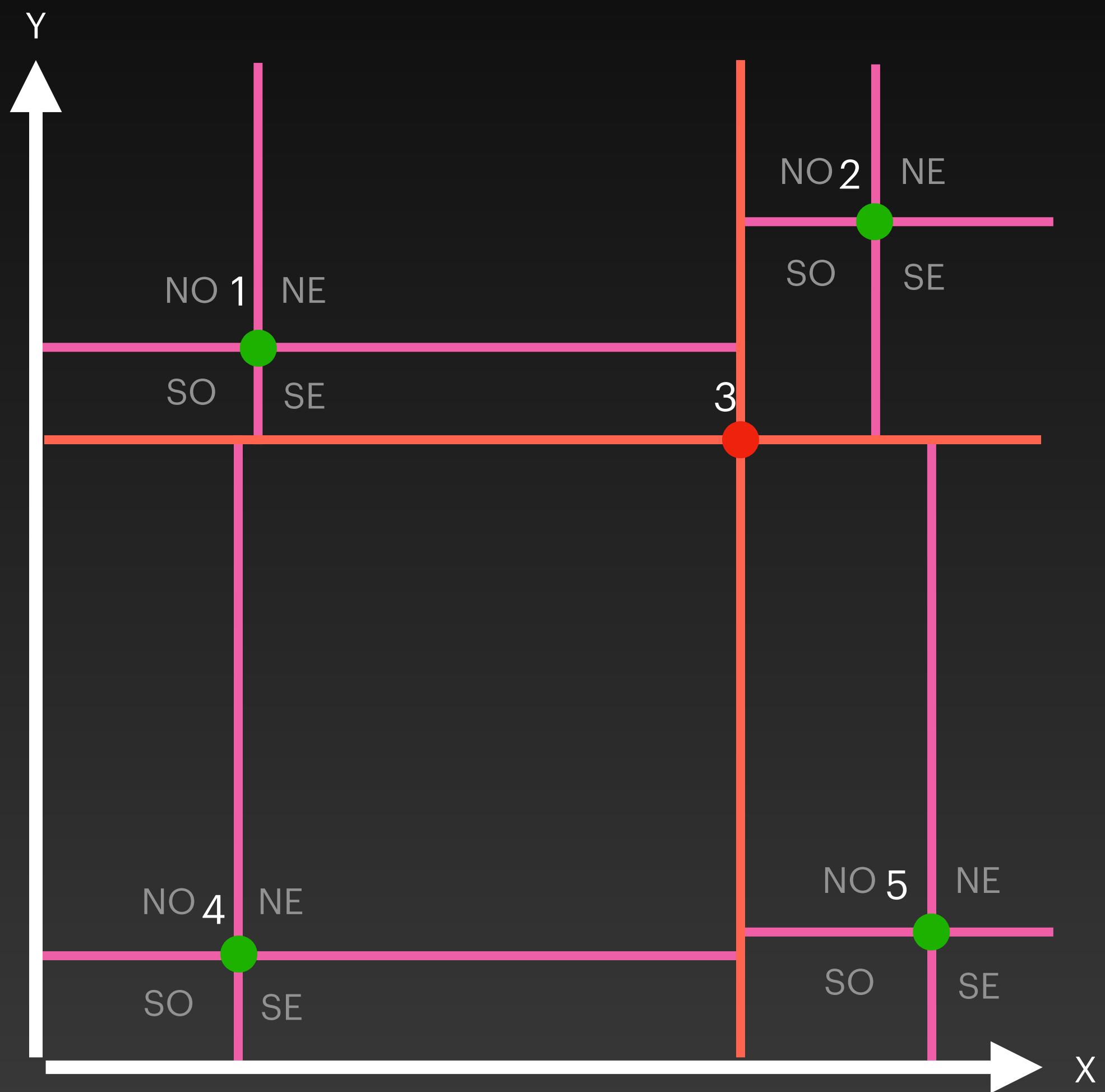
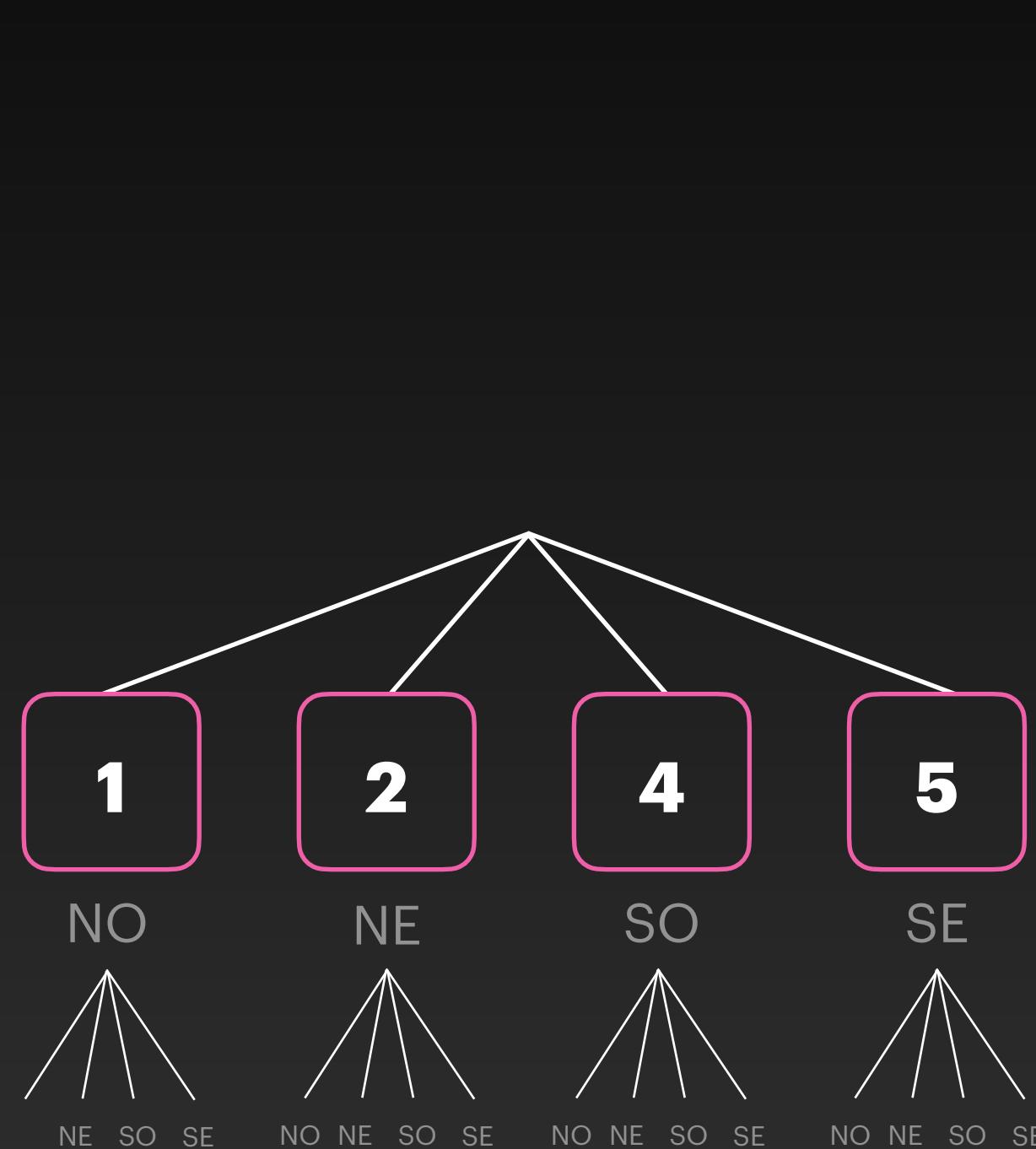
Point QuadTree

Remoção



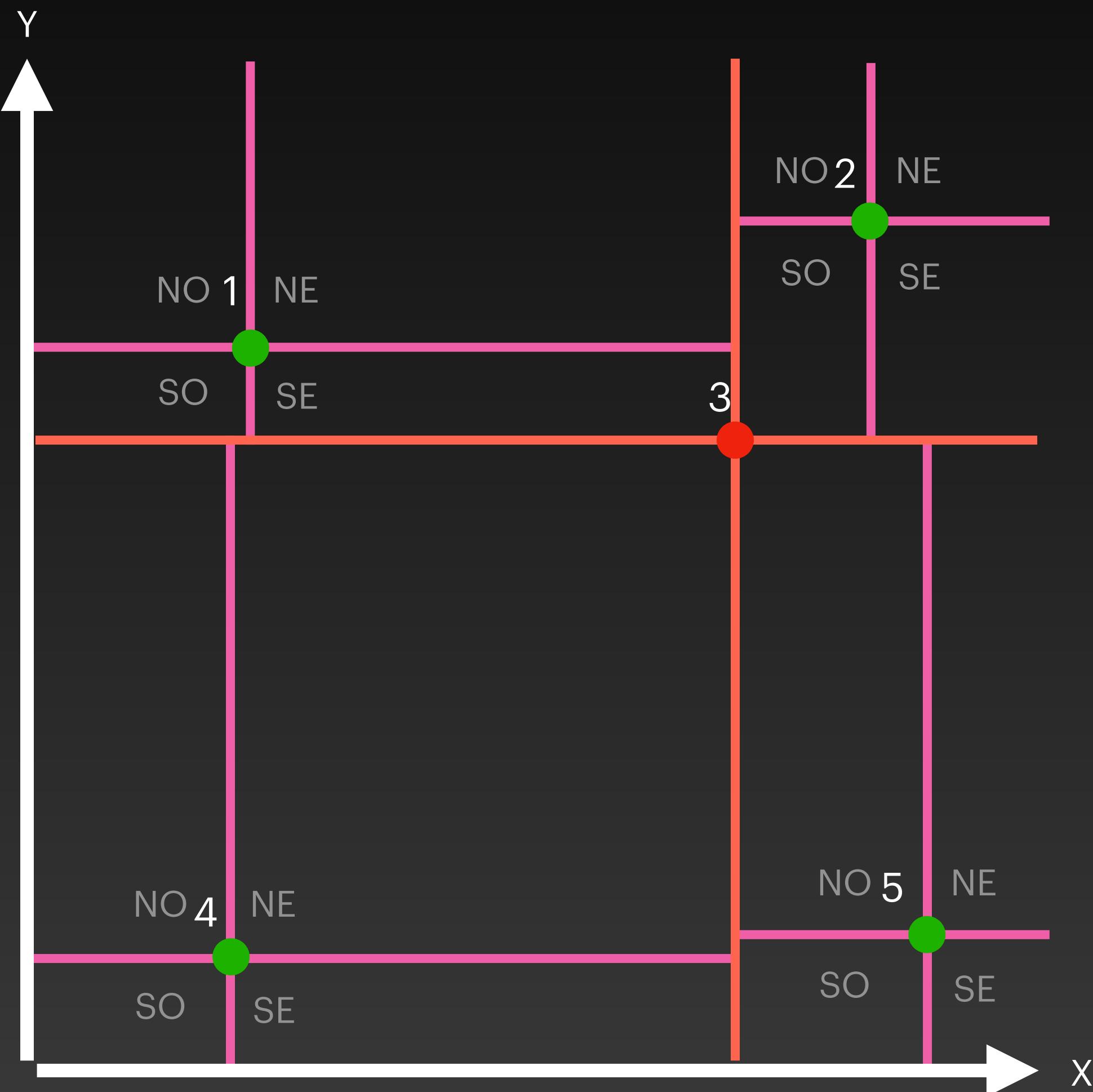
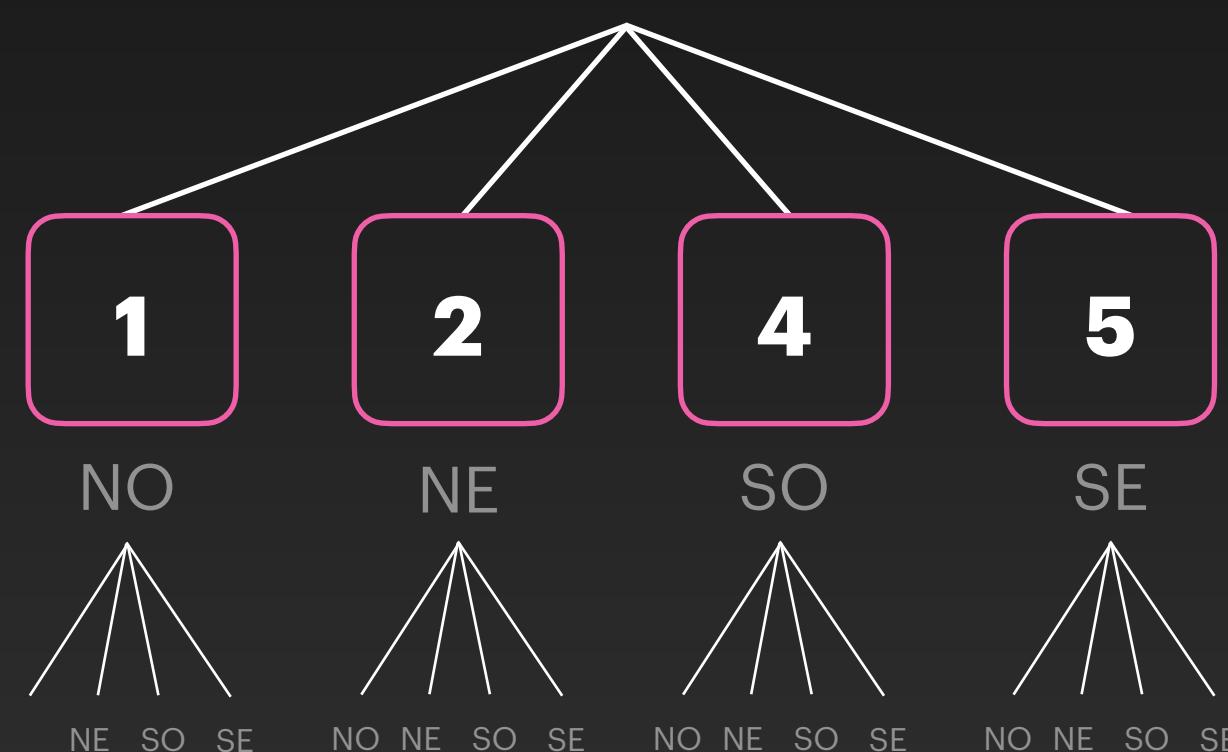
Point QuadTree

Remoção



Point QuadTree

Remoção



- Qual é a nova raiz da árvore?
- Qual é o novo ponto central do espaço bidimensional?

Point QuadTree

Remoção

- Os nós de uma quadtree estão **diretamente ligados** a sua posição no espaço, e ao nó anterior que ele está conectado.
- Remover um ponto significa remover a divisão de quadrante montada.
- Sendo assim, após remover um ponto, é necessário remontar **toda a árvore a partir daquele ponto deletado!**

Point QuadTree

Remoção

- Em termos de processamento, isso é extremamente **custoso**.
- Para isso, a maneira mais ideal é ter uma variável em cada ponto que informa se ele foi **deletado ou não**.
- Dessa maneira, o ponto continua na árvore, porém, ao ser buscado/ processado, o código verifica primeiramente se ele foi deletado.
- Caso verdadeiro, nenhuma ação é feita com ele.

A photograph of a dark night sky filled with stars. A bright, green aurora borealis (Northern Lights) arches across the upper portion of the frame, its light glowing against the dark background. In the foreground, the dark silhouettes of mountain peaks are visible, their rugged textures catching some of the ambient light.

That's all!