ÁRVORE BINÁRIA DE BUSCA

Objetivo

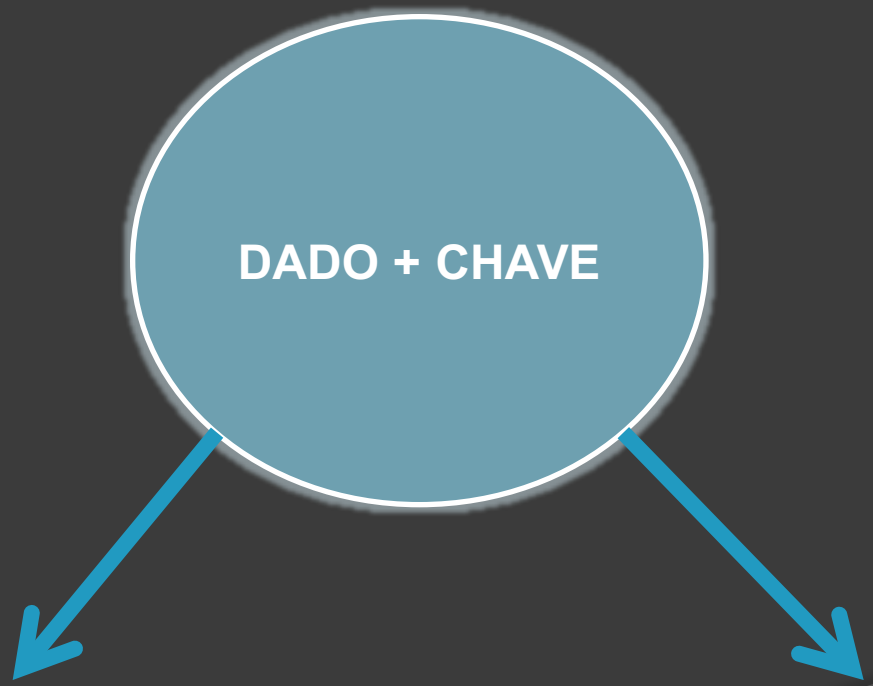
- Compreender o funcionamento de uma árvore binária de busca

Árvore Binária de Busca

- É uma estrutura de dados que foi criada com o intuito de otimizar o tempo de busca, em comparação às listas.
- Todo nó de uma árvore binária de busca possui um dado.
- Neste dado, deve haver um índice (ou chave), que deverá ser utilizado como base para as comparações.
- IMPORTANTE: não podem haver dois nós com a mesma chave

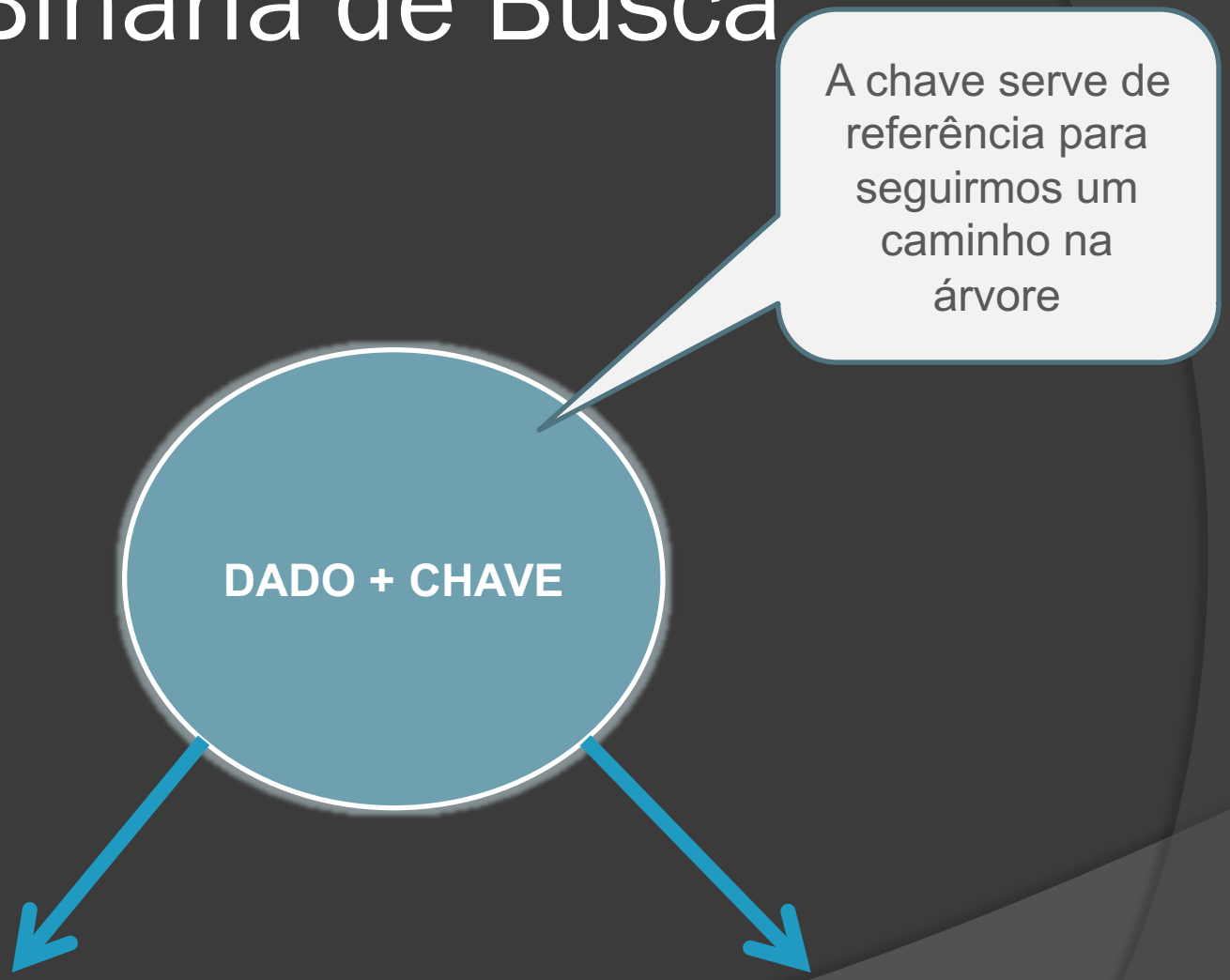
Árvore Binária de Busca

- Vejamos



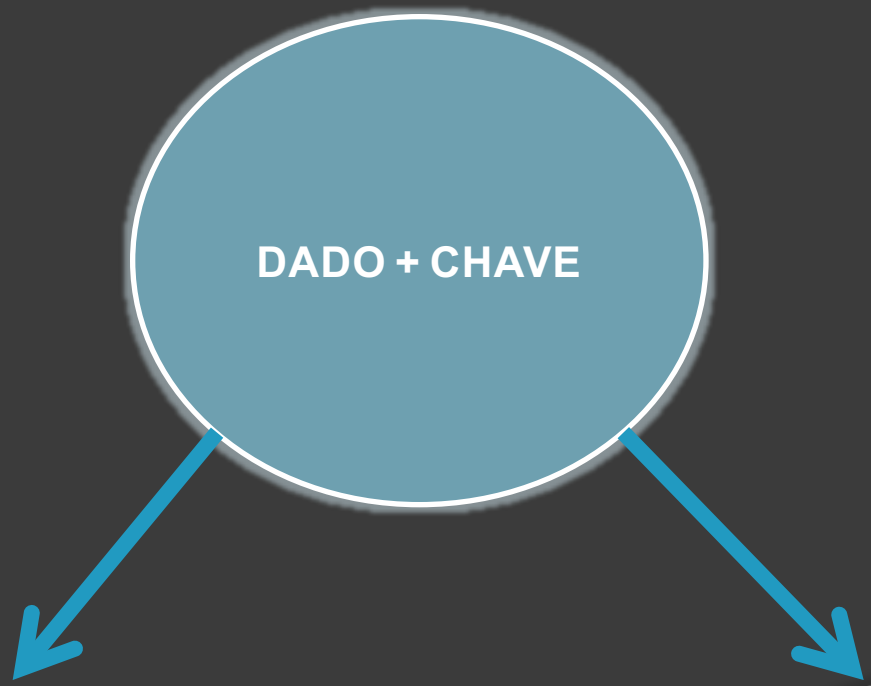
Árvore Binária de Busca

- Vejamos



Árvore Binária de Busca

- Vejamos



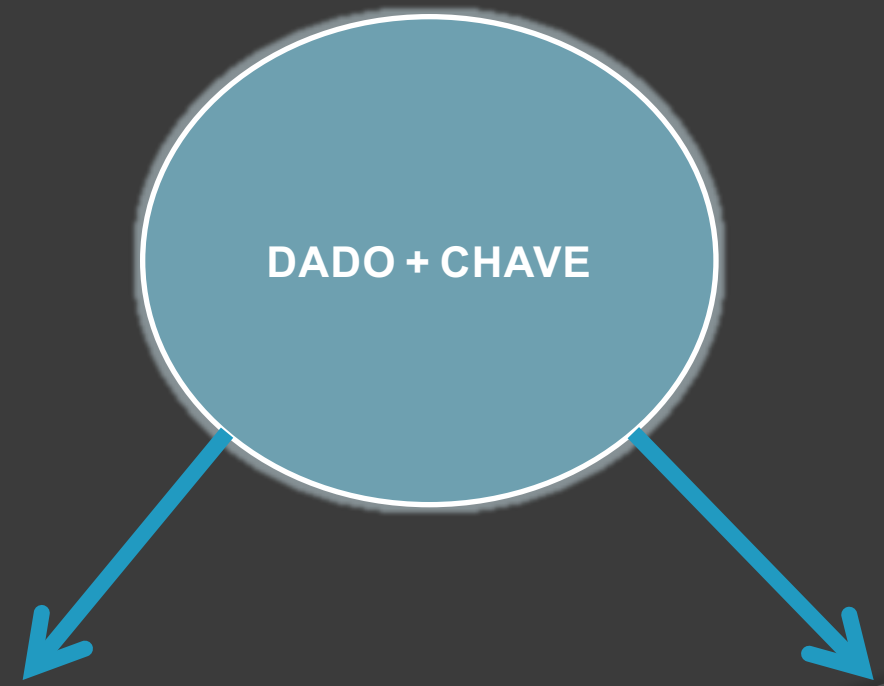
Se a chave que procuramos for maior que a armazenada no nó, deve-se pegar o caminho da direita

MAIOR (>)

Árvore Binária de Busca

- Vejamos

Se a chave que procuramos for menor que a armazenada no nó, deve-se pegar o caminho da esquerda



(<) MENOR

MAIOR (>)

Árvore Binária de Busca

E SE FOR IGUAL?

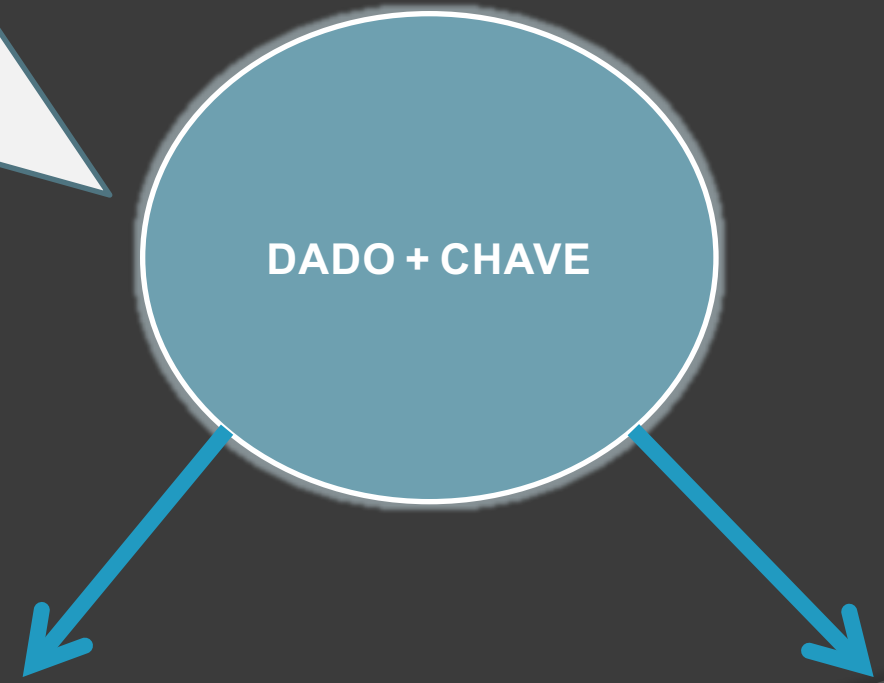
(<) MENOR

MAIOR (>)

Árvore Binária de Busca

- Vejamos

Se for igual, significa que encontramos o nó que estávamos procurando!



(<) MENOR

MAIOR (>)

Exemplo de Busca

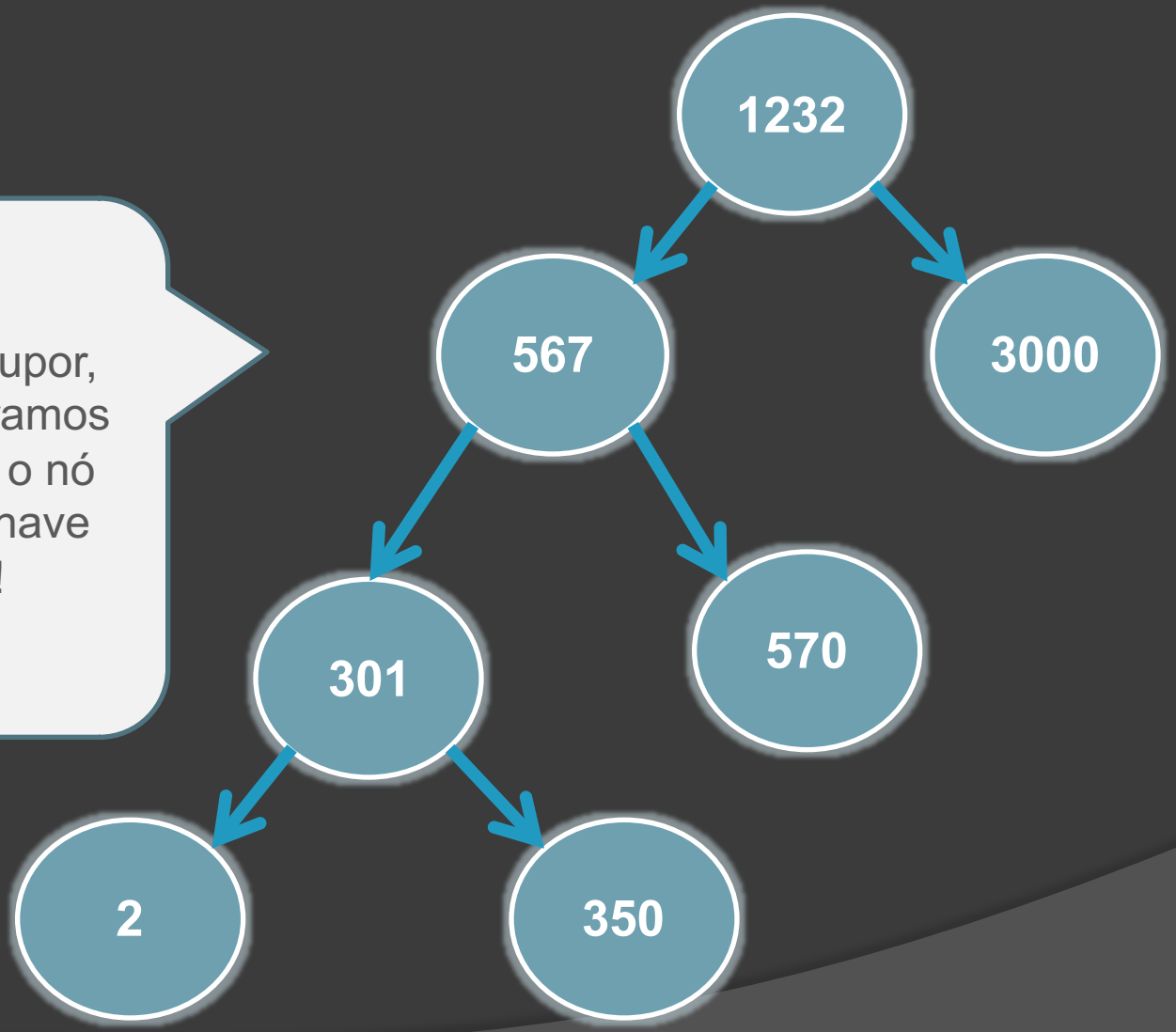
Iremos realizar dois exemplos de busca de valores para ilustrar processo.

2

350

Exemplo de Busca

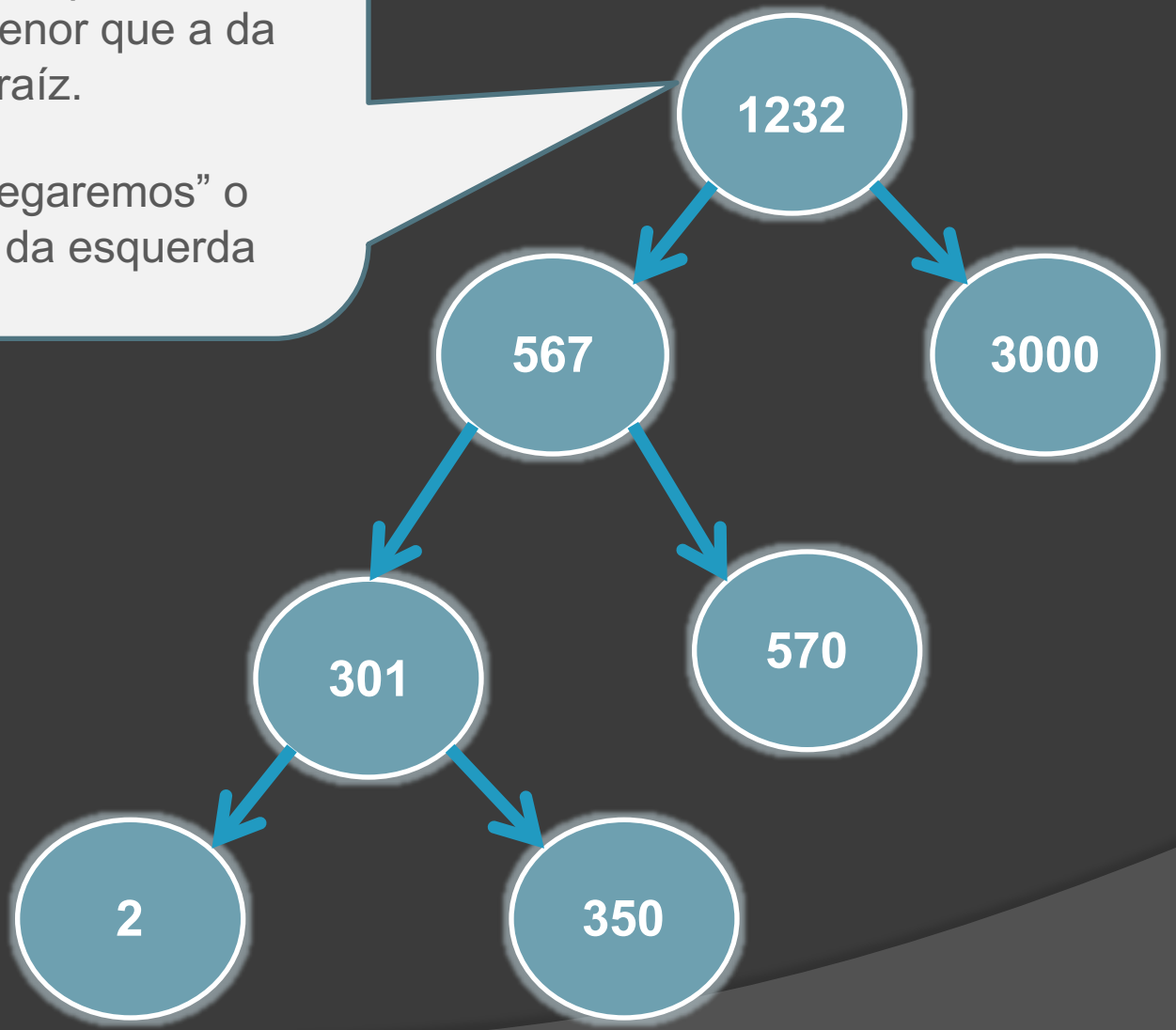
Vamos supor, que queiramos localizar o nó com a chave 570!



Busca

Iniciando as comparações a partir da raíz, constatamos que a chave procurada (570) é menor que a da raíz.

Logo, “pegaremos” o caminho da esquerda



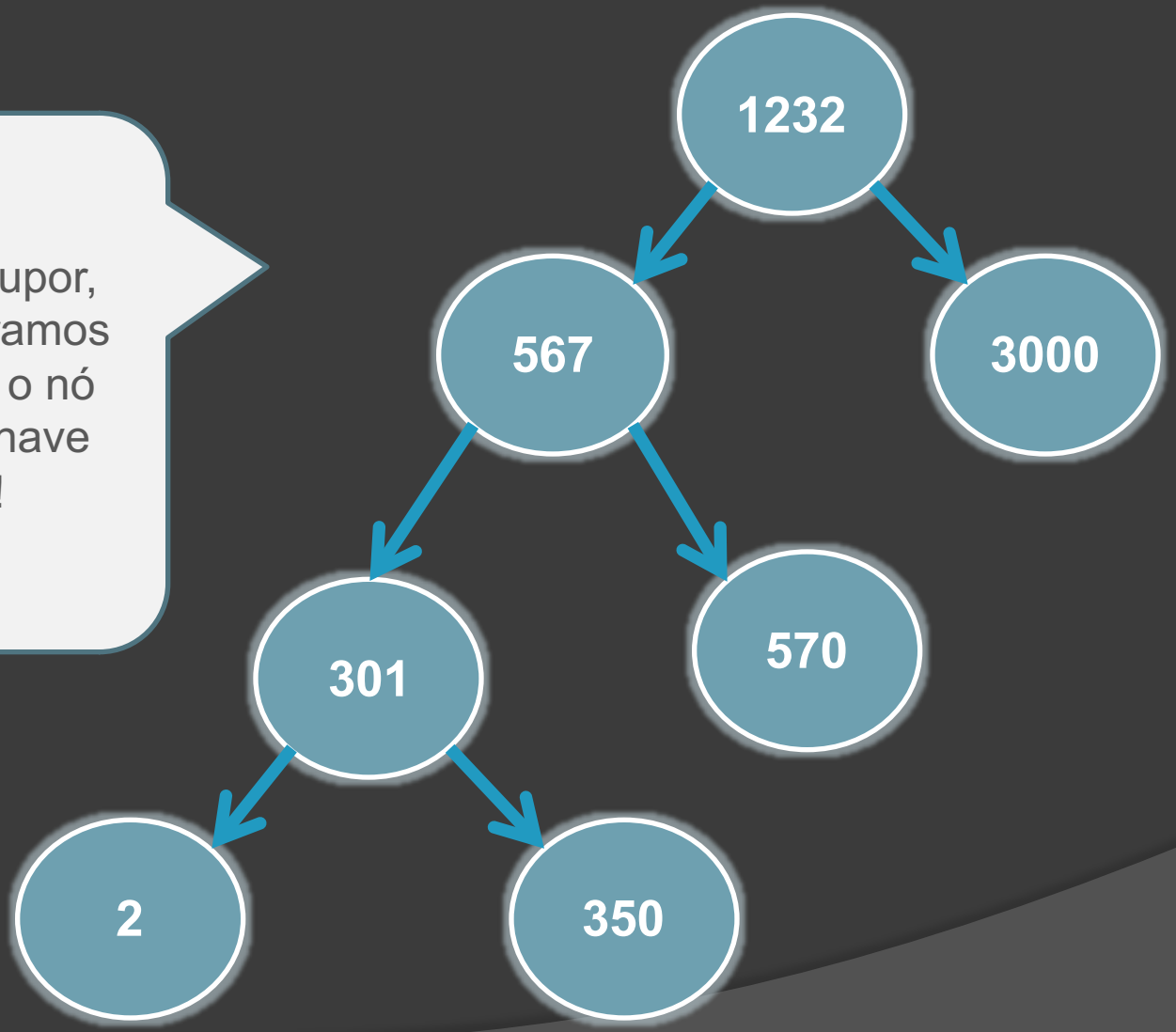
Busca

Analisando o nó recém visitado, percebe-se que a chave (567) do mesmo é menor do que a procurada. Logo, deve-se utilizar o caminho da direita

Exemplo de Busca

Exemplo de Busca

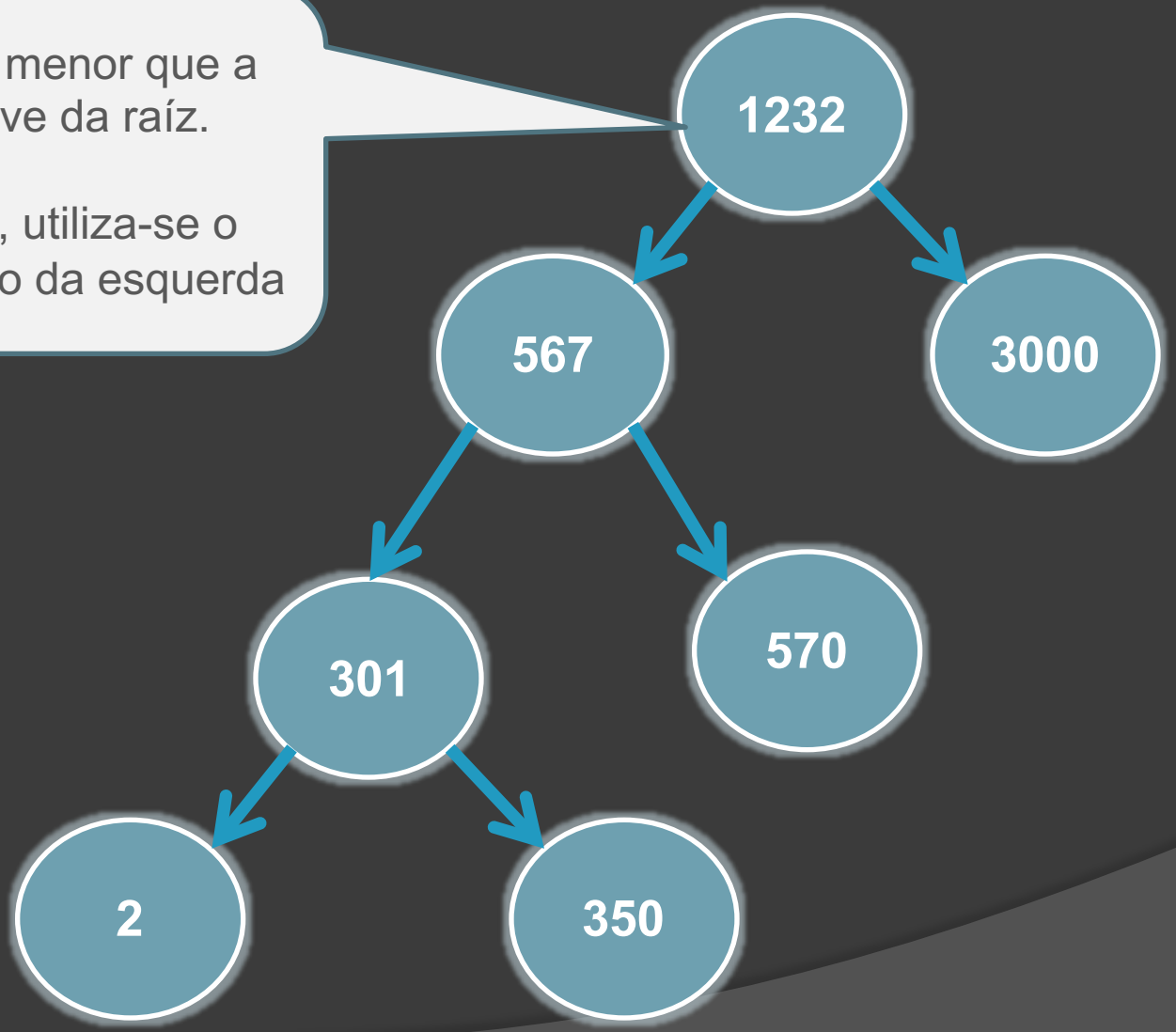
Vamos supor, que queiramos localizar o nó com a chave 351!



Exemplo de Busca

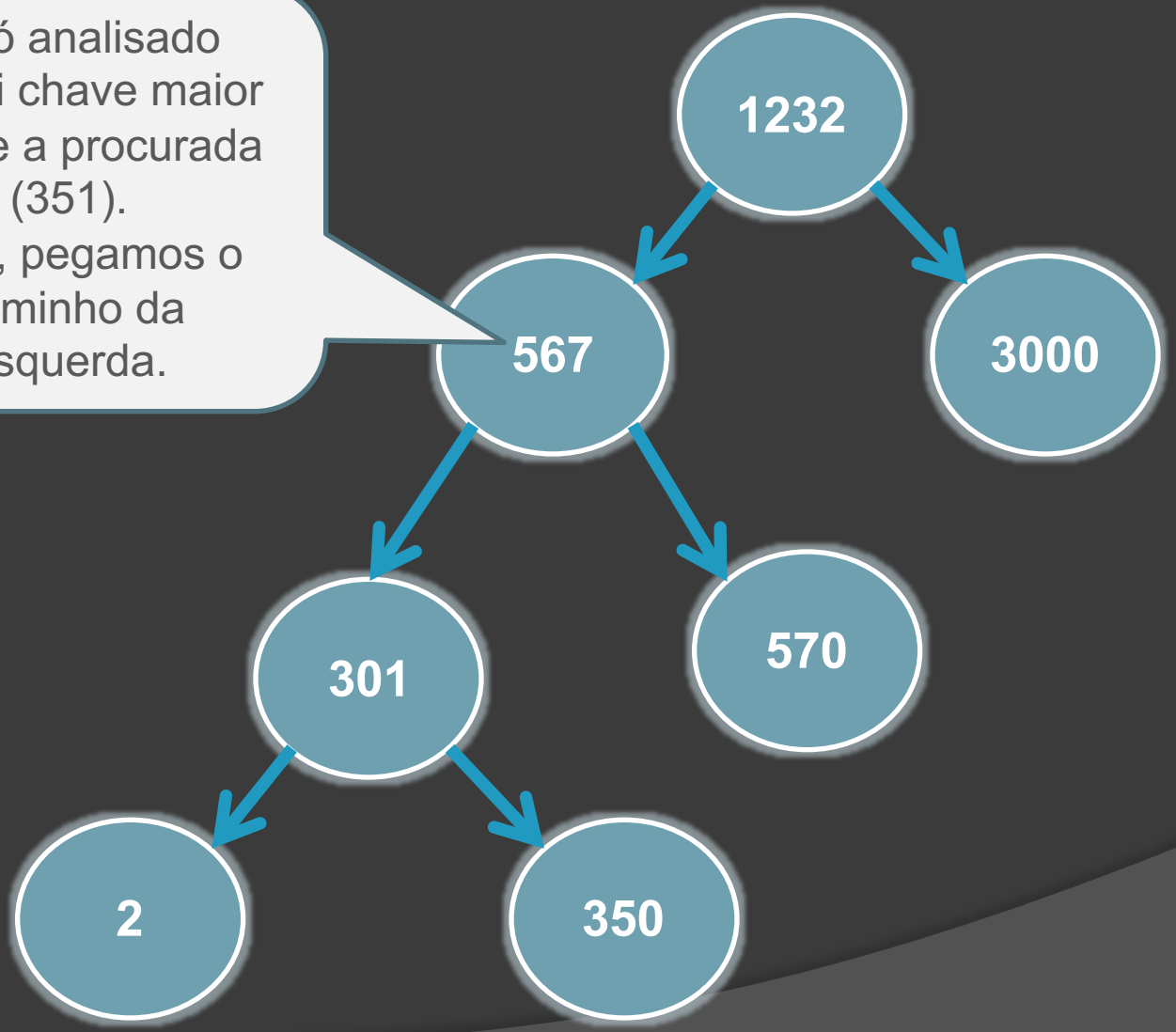
351 é menor que a chave da raíz.

Logo, utiliza-se o caminho da esquerda



Exemplo de Busca

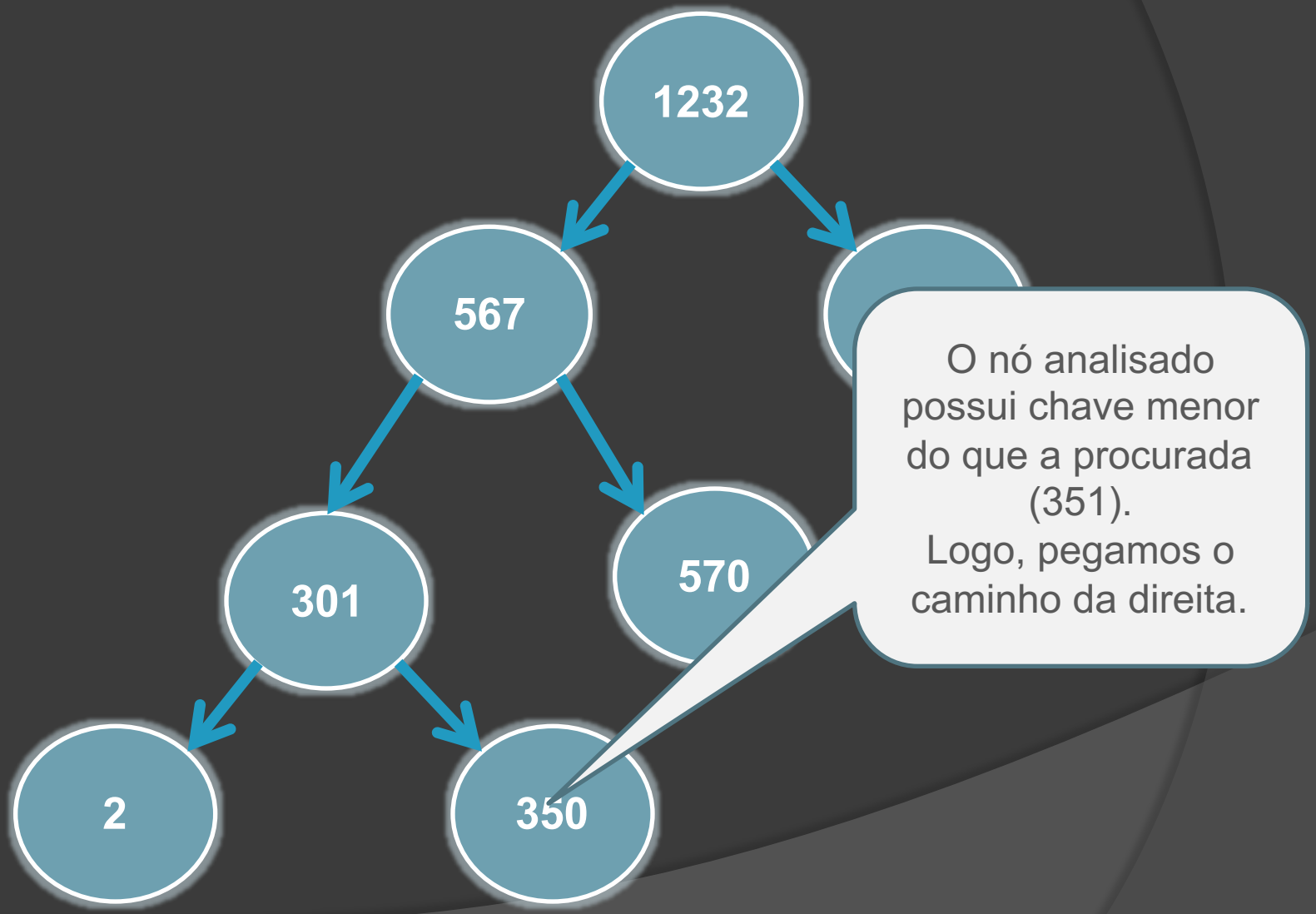
O nó analisado possui chave maior do que a procurada (351).
Logo, pegamos o caminho da esquerda.



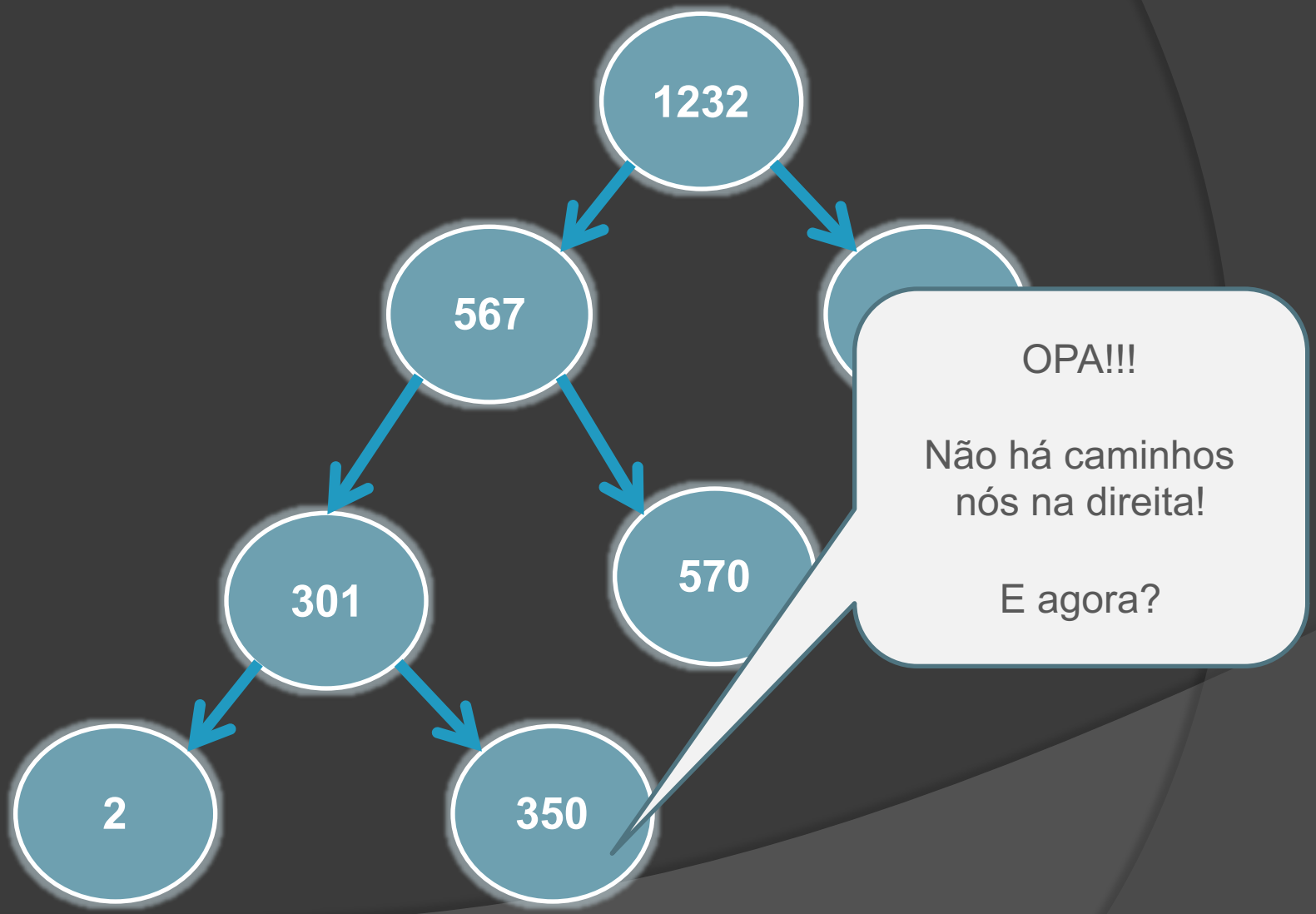
Exemplo de Busca

O nó analisado possui chave menor do que a procurada (351).
Logo, pegamos o caminho da direita.

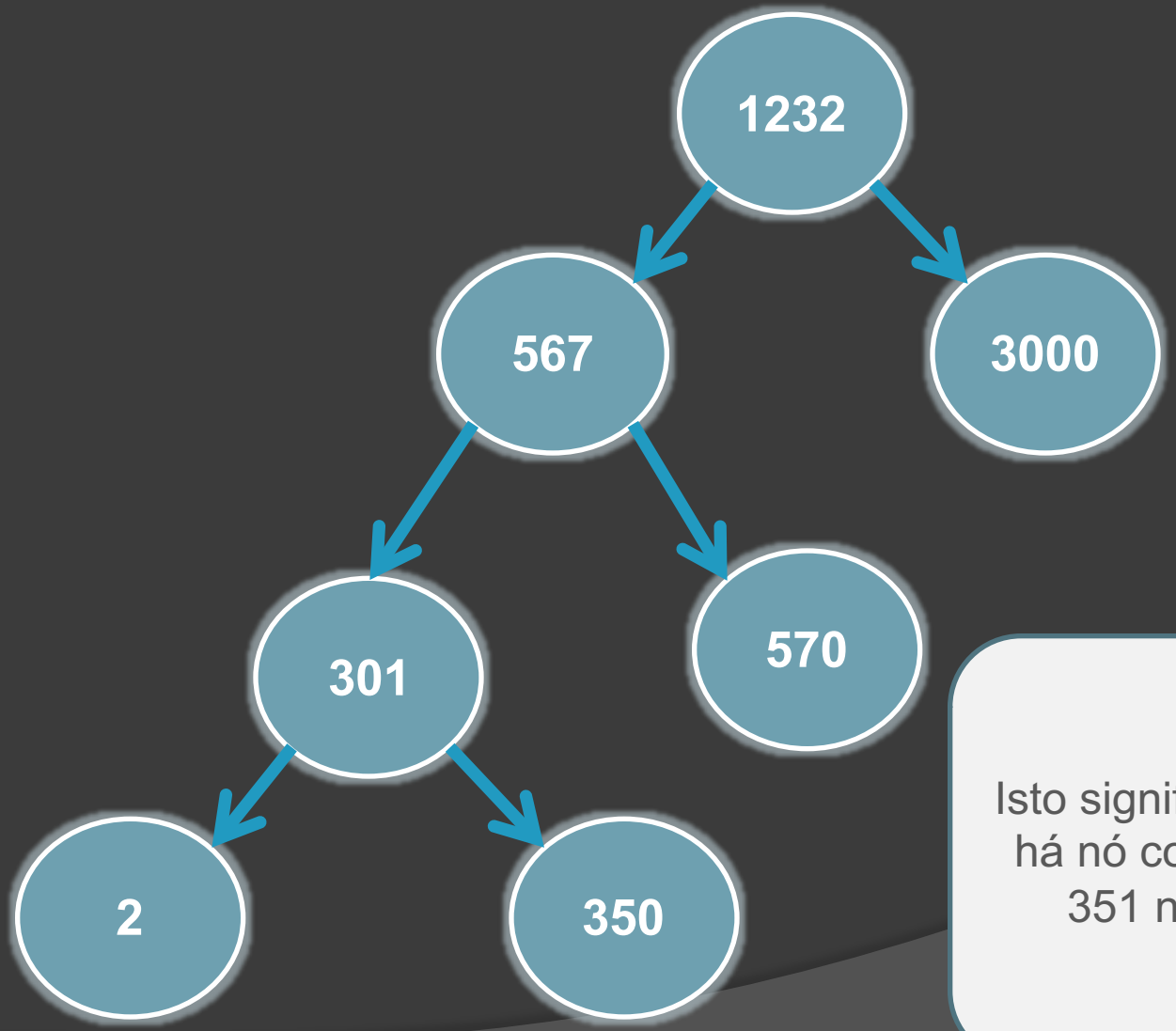
Exemplo de Busca



Exemplo de Busca



Exemplo de Busca



Isto significa que não há nó com a chave 351 na árvore

Agora, iremos planejar a TAD!

TAD

Cada nó deve possuir uma estrutura para acomodar o dado, a chave e dois ponteiros: um para direita e um para esquerda

TAD

Cada nó deve possuir uma estrutura para acomodar o dado, a chave e dois ponteiros: um para direita e um para esquerda

TAD

É importante observarmos que a chave não é necessariamente um número!
Iremos utilizar o exemplo do número, apenas para ilustrar o valor de referência do nó.

TAD

Transformando a estrutura em código (C++), veremos como podem ser representados cada nó.


```
struct TNo{
    int chave;
    TIPO dado;
    TNo * direita;
    TNo * esquerda;
};
```

TAD

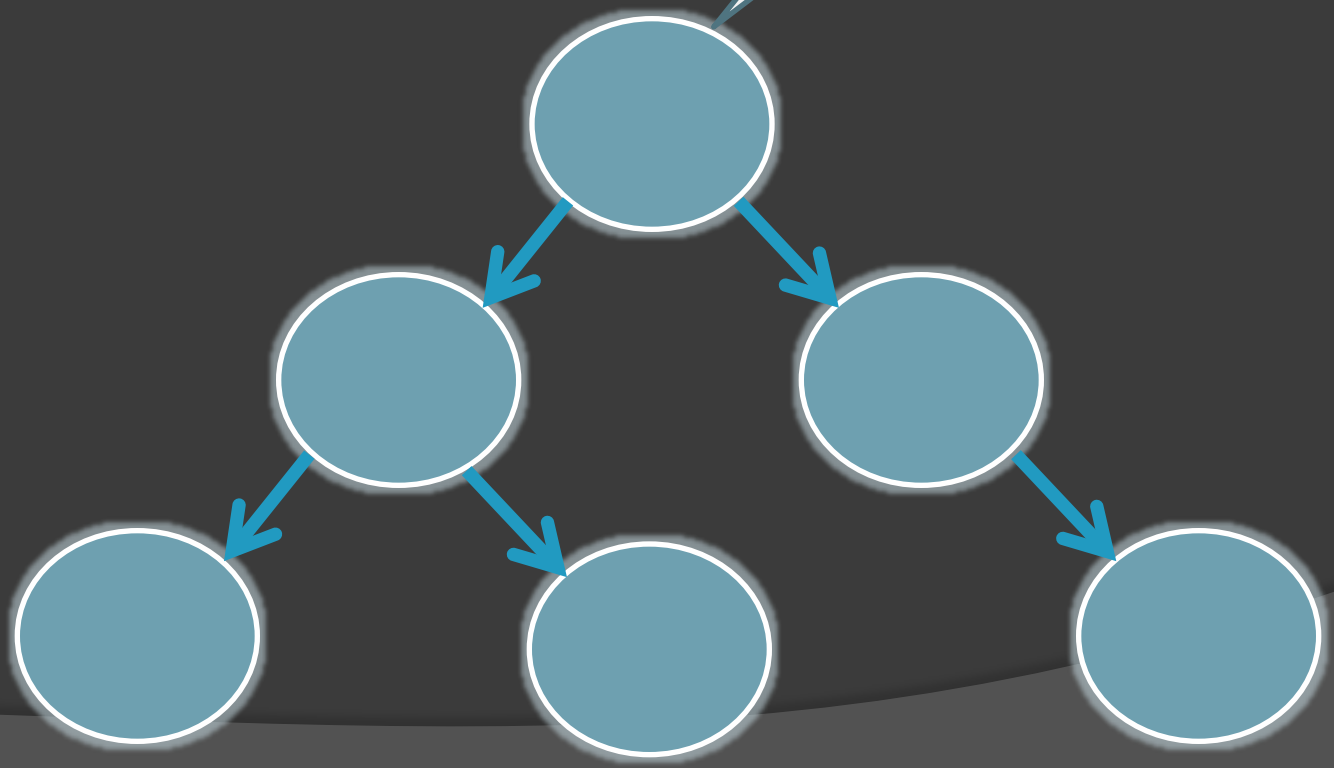
No entanto, ainda precisamos de uma estrutura que comporte a árvore em si, como fazíamos com as listas.

esquer

```
a;
da;
```

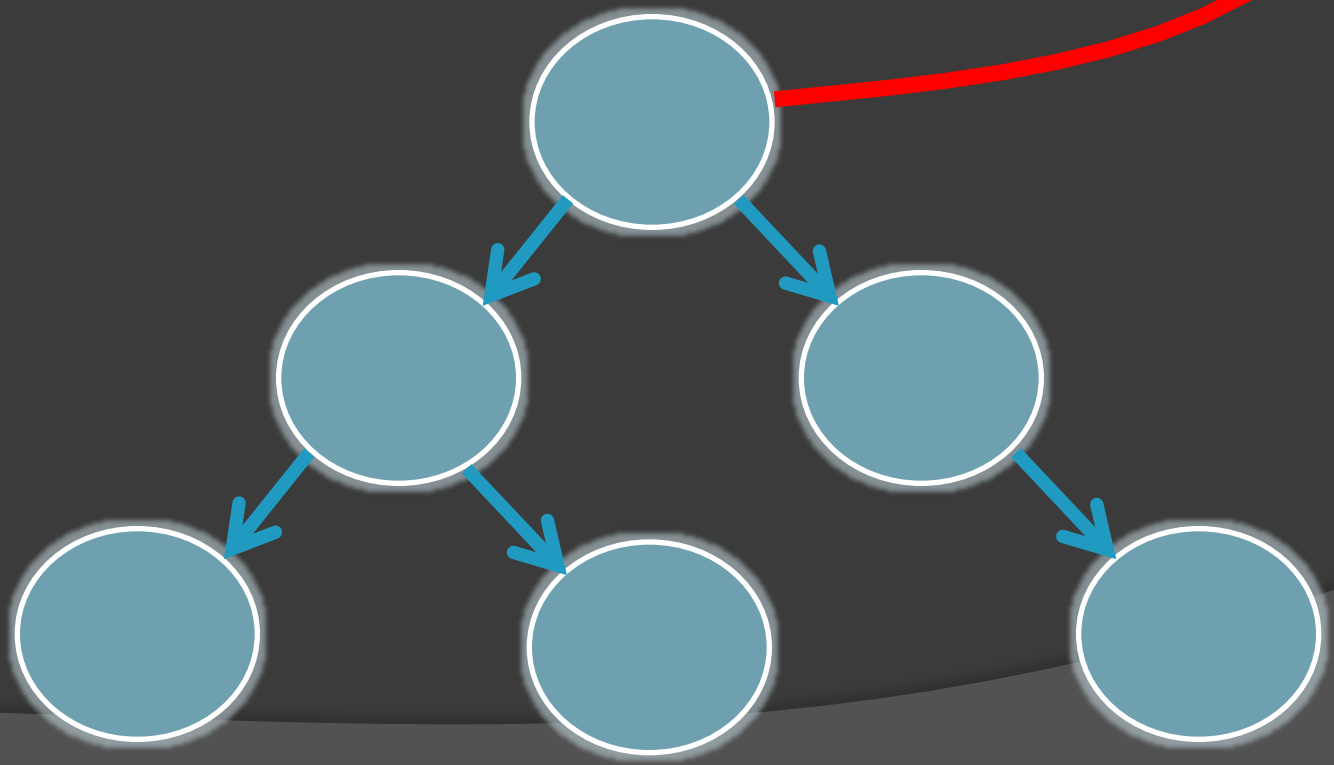
TAD

Possuindo a referência da raíz, conseguimos atingir todos os nós da árvore.



Logo, ...

```
struct TArvore{
      TNo * raiz;
};
```



Considerações

- Como operações da TAD, podemos citar:
  - inicializar;
  - inserir;
  - remover;
  - buscar.

Considerações

- Pode-se criar TADs de árvores binárias genéricas
  - Nestes casos é interessante sobrecarregar os operadores de igualdade, maior e menor (respectivamente ==, > e <) para comparar os valores das chaves.