"Tudo excelente é tão difícil quanto é raro" (Baruch Spinoza).

Exclusão em Árvore de Busca Binária

Paulo Ricardo Lisboa de Almeida

DSBD

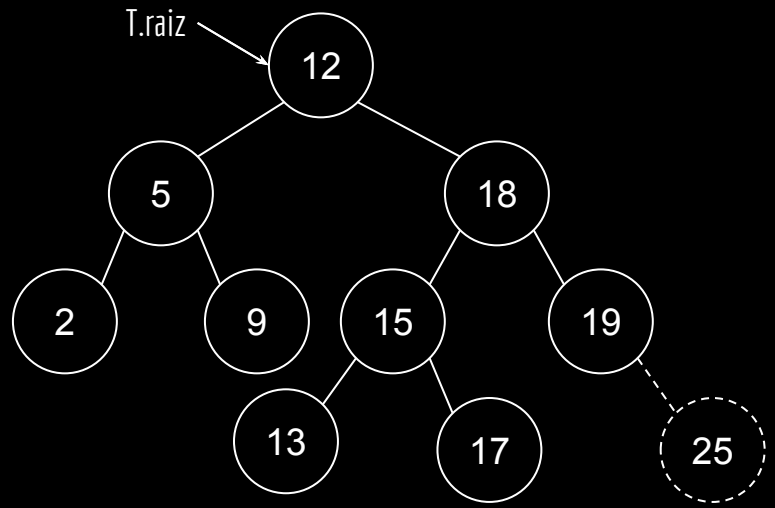
UFPR

Excluir

Como excluir um nodo de uma árvore?

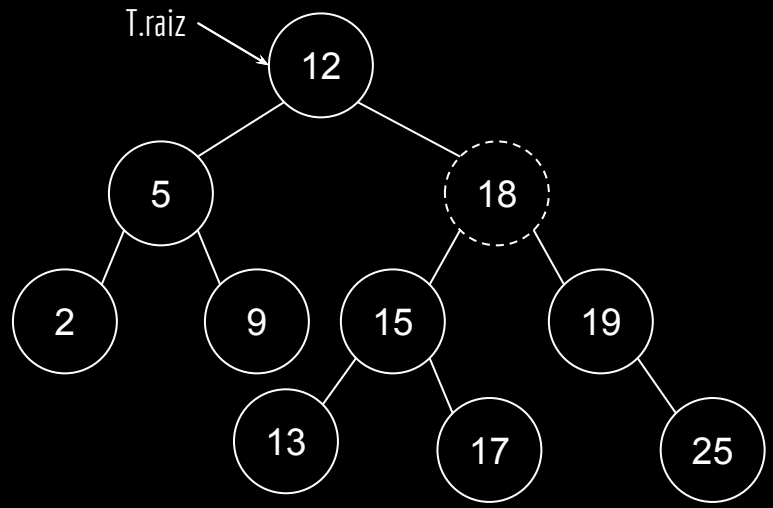
Alguns casos são triviais

Excluir o 25 no exemplo é simples.



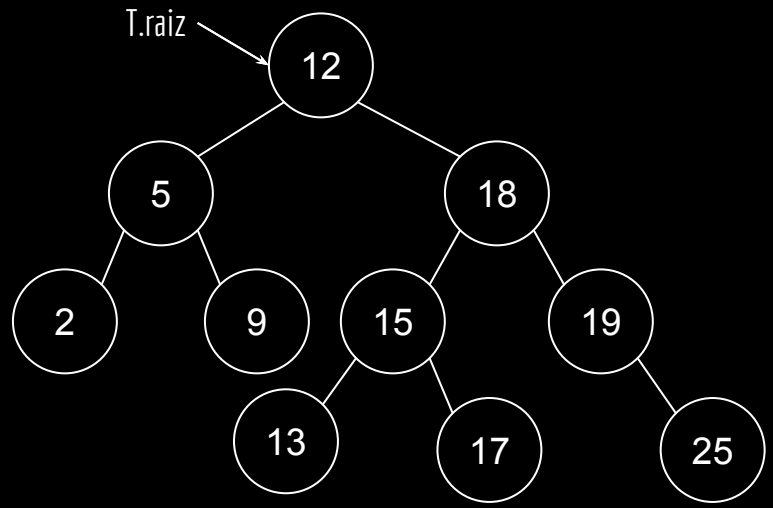
Já outros ...

Excluir o 18, nem tanto...



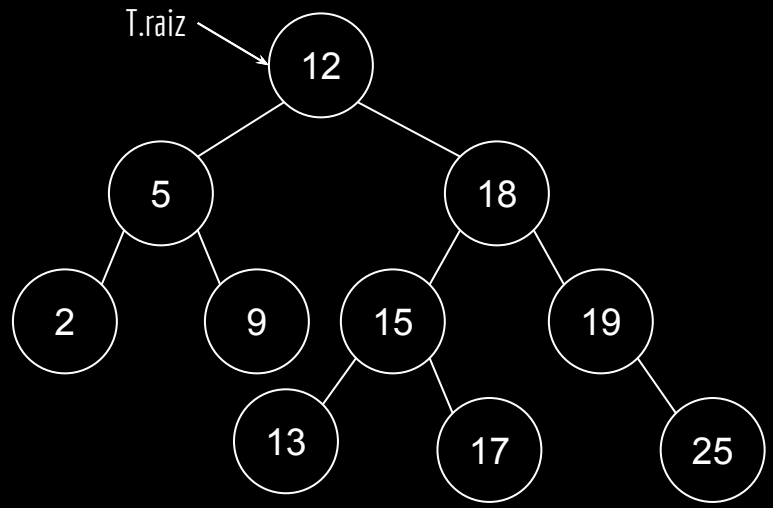
Casos

1. Se o nodo a ser excluído é uma folha, como proceder?



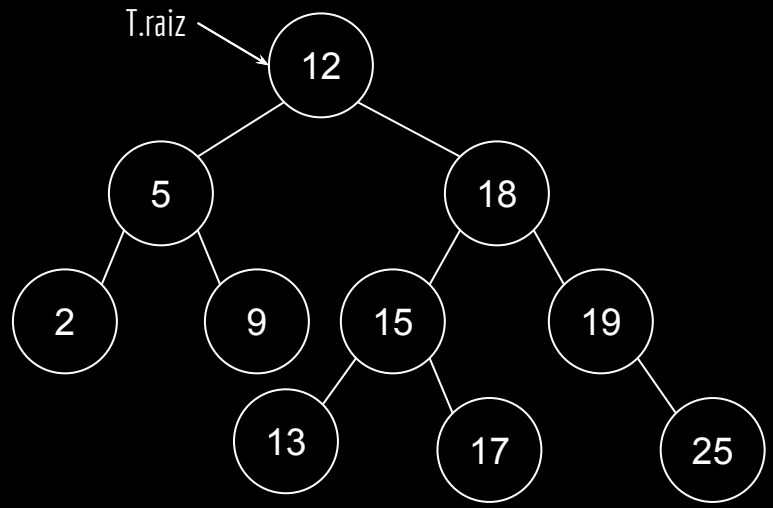
Casos

1. Se o nodo a ser excluído é uma folha, como proceder?
    a. Remover da memória, e o pai agora aponta para NULO.



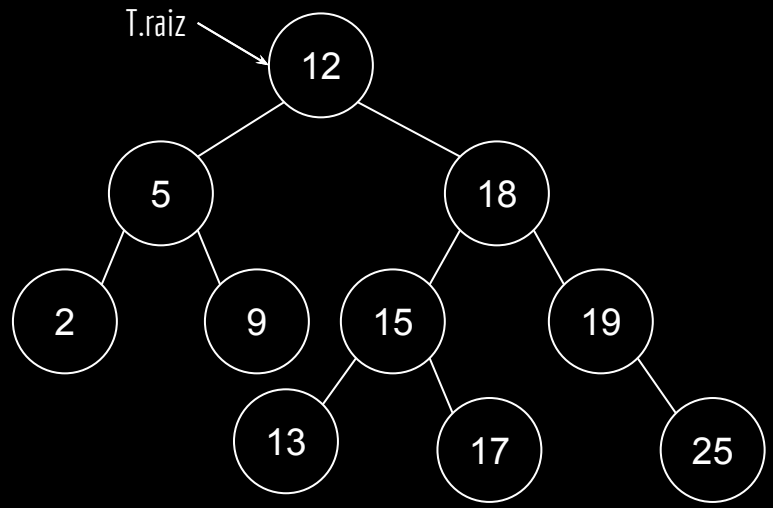
Casos

1. Se o nodo a ser excluído é uma folha, como proceder?
    a. Remover da memória, e o pai agora aponta para NULO.
2. Se o nodo a ser excluído possui um filho, como proceder?



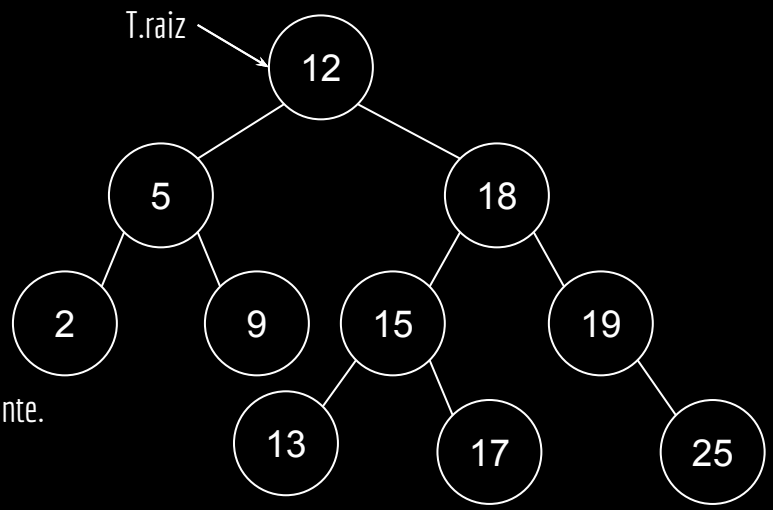
Casos

1. Se o nodo a ser excluído é uma folha, como proceder?
    a. Remover da memória, e o pai agora aponta para NULO.
2. Se o nodo a ser excluído possui um filho, como proceder?
    a. O filho do nodo vai tomar o lugar do nodo sendo excluído.



Casos

1. Se o nodo a ser excluído é uma folha, como proceder?
   a. Remover da memória, e o pai agora aponta para NULO.
2. Se o nodo a ser excluído possui um filho, como proceder?
   a. O filho do nodo vai tomar o lugar do nodo sendo excluído.
3. Se o nodo possuir ambos filhos?
   a. Encontre o sucessor y do nodo;
      i. Pertence a subárvore direita.
      ii. O sucessor garantidamente não tem subárvore esquerda.
   b. O resto da subárvore original direita se torna a subárvore direita de y.
   c. A subárvore esquerda original se torna a subárvore esquerda de y.
   d. A subárvore direita original de y vai para a posição que y estava originalmente.



Transplante

Será necessária uma função para transplantar subárvores.

A função coloca uma subárvore *v* no lugar da subárvore *u*.

v pode ser nulo (mas *u* não).

```
função transplantar(T,u,v)
entrada: árvore de busca binária T, uma subárvore u e uma subárvore v.
saída: a subárvore v irá tomar o lugar da subárvore u em T.

se u.p é NULO //u estava na raiz
     T.raiz = v
senão
     se u == u.pai.fe //???
          u.pai.fe = v
     senão
          u.pai.fd = v
se v não é NULO
     v.pai = u.pai
```

Transplante

Será necessária uma função para transplantar subárvores.

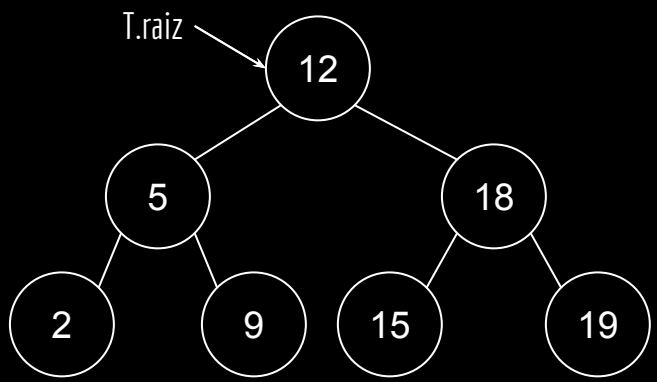
A função coloca uma subárvore *v* no lugar da subárvore *u*.

v pode ser nulo (mas *u* não).

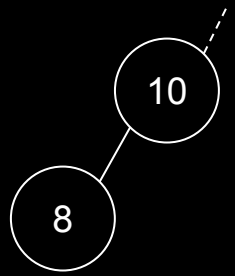
```
função transplantar(T,u,v)
entrada: árvore de busca binária T, uma subárvore u e uma subárvore v.
saída: a subárvore v irá tomar o lugar da subárvore u em T.

se u.p é NULO //u estava na raiz
     T.raiz = v
senão
     se u == u.pai.fe //u é um filho esquerdo?
          u.pai.fe = v
     senão
          u.pai.fd = v
se v não é NULO
     v.pai = u.pai
```

Teste de mesa

transplantar(T,5,10)

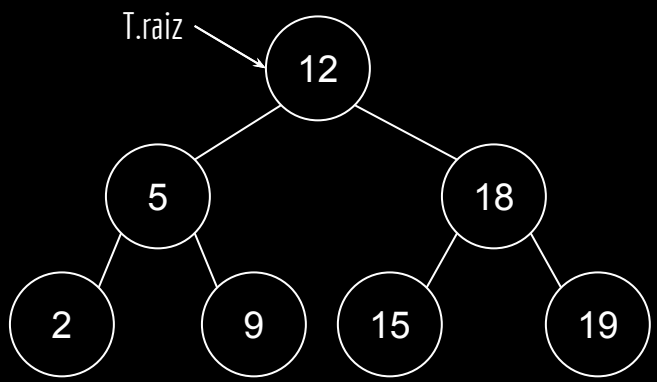



```
função transplantar(T,u,v)
se u.p é NULO //u estava na raiz
     T.raiz = v
senão
     se u == u.pai.fe //u é um filho esquerdo?
          u.pai.fe = v
     senão
          u.pai.fd = v
se v não é NULO
     v.pai = u.pai
```

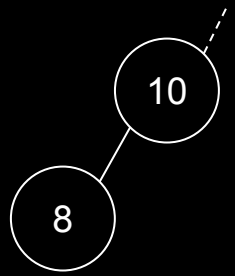


Teste de mesa

transplantar(T,5,10)

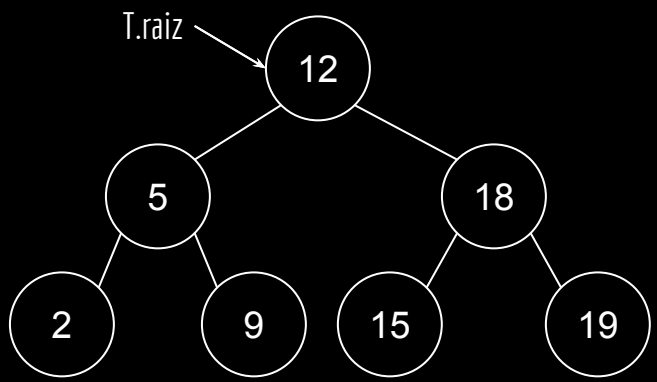



```
função transplantar(T,u,v)
se u.p é NULO //u estava na raiz
     T.raiz = v
senão
     se u == u.pai.fe //u é um filho esquerdo?
          u.pai.fe = v
     senão
          u.pai.fd = v
se v não é NULO
     v.pai = u.pai
```

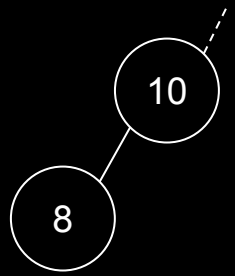


Teste de mesa

transplantar(T,5,10)

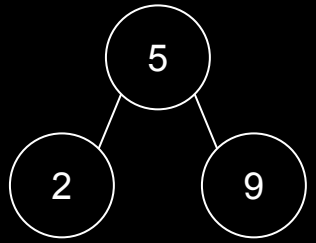


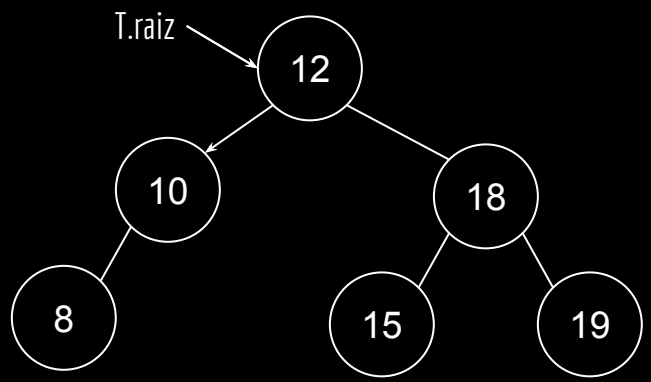

```
função transplantar(T,u,v)
se u.p é NULO //u estava na raiz
     T.raiz = v
senão
     se u == u.pai.fe //u é um filho esquerdo?
          u.pai.fe = v
     senão
          u.pai.fd = v
se v não é NULO
     v.pai = u.pai
```



Teste de mesa

transplantar(T,5,10)

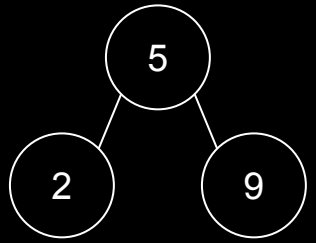


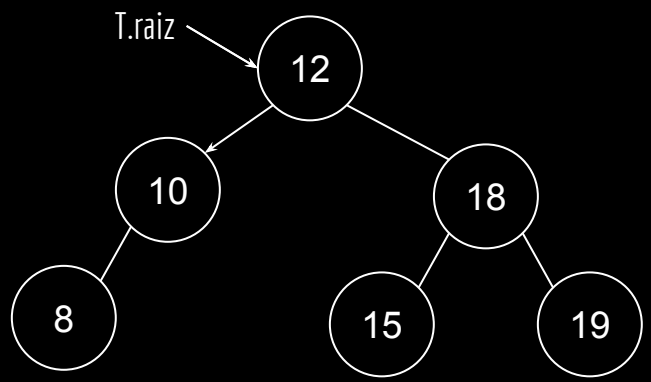



```
função transplantar(T,u,v)
se u.p é NULO //u estava na raiz
     T.raiz = v
senão
     se u == u.pai.fe //u é um filho esquerdo?
          u.pai.fe = v
     senão
          u.pai.fd = v
se v não é NULO
     v.pai = u.pai
```

Teste de mesa

transplantar(T,5,10)

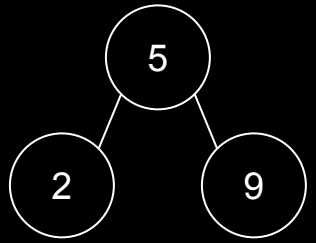


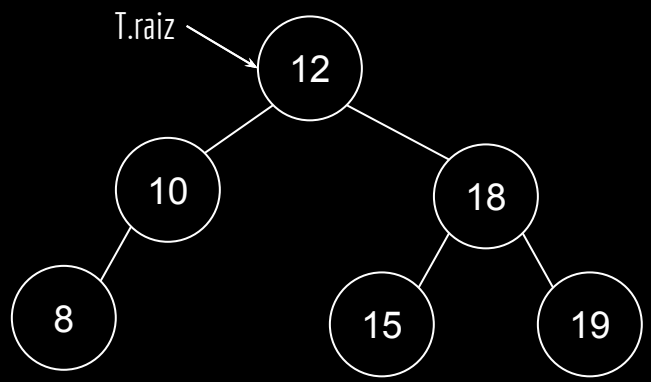



```
função transplantar(T,u,v)
se u.p é NULO //u estava na raiz
     T.raiz = v
senão
     se u == u.pai.fe //u é um filho esquerdo?
          u.pai.fe = v
     senão
          u.pai.fd = v
se v não é NULO
     v.pai = u.pai
```

Teste de mesa

transplantar(T,5,10)






```
função transplantar(T,u,v)
se u.p é NULO //u estava na raiz
     T.raiz = v
senão
     se u == u.pai.fe //u é um filho esquerdo?
          u.pai.fe = v
     senão
          u.pai.fd = v
se v não é NULO
     v.pai = u.pai
```

Excluir

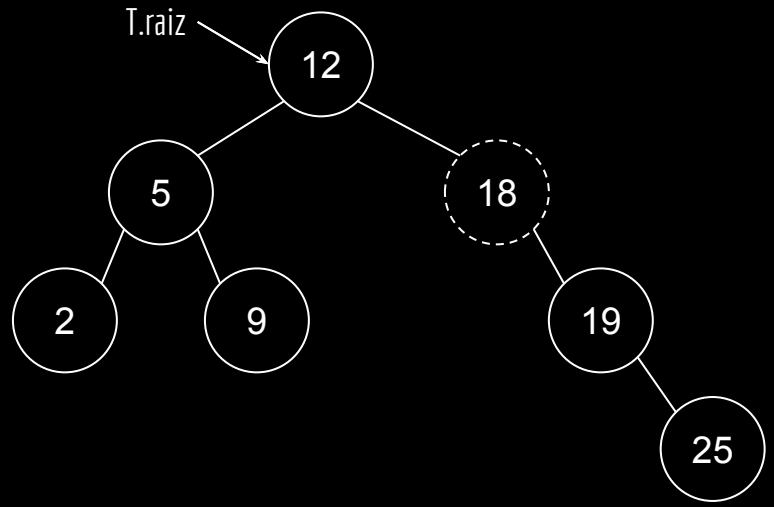
função **excluir**(T,z)
entrada: árvore de busca binária T, e o nodo a ser excluído z.
saída: o nodo z é excluído de forma a manter a propriedade da árvore de busca binária.

```
se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```

Teste de mesa

excluir(T,18)

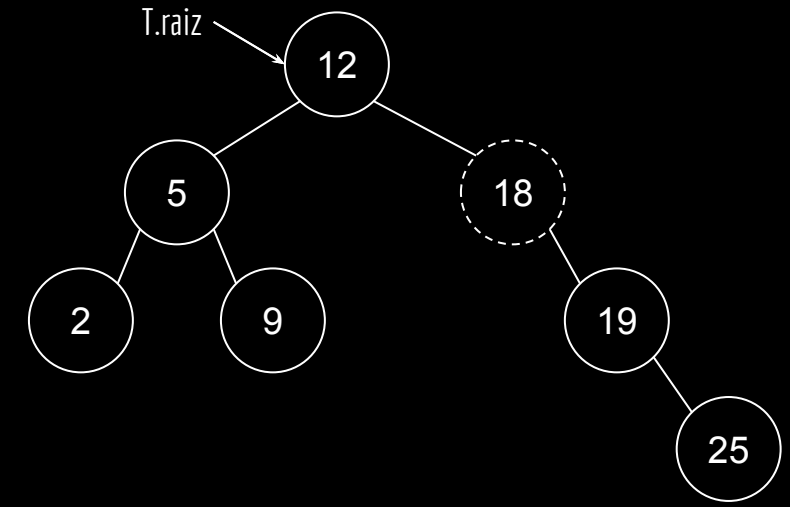
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,18)

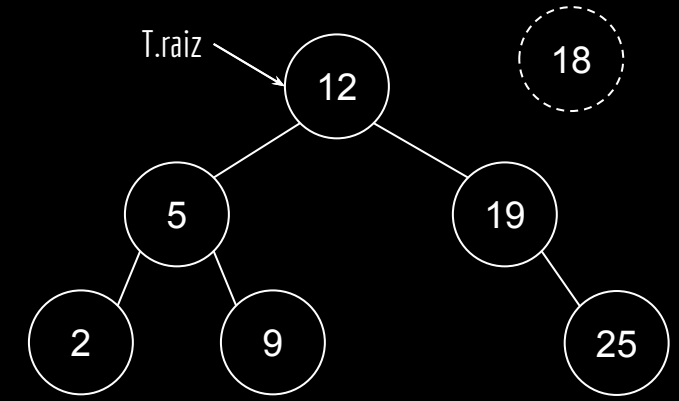
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,18)

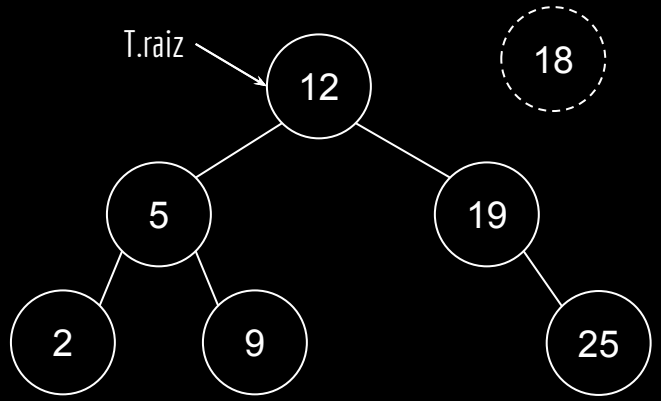
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,18)

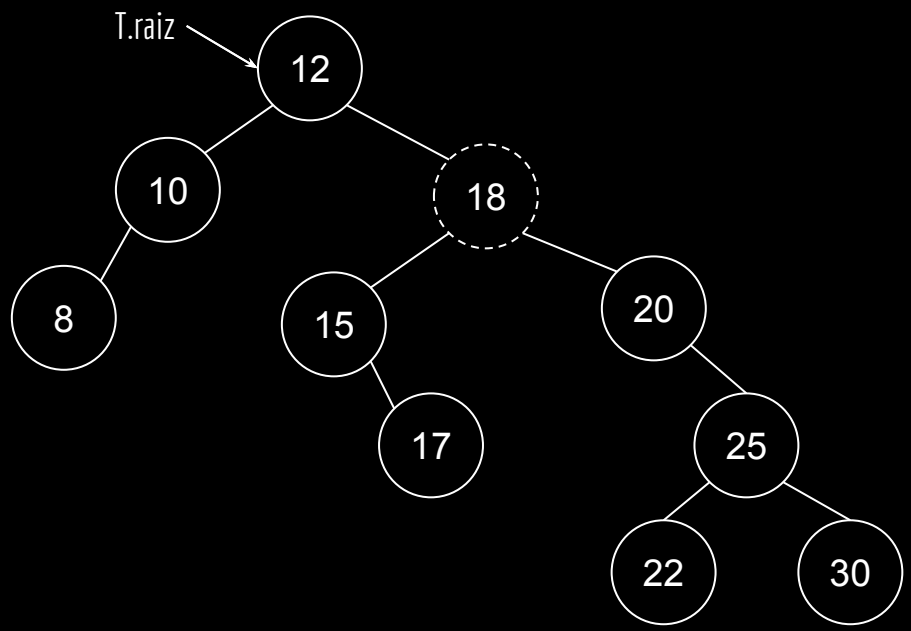
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,18)

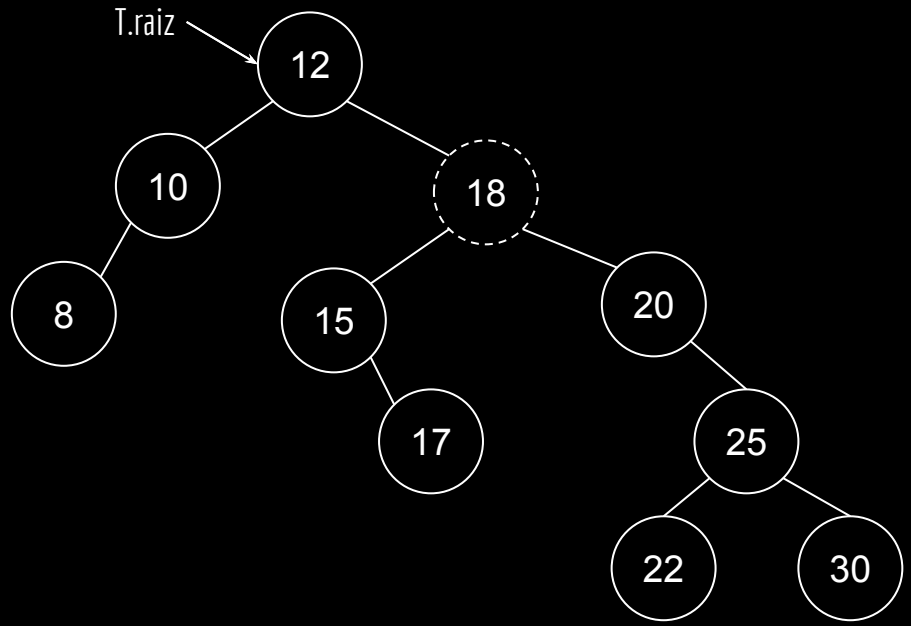
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,18)

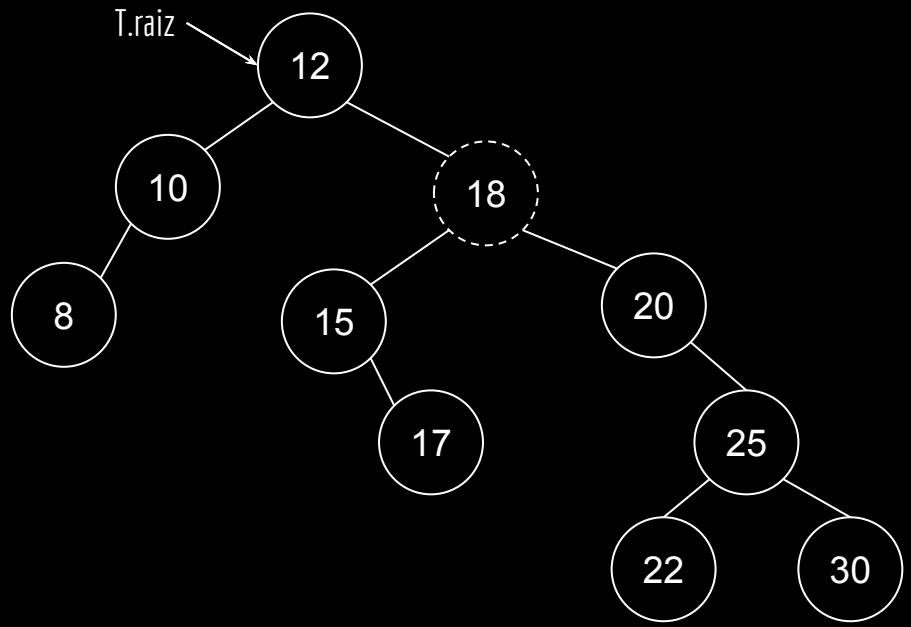
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,18)

```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```

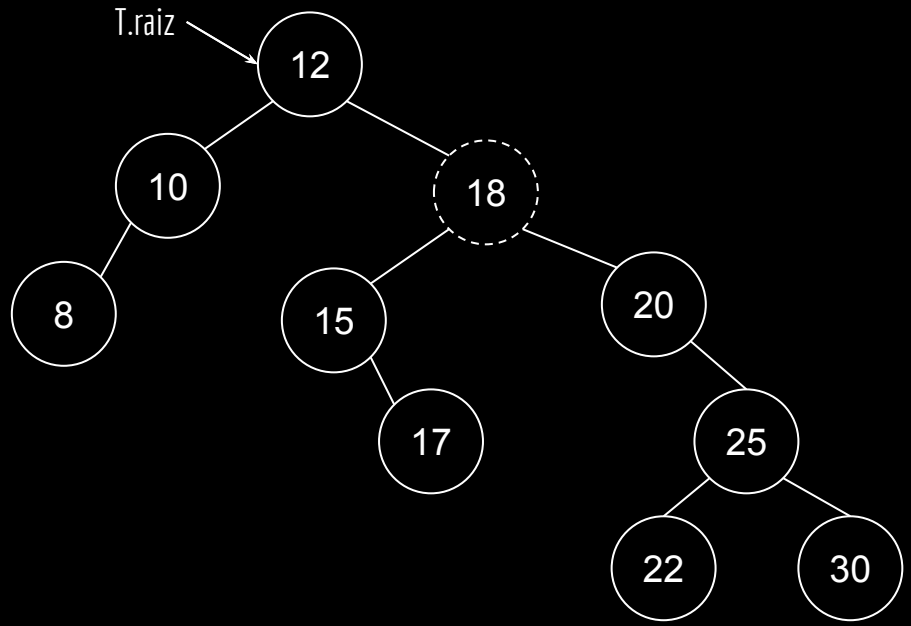


Teste de mesa

excluir(T,18)
y
20

```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```

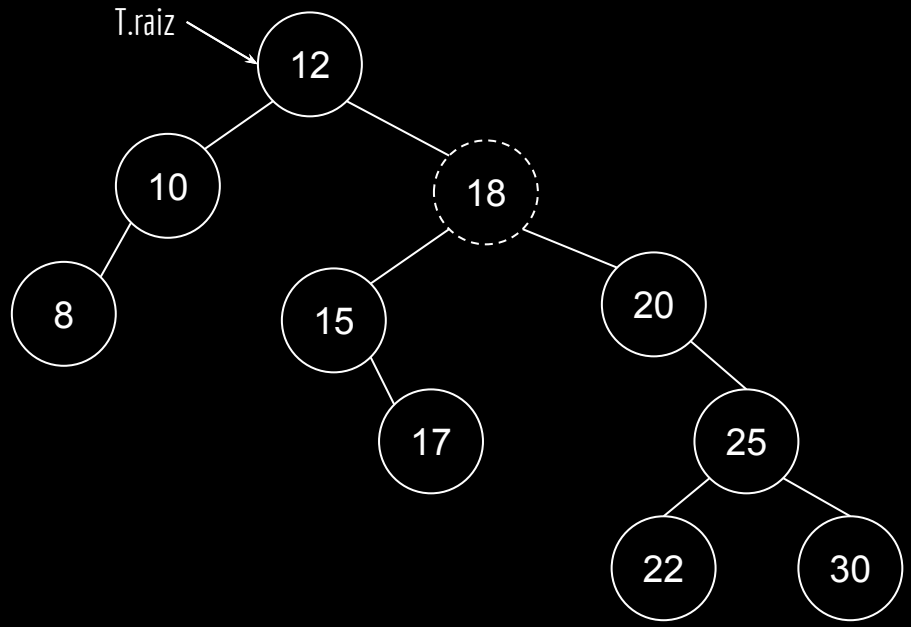
Calcular o sucessor.



Teste de mesa

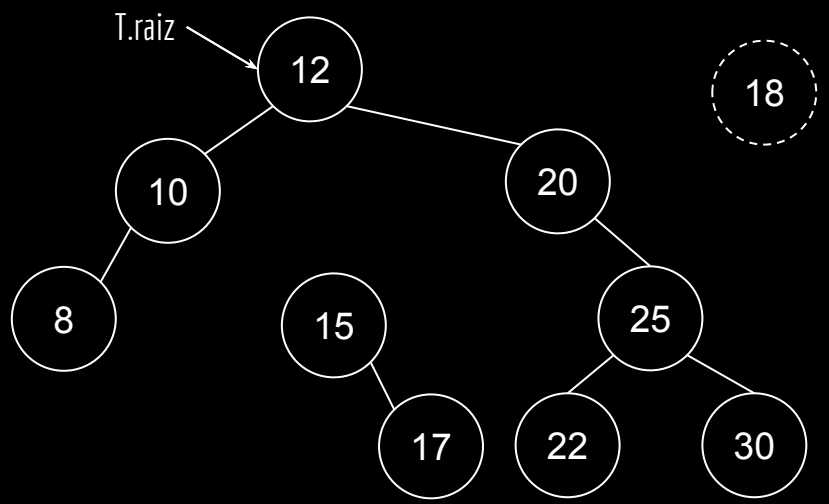
excluir(T,18)
y
20

```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,18)
y
20

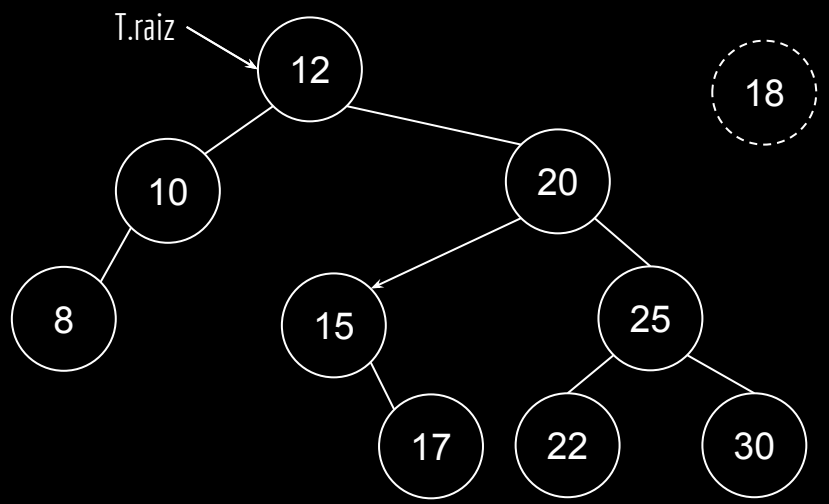



```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```

Teste de mesa

excluir(T,18)
y
20

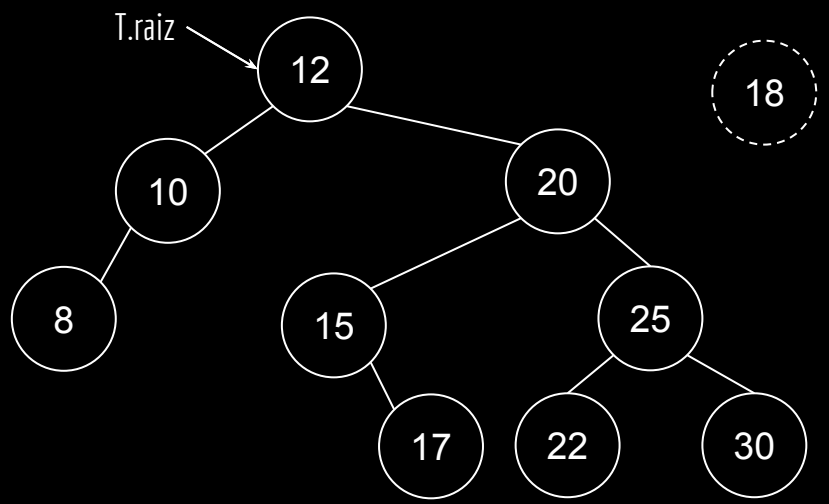
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,18)
y
20

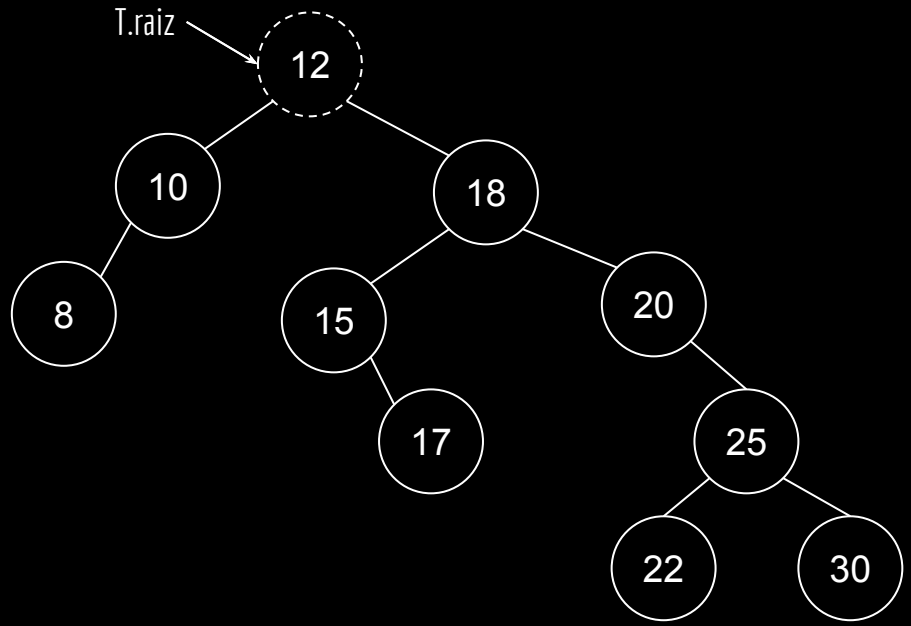
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,12)

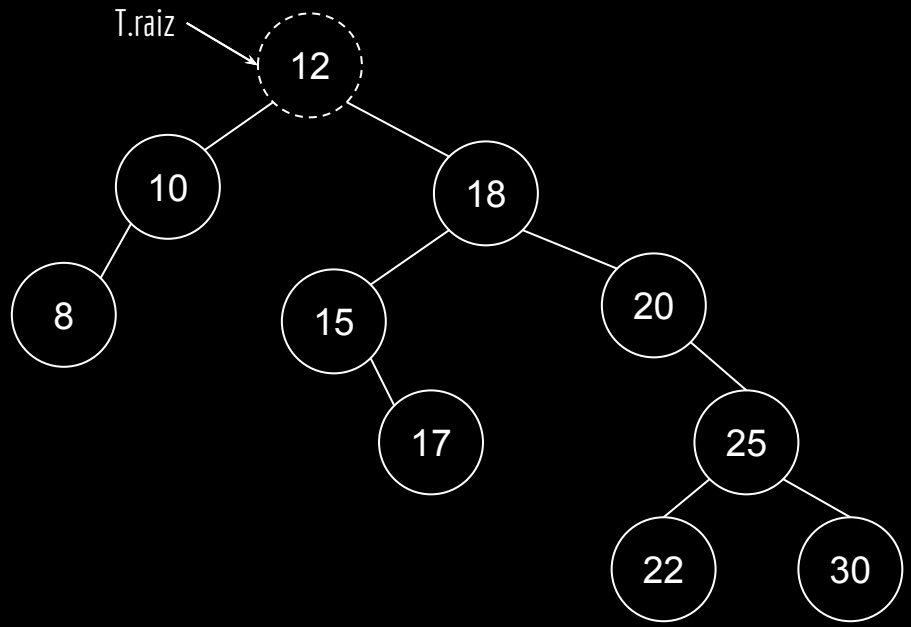
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,12)

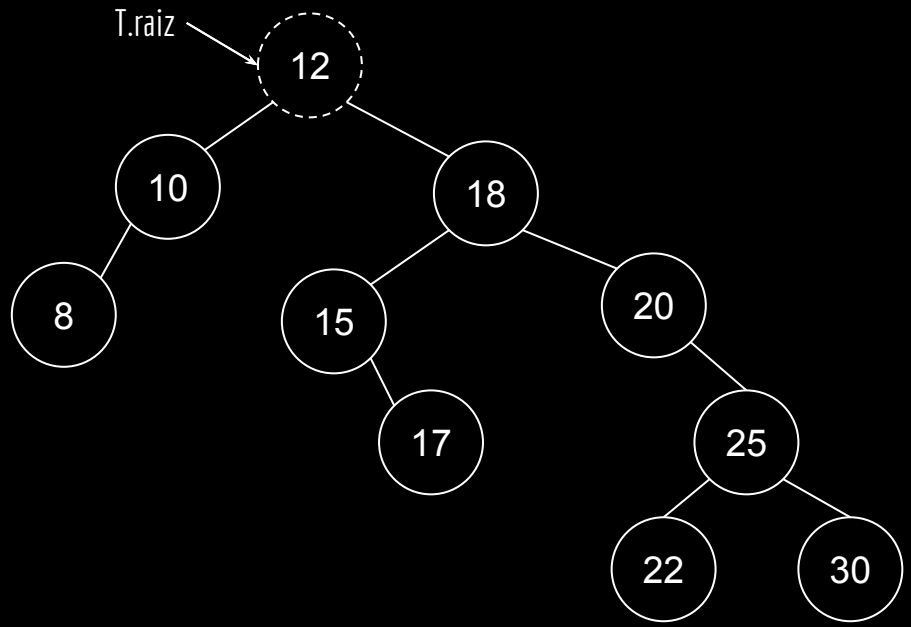
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,12)

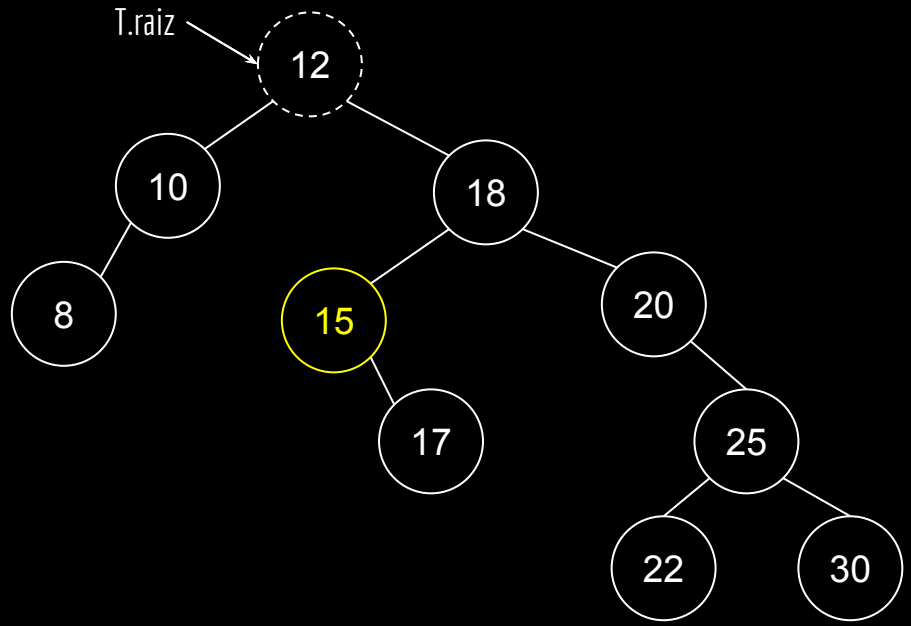
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,12)
y
15

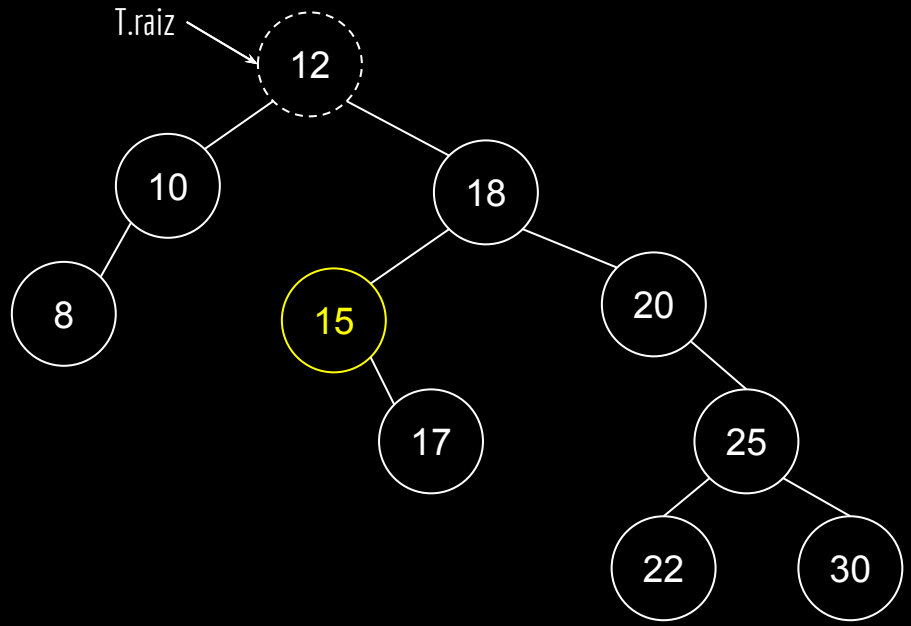
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,12)
y
15

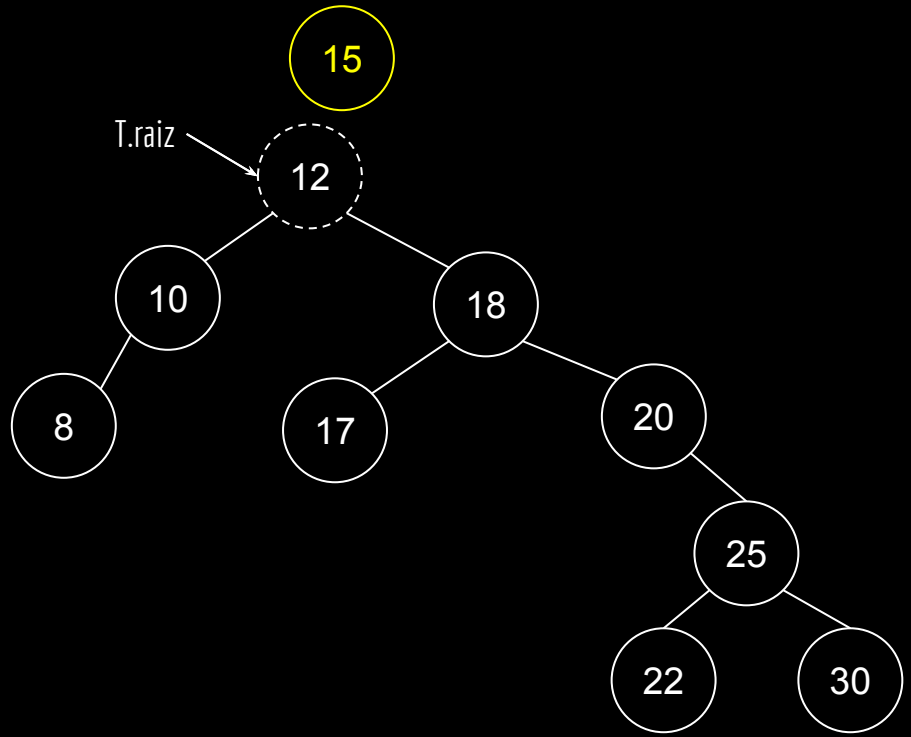
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,12)
y
15

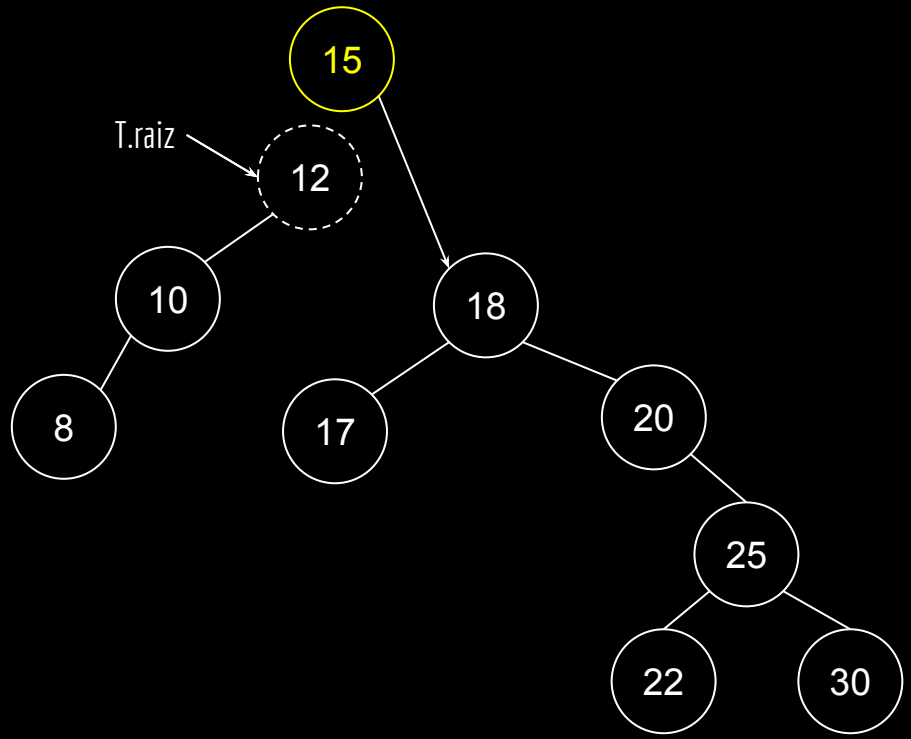
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,12)
y
15

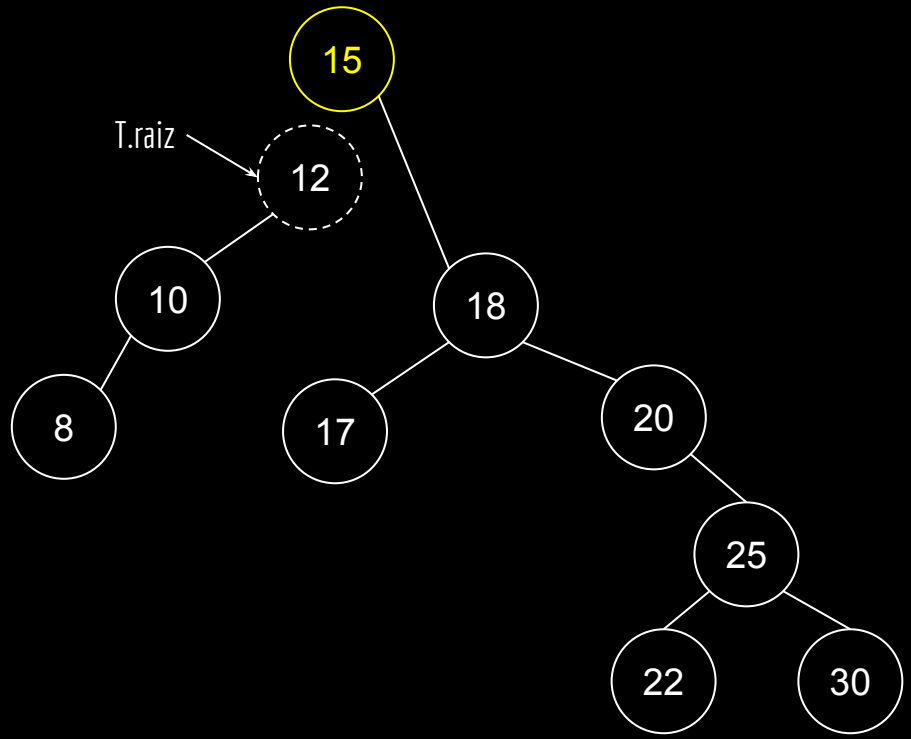
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,12)
y
15

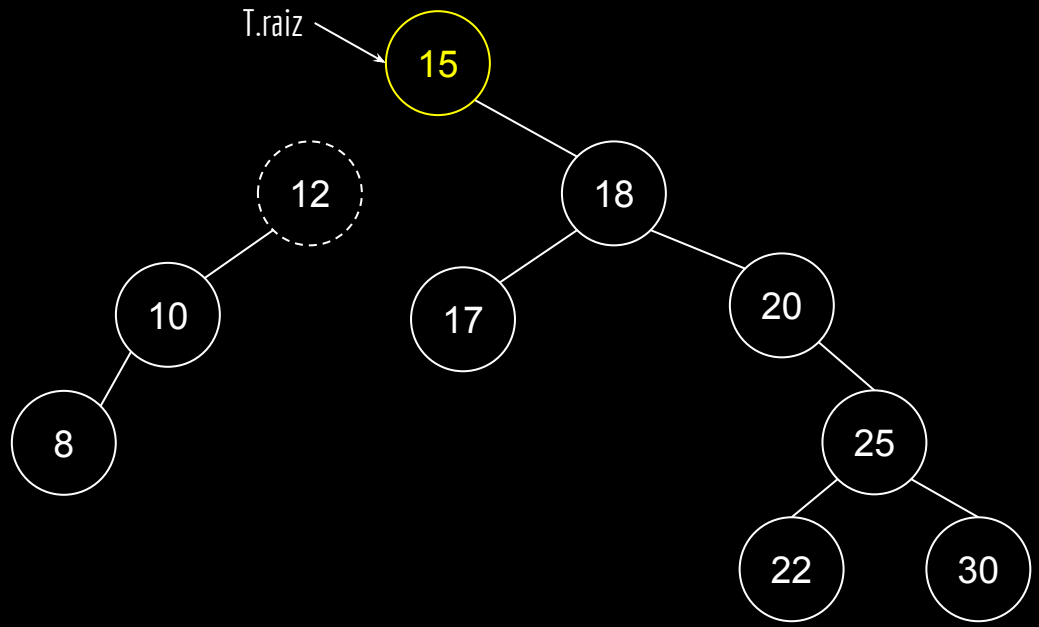
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

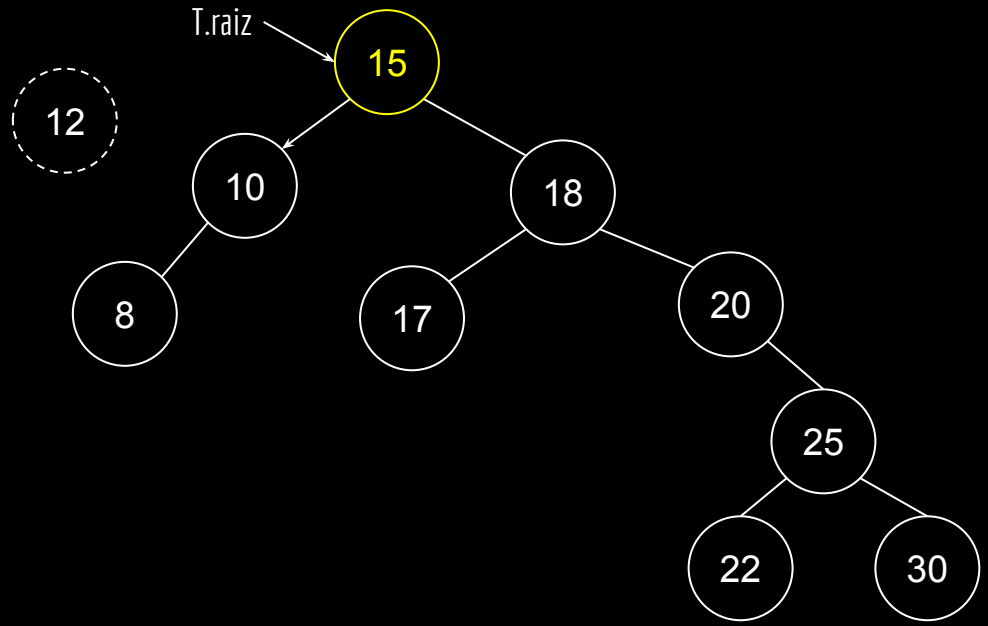
excluir(T,12)
y
15

```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```



Teste de mesa

excluir(T,12)
y
15

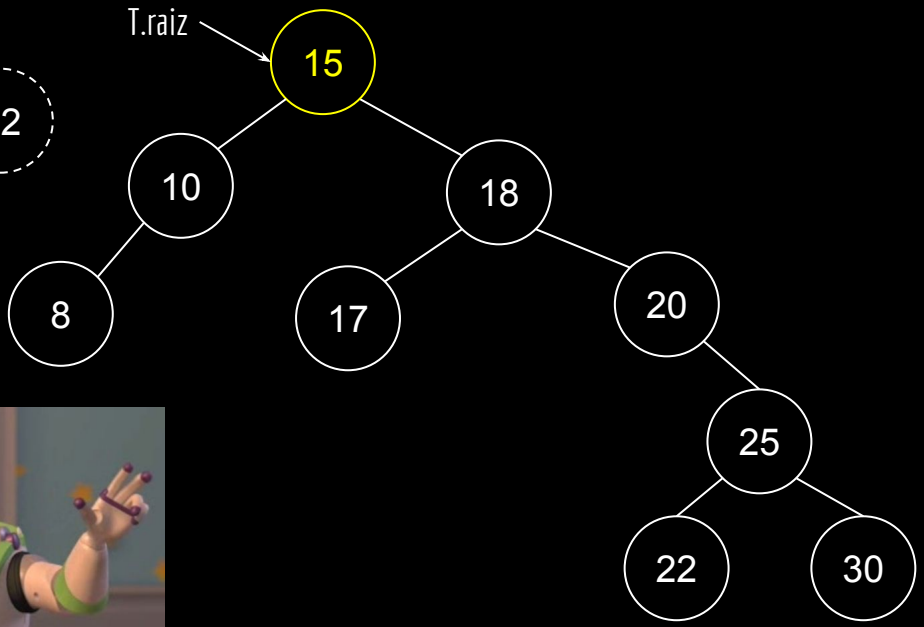


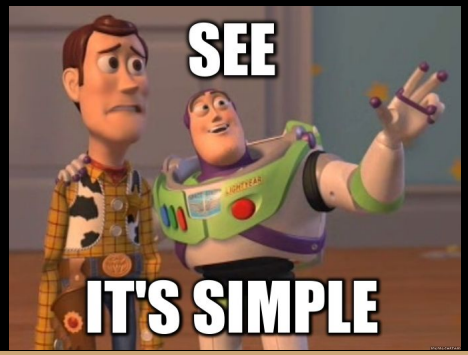

```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```

Teste de mesa

excluir(T,12)
y
15

```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```





Custo

Todas as linhas exceto a chamada da função **`minimo`** custam *O(1)*.

São de tempo constante.

A função **`minimo`** custa *O(h)* - Veja uma prova em Cormen et. al (2012).

Onde *h* é a altura da árvore.

Logo, assim como inserir e buscar, excluir custa *O(h)*.

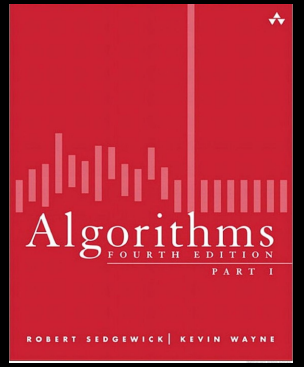
```
função excluir(T,z)

se z.fe é NULO
     transplantar(T, z, z.fd)
     retorne
se z.fd é NULO
     transplantar(T, z, z.fe)
     retorne
y = minimo(z.fd)
se y ≠ z.fd
     transplantar(T,y,y.fd)
     y.fd = z.fd
     y.fd.pai = y
transplantar(T,z,y)
y.fe = z.fe
y.fe.pai = y
```

Exercícios

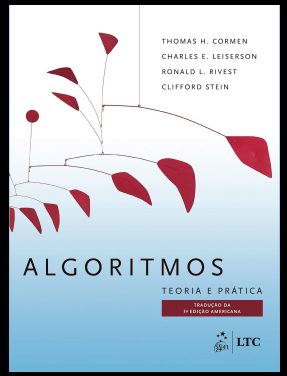
1. Implemente os algoritmos em C.

Referências

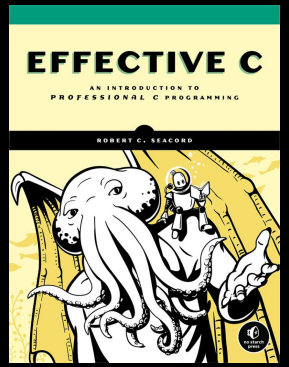
R. Sedgewick, K. Wayne. Algorithms Part I. 4a ed. 2011



T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos: Teoria e Prática. 3a ed. 2012.



Seacord, R. C. Effective C: An ntroduction to Professional C Programming. 2020.



Licença