Estruturas de Dados

Cristina Gomes Fernandes

Árvores de busca binária (ABB)

Dicionário: tipo abstracto de dados para um conjunto com operações de inserção, remoção e busca.

Árvores de busca binária (ABB)

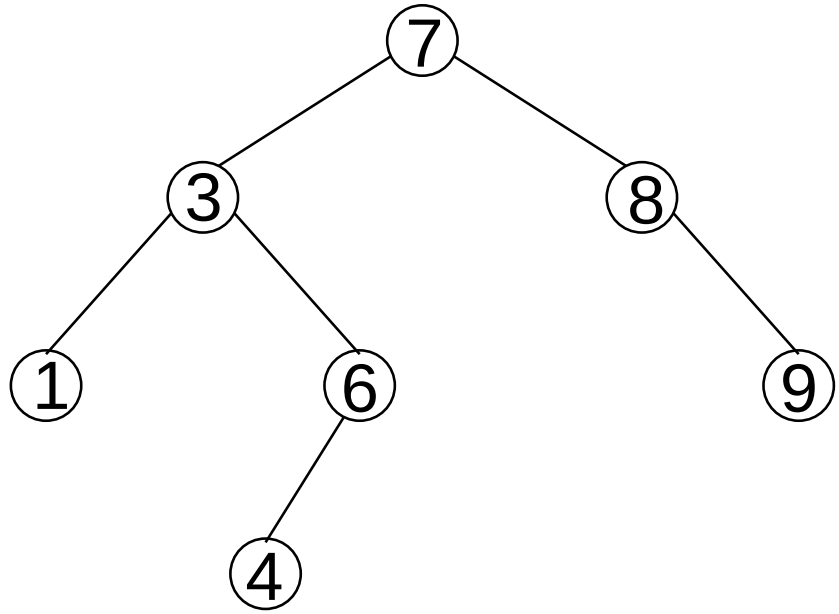
Dicionário: tipo abstracto de dados para um conjunto com operações de inserção, remoção e busca.

ABB: árvore binária onde, para cada nó x, todo nó da subárvore esquerda de x tem $info$ menor que $info(x)$, e todo nó da subárvore direita de x tem $info$ maior que $info(x)$.

Árvores de busca binária (ABB)

Dicionário: tipo abstracto de dados para um conjunto com operações de inserção, remoção e busca.

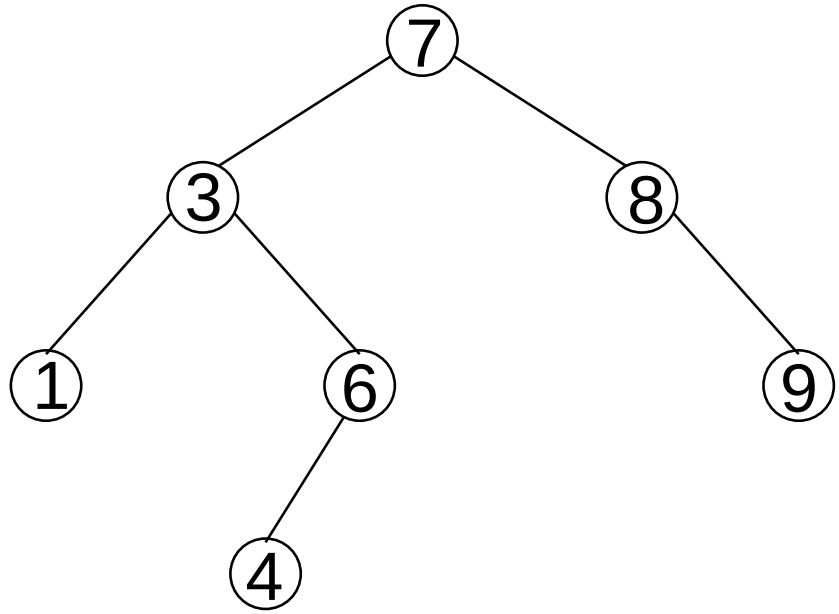
ABB: árvore binária onde, para cada nó x, todo nó da subárvore esquerda de x tem $info$ menor que $info(x)$, e todo nó da subárvore direita de x tem $info$ maior que $info(x)$.



Árvores de busca binária (ABB)

Dicionário: tipo abstracto de dados para um conjunto com operações de inserção, remoção e busca.

ABB: árvore binária onde, para cada nó x, todo nó da subárvore esquerda de x tem $info$ menor que $info(x)$, e todo nó da subárvore direita de x tem $info$ maior que $info(x)$.



ABBs implementam dicionários e filas de prioridades.

Busca em ABBs

```
BUSQUE (T, x)
1  se T = NIL ou info(T) = x
2     então devolva T
3     senão se x < info(T)
4              então devolva BUSQUE(esq(T), x)
5              senão devolva BUSQUE(dir(T), x)
```

Busca em ABBs

```
BUSQUE (T, x)
1  se T = NIL ou info(T) = x
2     então devolva T
3     senão se x < info(T)
4              então devolva BUSQUE(esq(T), x)
5              senão devolva BUSQUE(dir(T), x)
```

Versão iterativa:

```
BUSQUE (T, x)
1  p ← T
2  enquanto p ≠ NIL e info(p) ≠ x faça
3     se x < info(p)
4        então p ← esq(p)
5        senão p ← dir(p)
6  devolva p
```

Inserção em ABBs

```
INSIRA (T, x)
 1  p ← T
 2  ant ← NIL
 3  enquanto p ≠ NIL faça
 4     ant ← p
 5     se x < info(p)
 6        então p ← esq(p)
 7        senão p ← dir(p)
 8  q ← NOVACÉLULA(x, NIL, NIL)
 9  se ant = NIL
10     então T ← q
11     senão se x < info(ant)
12                 então esq(ant) ← q
13                 senão dir(ant) ← q
```

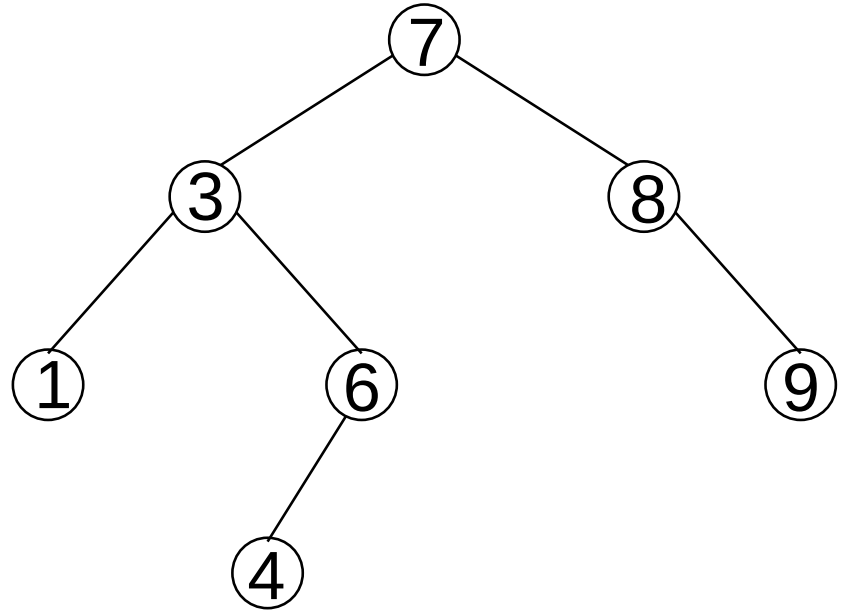
Versão recursiva da inserção

```
INSIRA (T, x)
 1  T ← INSIRAREC(T, x)

INSIRAREC (T, x)
 1  se T = NIL
 2     então q ← NOVACÉLULA(x, NIL, NIL)
 3           devolva q
 4  se x < info(T)
 5     então esq(T) ← INSIRAREC(esq(T), x)
 6     senão dir(T) ← INSIRAREC(dir(T), x)
 7  devolva T
```

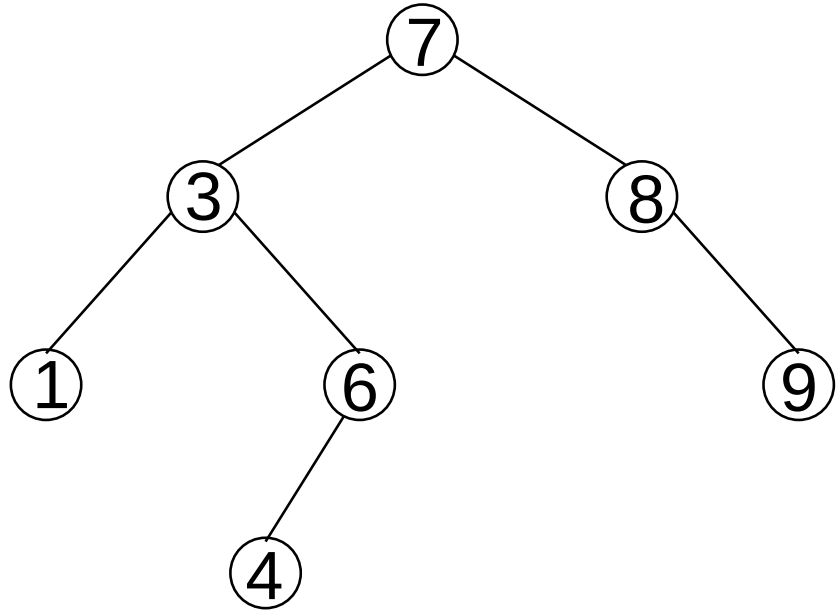
Altura de uma ABB

Estas operações consomem tempo,
no pior caso, proporcional à altura da árvore.



Altura de uma ABB

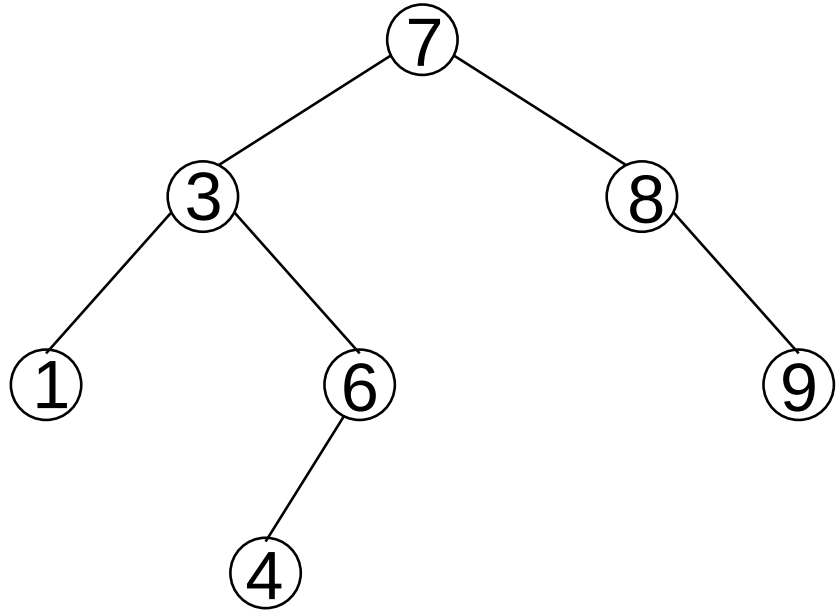
Estas operações consomem tempo,
no pior caso, proporcional à altura da árvore.



A altura de uma ABB por chegar a $n - 1$,
onde n é o número de elementos armazenados na ABB.
No pior caso, as operações consomem tempo linear em n.

Altura de uma ABB

Estas operações consomem tempo,
no pior caso, proporcional à altura da árvore.



A altura de uma ABB por chegar a $n - 1$,
onde n é o número de elementos armazenados na ABB.
No pior caso, as operações consomem tempo linear em n.

Como evitar esse caso ruim?

Árvores balanceadas

Uma árvore 2-3 é uma árvore vazia ou

Árvores balanceadas

Uma árvore 2-3 é uma árvore vazia ou

- um 2-nó, com uma chave e dois filhos;
  filho esquerdo: árvore 2-3 com chaves menores;
  filho direito: árvore 2-3 com chaves maiores.

Árvores balanceadas

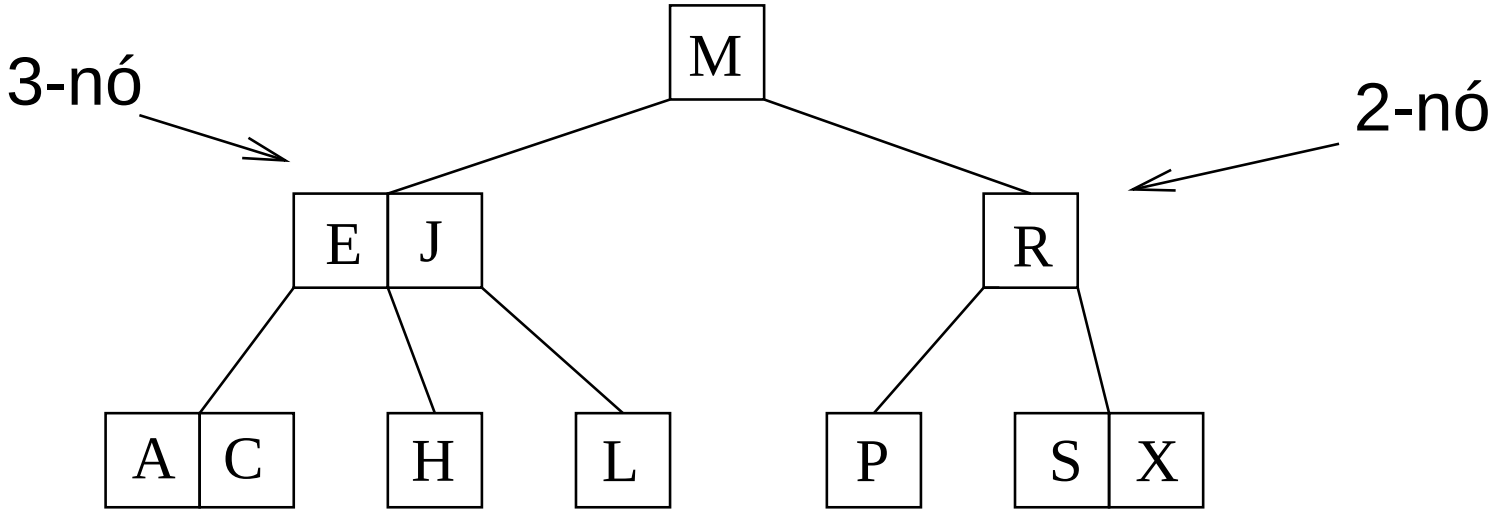
Uma árvore 2-3 é uma árvore vazia ou

- um 2-nó, com uma chave e duas subárvores 2-3;
- um 3-nó, com duas chaves e três filhos;
  filho esquerdo: árvore 2-3 com as chaves menores;
  filho do meio: árvore 2-3 com chaves entre as duas;
  filho direito: árvore 2-3 com as chaves maiores.

Árvores balanceadas

Uma árvore 2-3 é uma árvore vazia ou

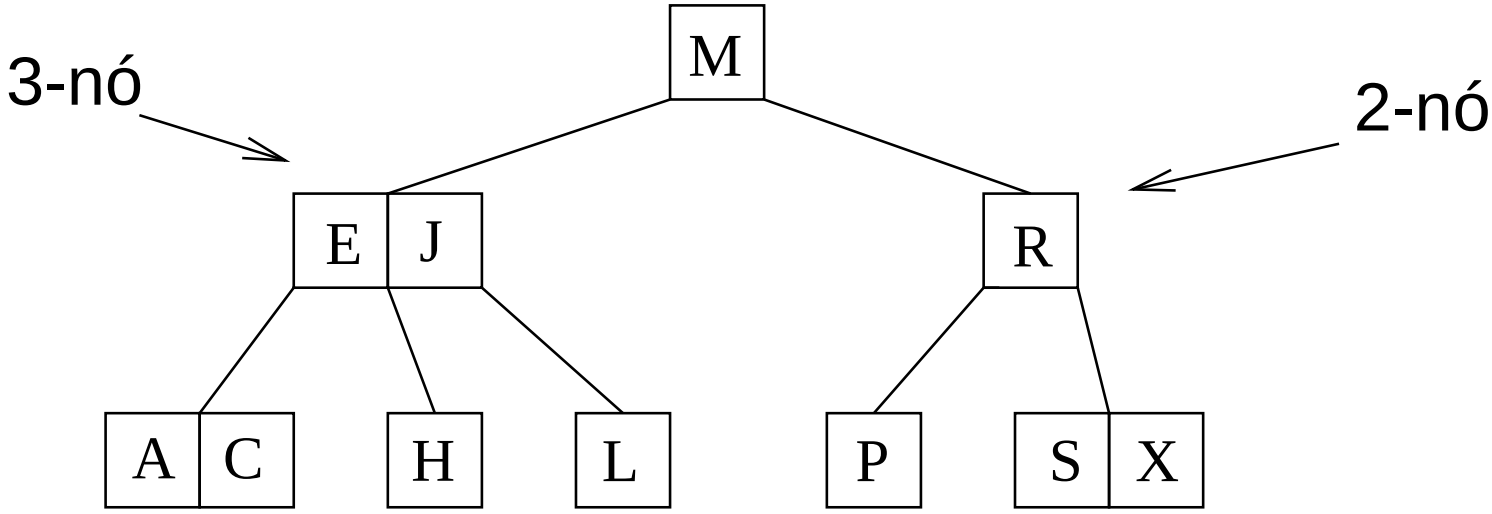
- um 2-nó, com uma chave e duas subárvores 2-3;
- um 3-nó, com duas chaves e três subárvores 2-3;
- todos os apontadores NIL no mesmo nível.



Árvores balanceadas

Uma árvore 2-3 é uma árvore vazia ou

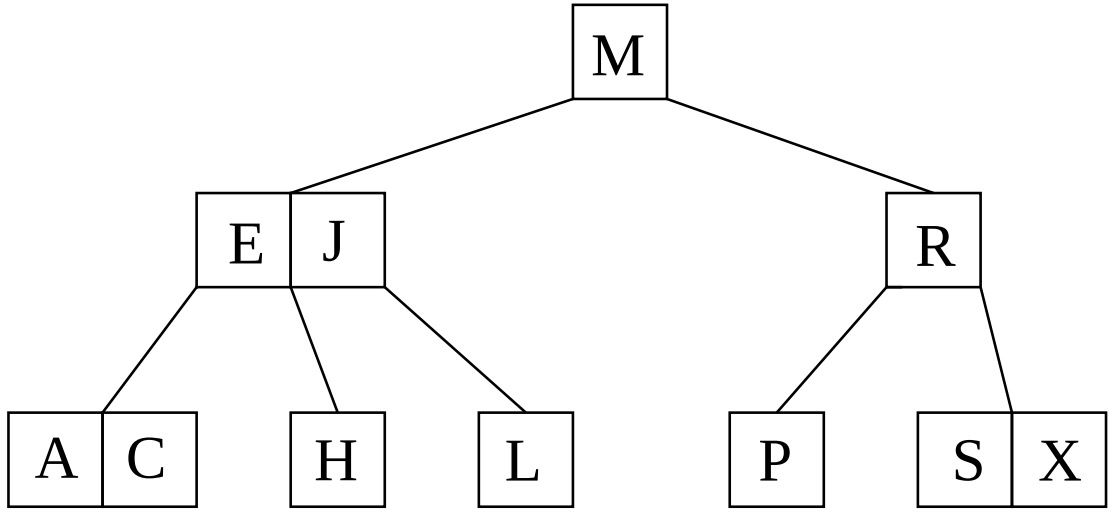
- um 2-nó, com uma chave e duas subárvores 2-3;
- um 3-nó, com duas chaves e três subárvores 2-3;
- todos os apontadores NIL no mesmo nível.



Qual é a altura de uma árvore 2-3 com n chaves?

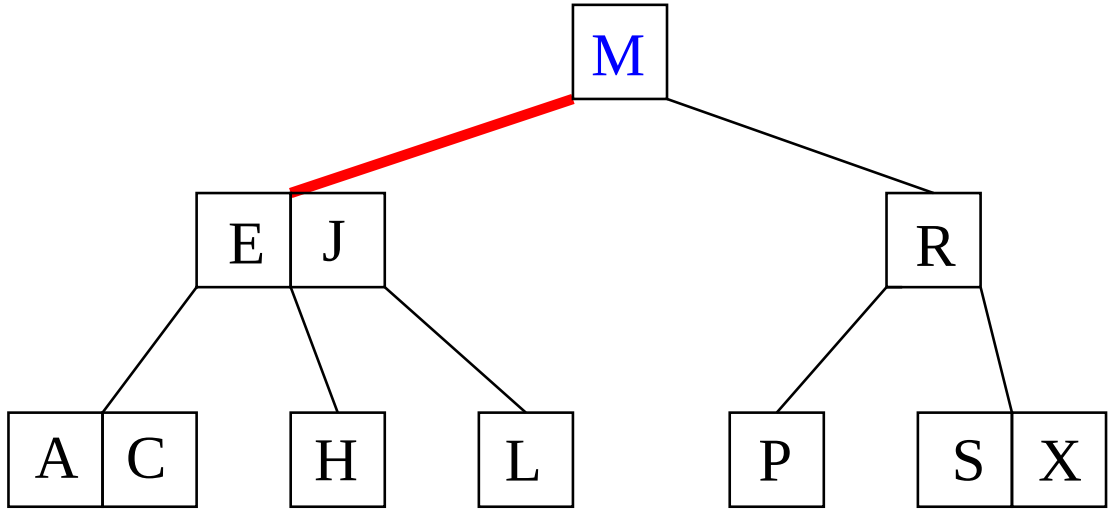
Busca em árvore 2-3

Busca por H.



Busca em árvore 2-3

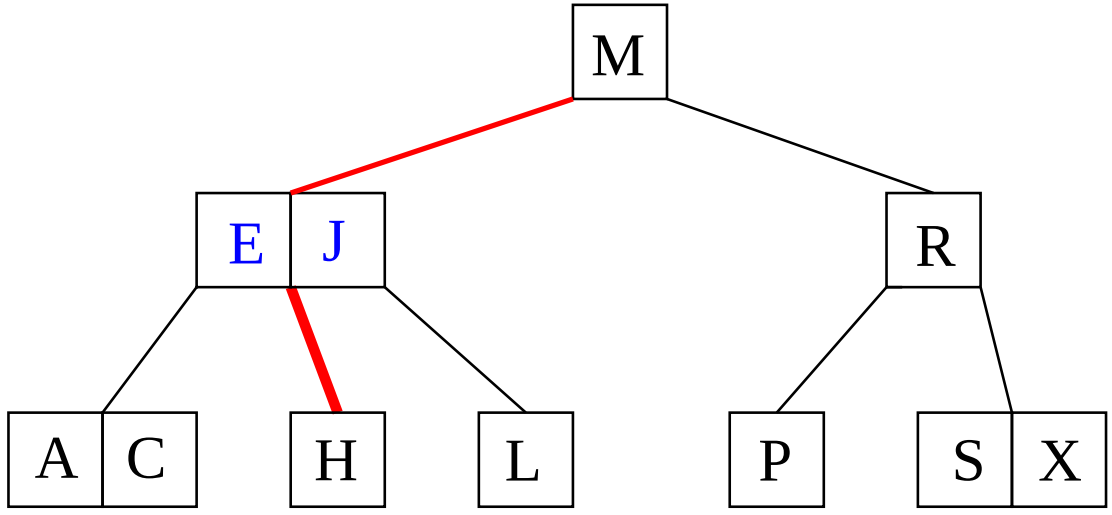
Busca por H.



Compara com M e vai para a esquerda.

Busca em árvore 2-3

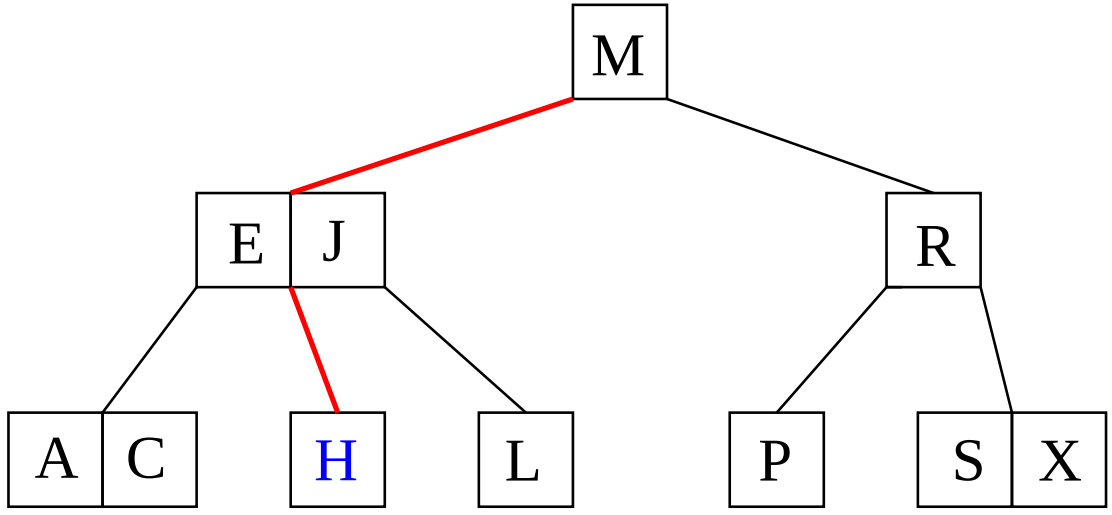
Busca por H.



Compara com E e J e vai pelo meio.

Busca em árvore 2-3

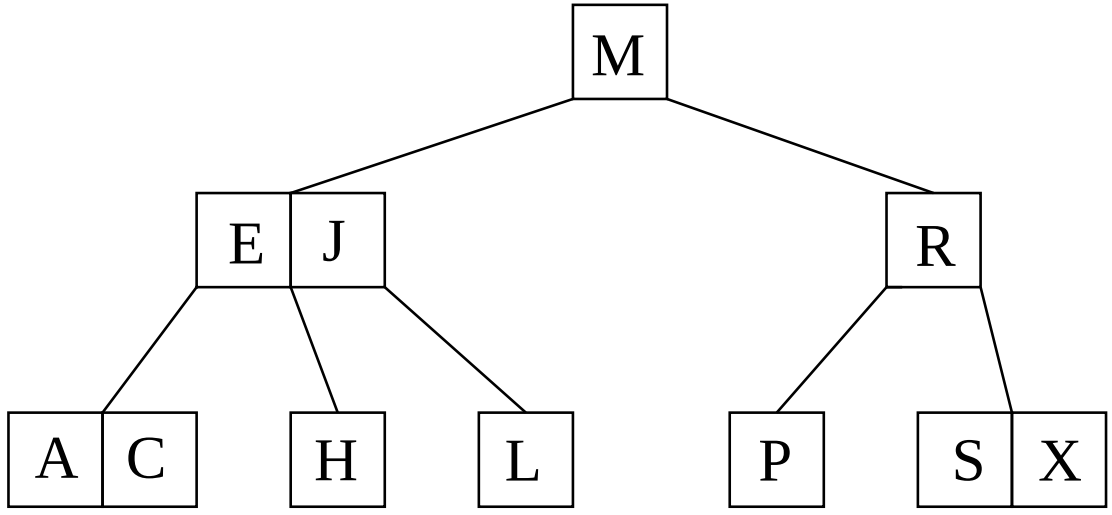
Busca por H.



Encontra H!

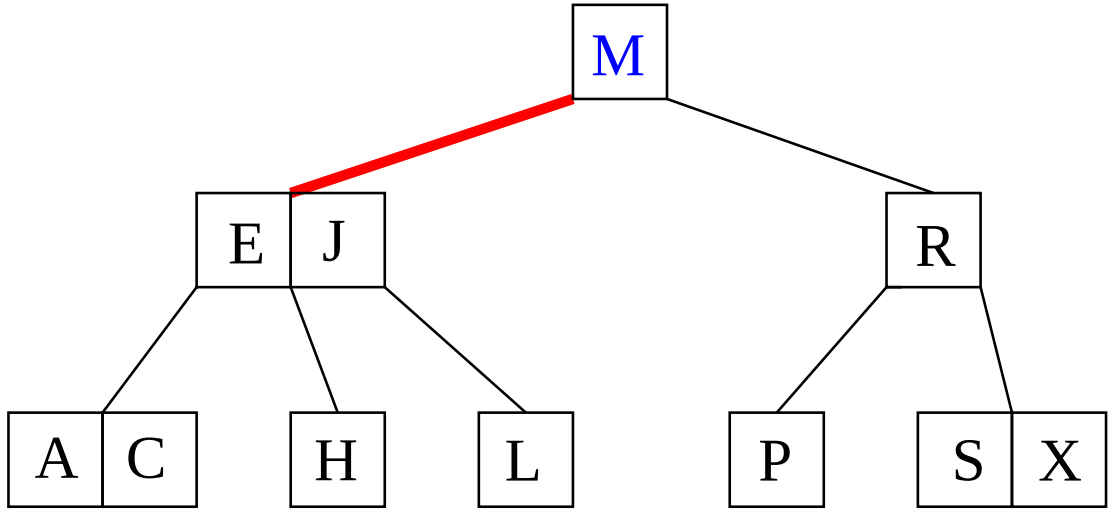
Busca em árvore 2-3

Busca por B.



Busca em árvore 2-3

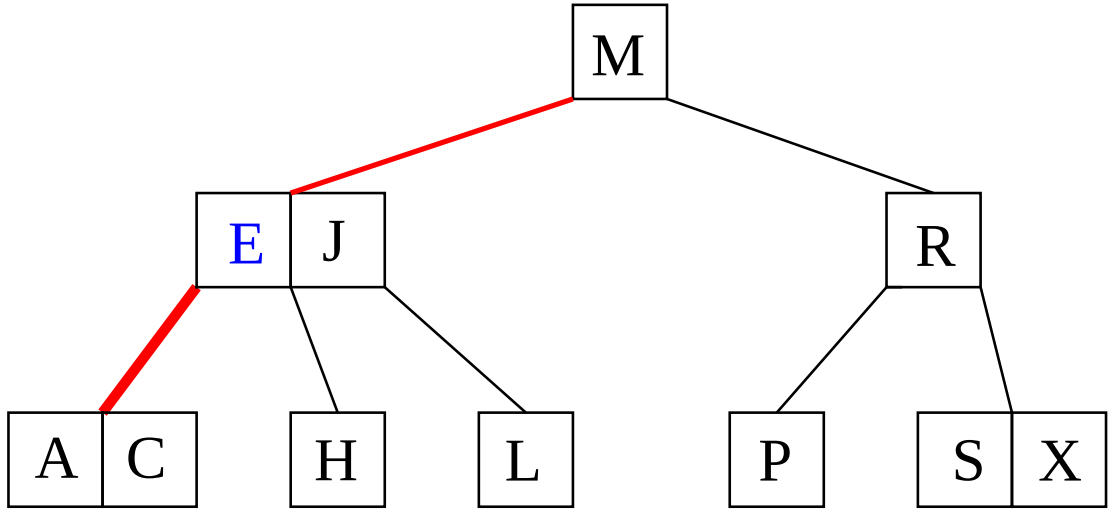
Busca por B.



Compara com M e vai para a esquerda.

Busca em árvore 2-3

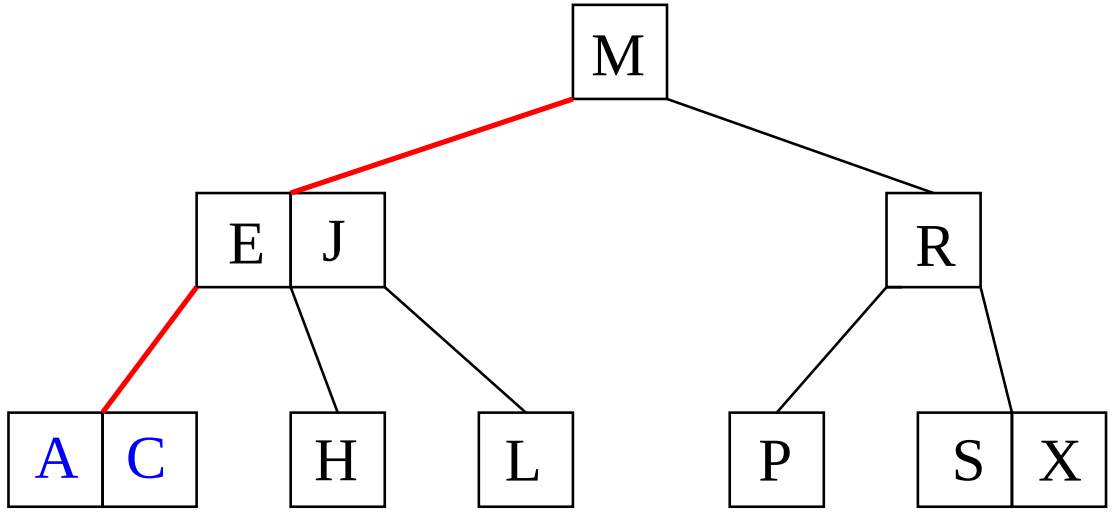
Busca por B.



Compara com E e vai para a esquerda.

Busca em árvore 2-3

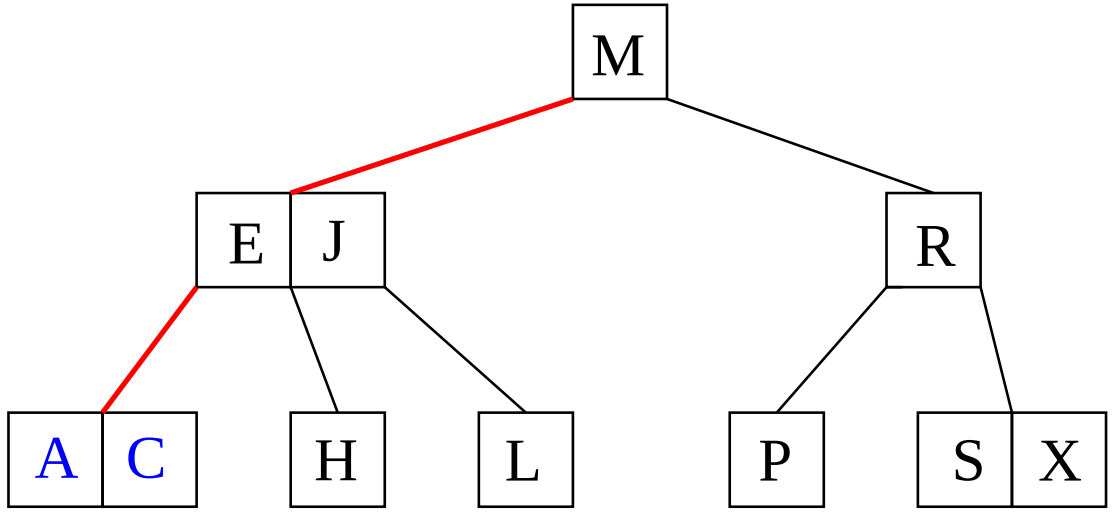
Busca por B.



Compara com A e C e vai pelo meio.

Busca em árvore 2-3

Busca por B.

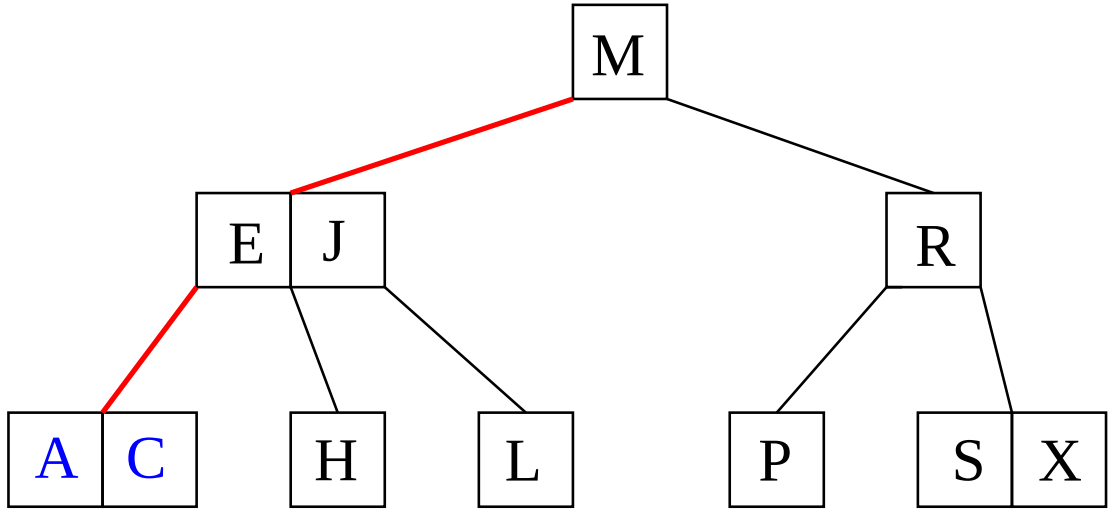


Compara com A e C e vai pelo meio.

Mas o meio é NIL, logo B não está na árvore...

Busca em árvore 2-3

Busca por B.



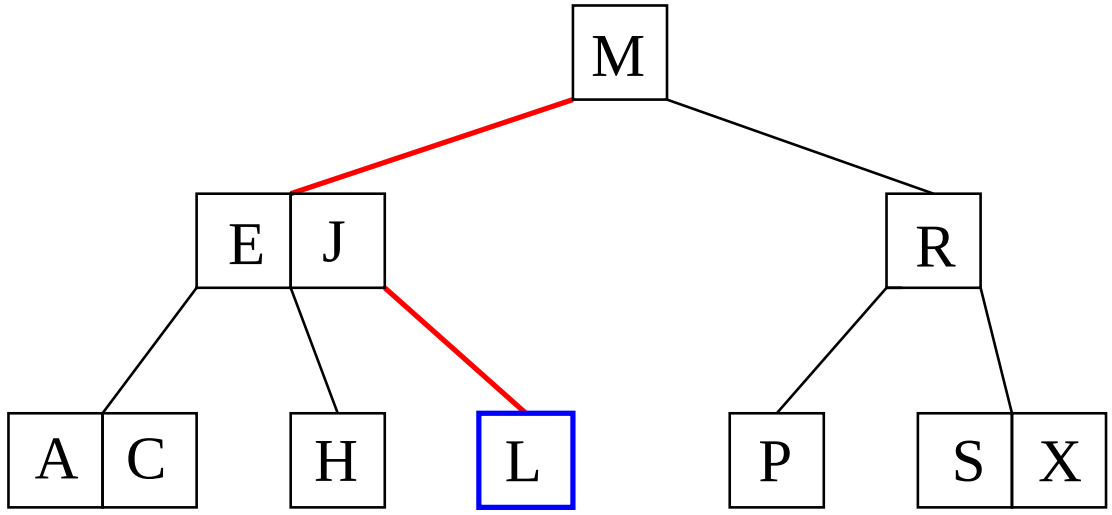
Compara com A e C e vai pelo meio.

Mas o meio é NIL, logo B não está na árvore...

Consumo de tempo: no máximo proporcional a $\lg n$,
onde n é o número de chaves na árvore.

Inserção em árvore 2-3

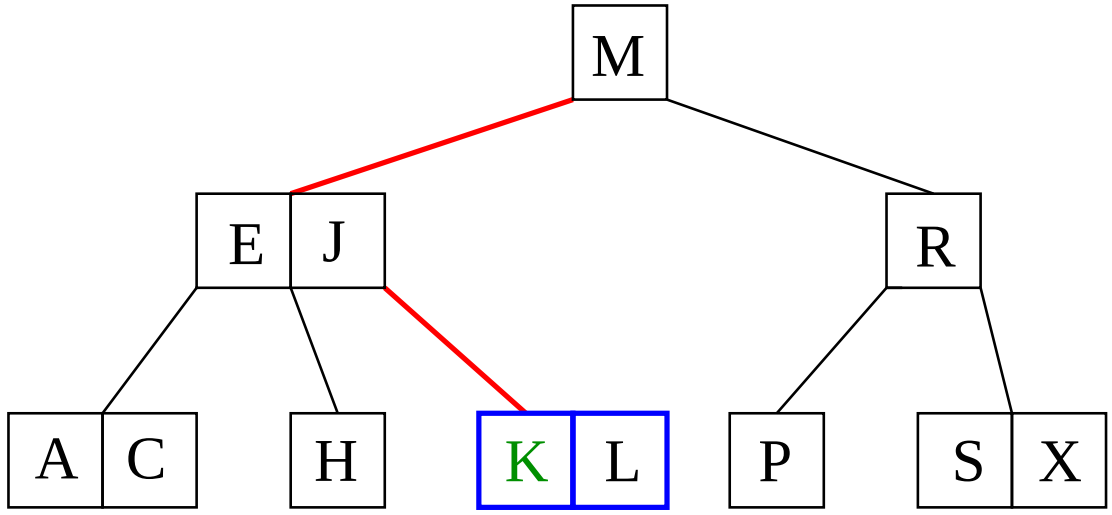
Inserção do K.



Busca pelo K.

Inserção em árvore 2-3

Inserção do K.

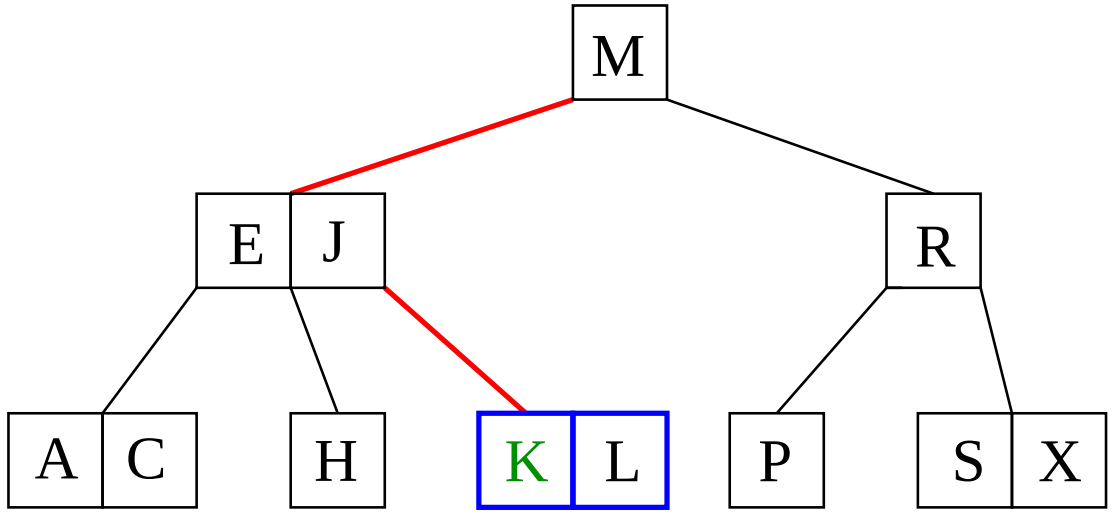


Busca pelo K.

Se a busca termina em um 2-nó,
insere K neste nó.

Inserção em árvore 2-3

Inserção do K.



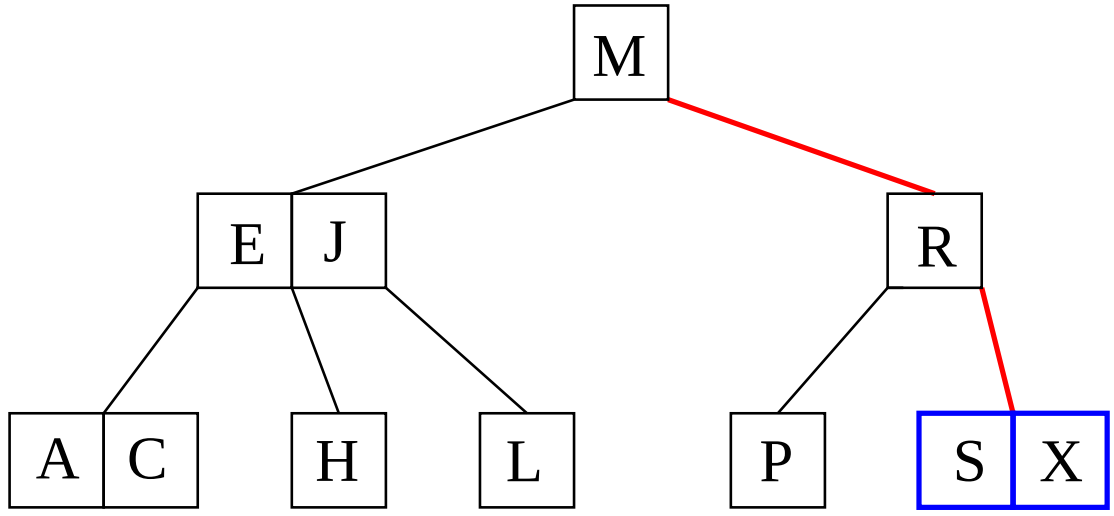
Busca pelo K.

Se a busca termina em um 2-nó,
insere K neste nó.

E se a busca termina num 3-nó?

Inserção em árvore 2-3

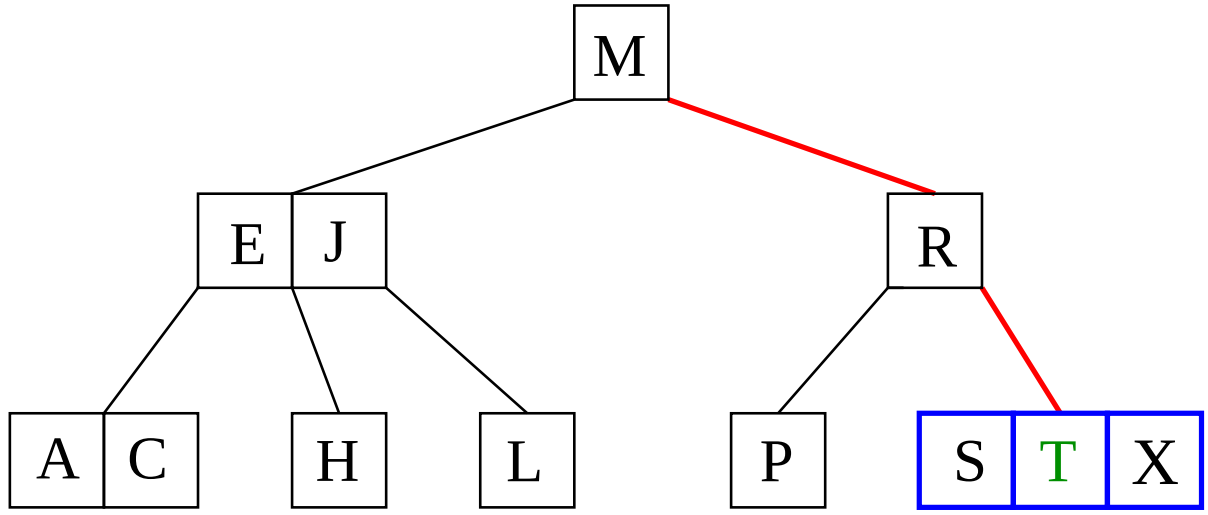
Inserção do T.



Busca pelo T.

Inserção em árvore 2-3

Inserção do T.

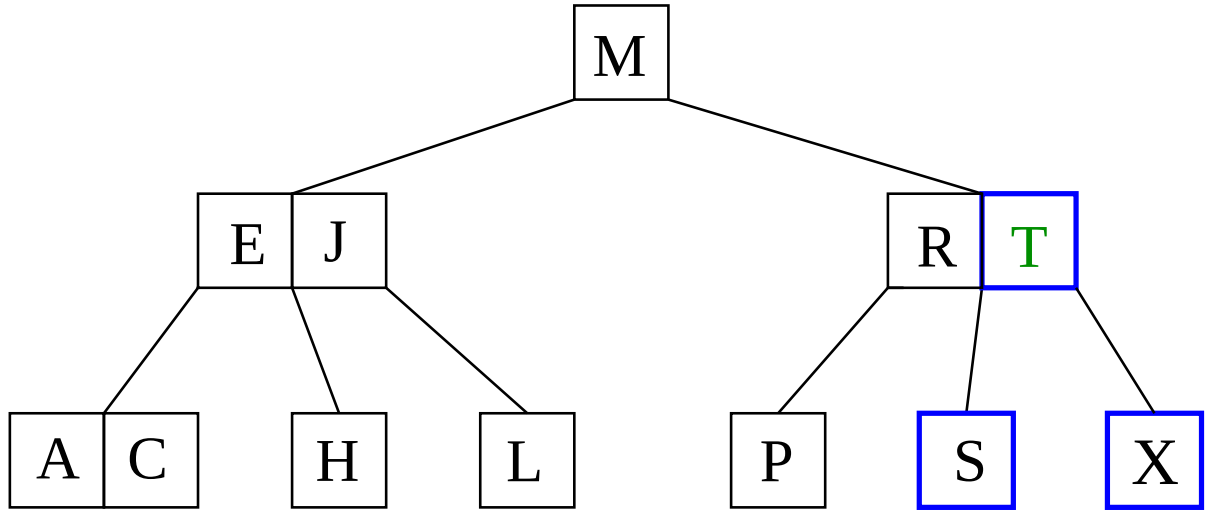


Busca pelo T.

Se a busca termina em um 3-nó,
momentaneamente insere T neste nó, obtendo um “4-nó”.

Inserção em árvore 2-3

Inserção do T.



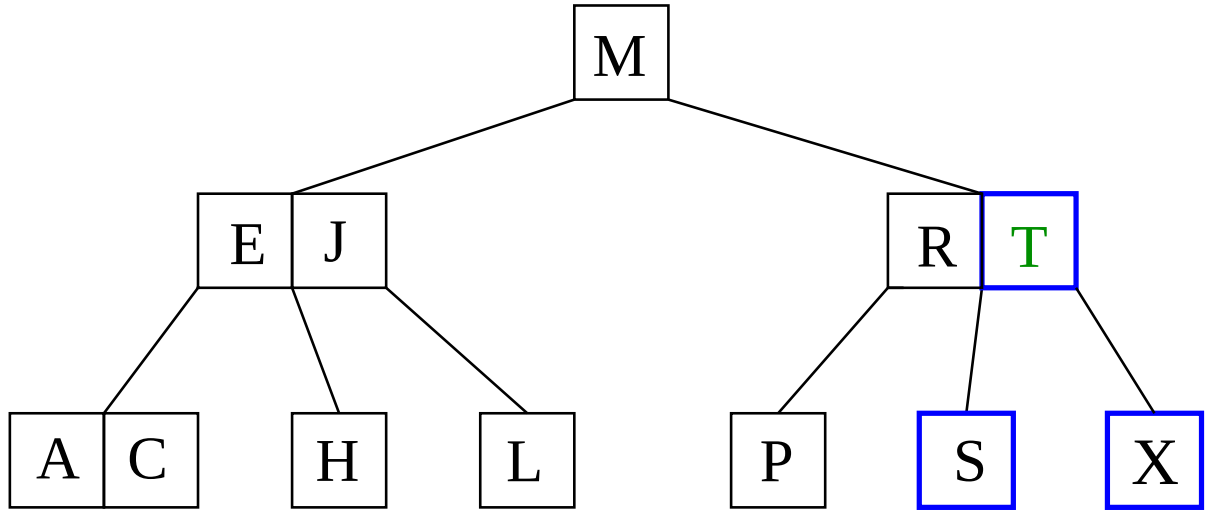
Busca pelo T.

Se a busca termina em um 3-nó,
momentaneamente insere T neste nó, obtendo um 4-nó.

Divide o 4-nó em dois 2-nós,
movendo a chave do meio para cima.

Inserção em árvore 2-3

Inserção do T.

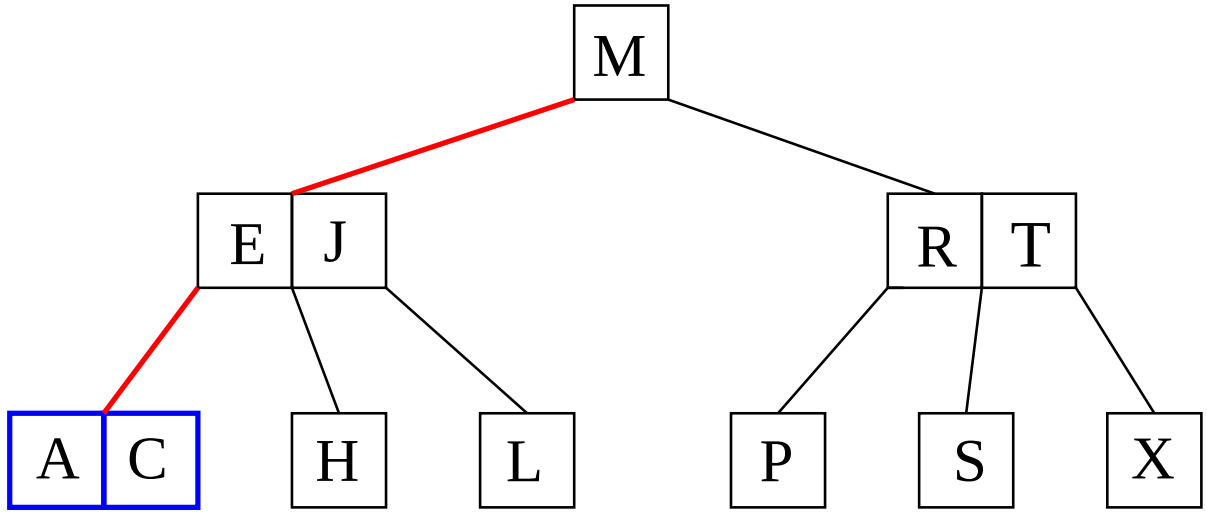


Busca pelo T.
Se a busca termina em um 3-nó,
momentaneamente insere T neste nó, obtendo um 4-nó.
Divide o 4-nó em dois 2-nós,
movendo a chave do meio para cima.

E se o pai do 4-nó é um 3-nó?

Inserção em árvore 2-3

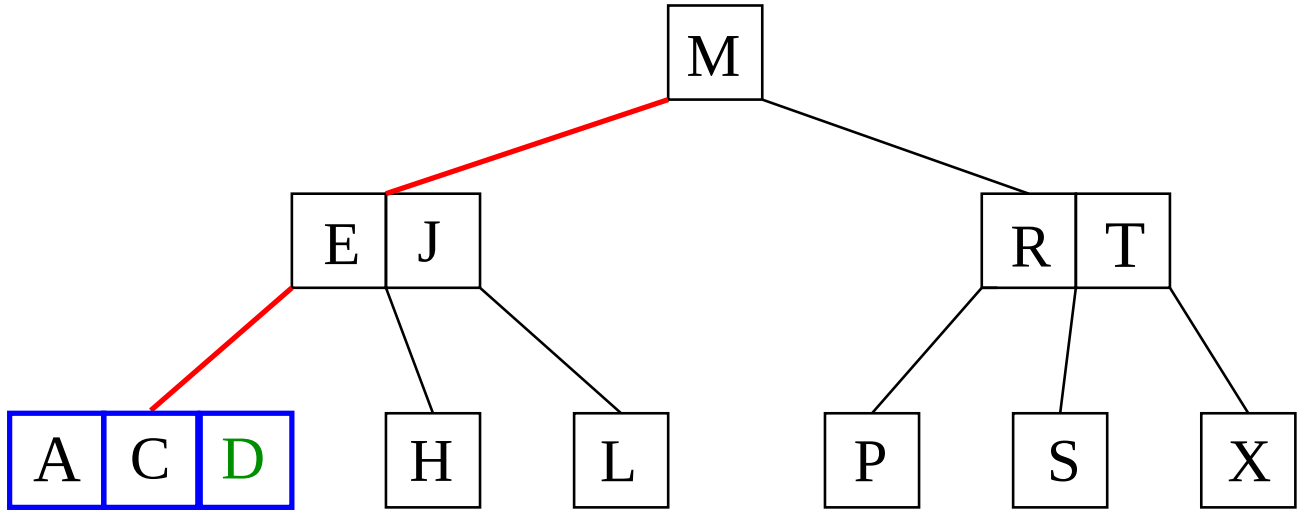
Inserção do D.



Busca pelo D.

Inserção em árvore 2-3

Inserção do D.

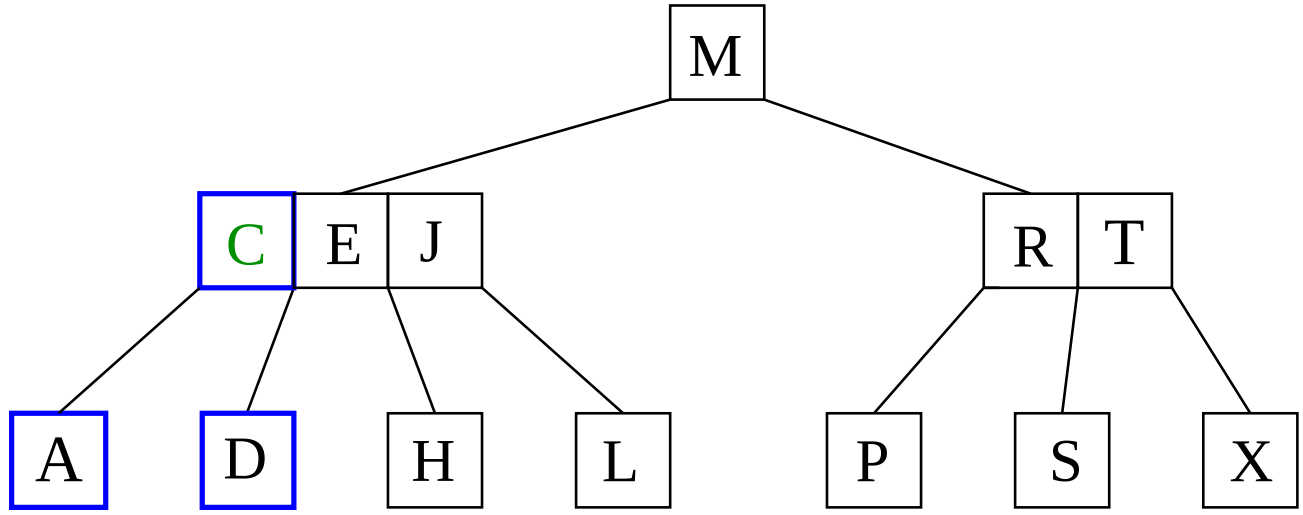


Busca pelo D.

Momentaneamente insere D no 3-nó.

Inserção em árvore 2-3

Inserção do D.



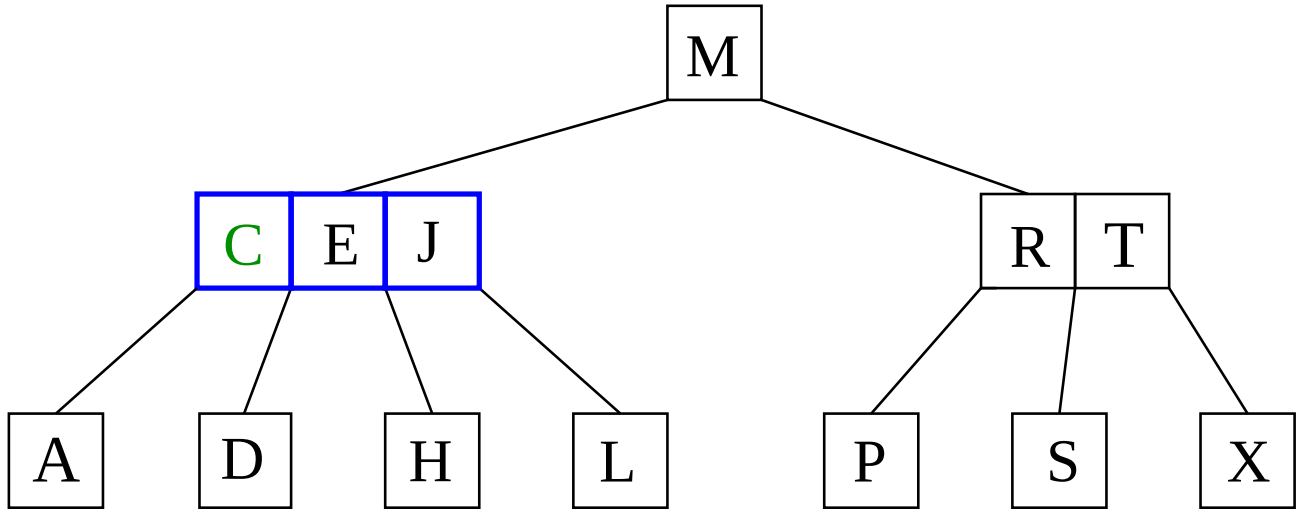
Busca pelo D.

Momentaneamente insere D no 3-nó.

Divide o 4-nó em dois 2-nós,
movendo a chave do meio para cima.

Inserção em árvore 2-3

Inserção do D.

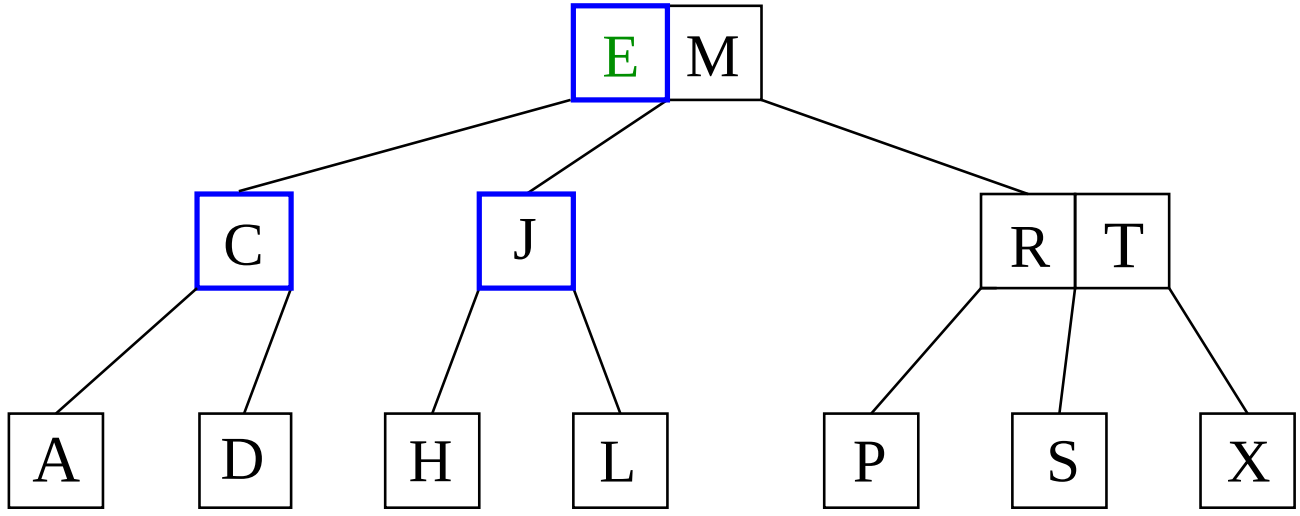


Busca pelo D.
Momentaneamente insere D no 3-nó.
Divide o 4-nó em dois 2-nós,
movendo a chave do meio para cima.

Se o pai virou um 4-nó, repete o processo!

Inserção em árvore 2-3

Inserção do D.

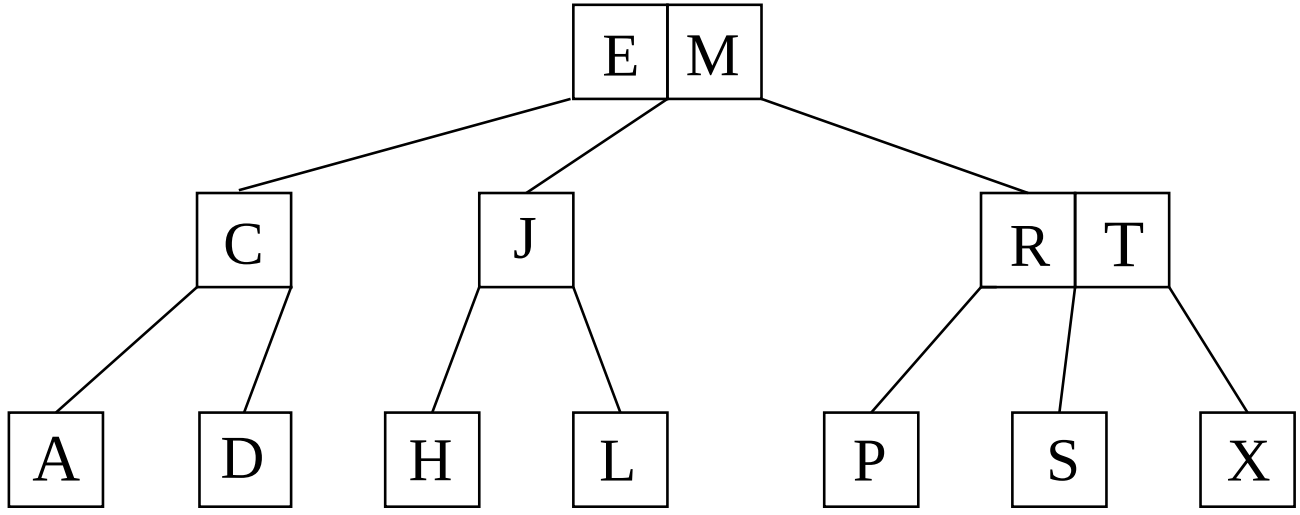


Busca pelo D.
Momentaneamente insere D no 3-nó.
Divide o 4-nó em dois 2-nós,
movendo a chave do meio para cima.

Se o pai virou um 4-nó, repete o processo!

Inserção em árvore 2-3

Inserção do D.

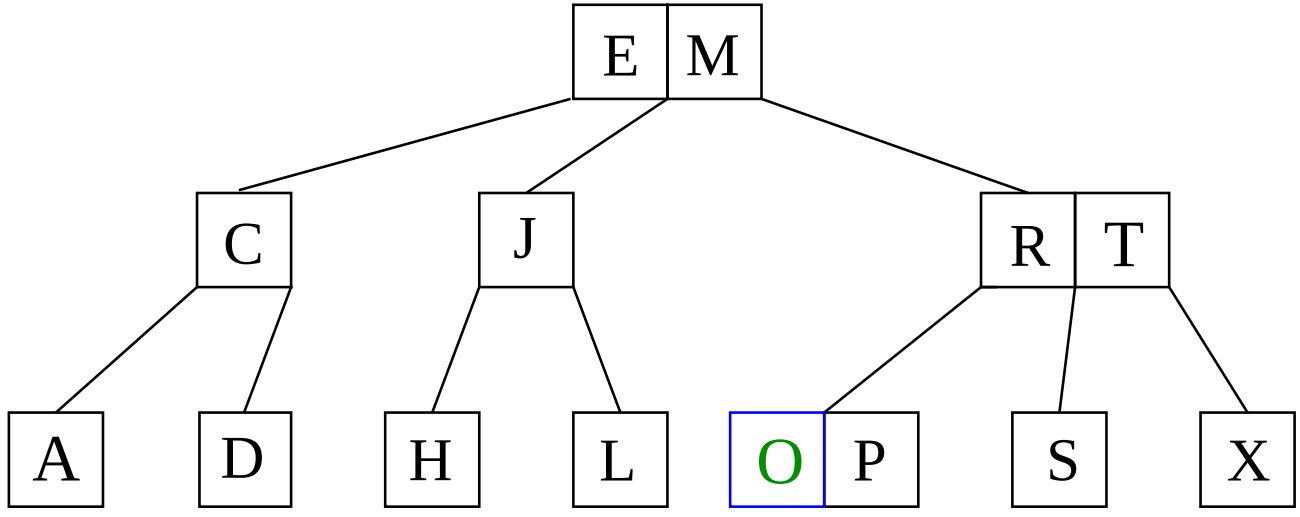


Busca pelo D.
Momentaneamente insere D no 3-nó.
Divide o 4-nó em dois 2-nós,
movendo a chave do meio para cima.

E se a raiz virar um 4-nó?

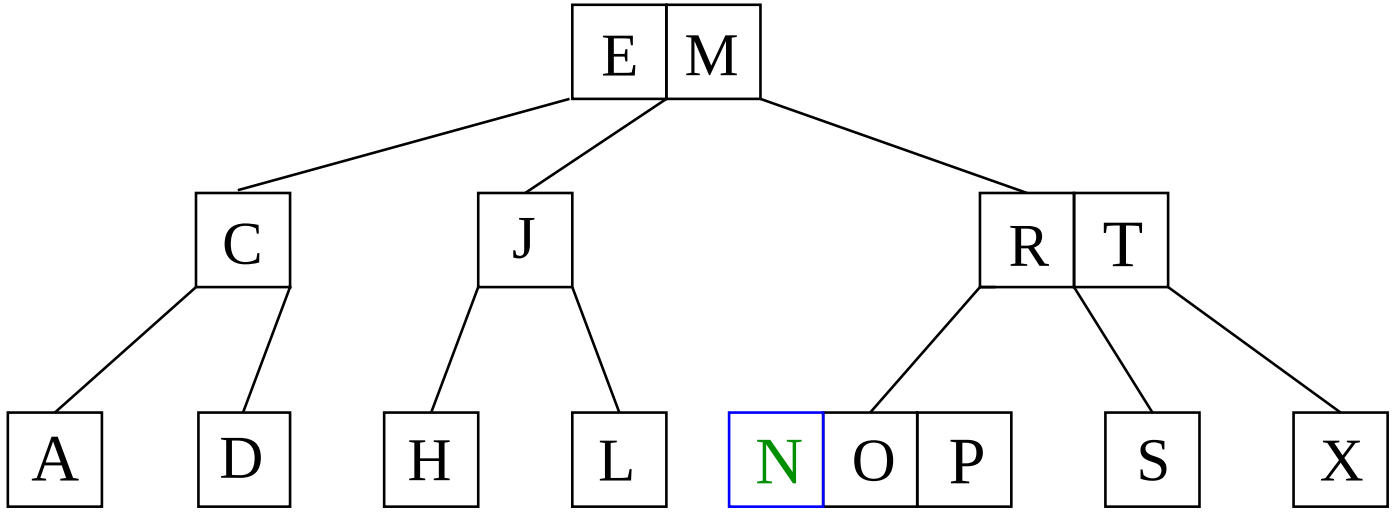
Inserção em árvore 2-3

Inserção do O...



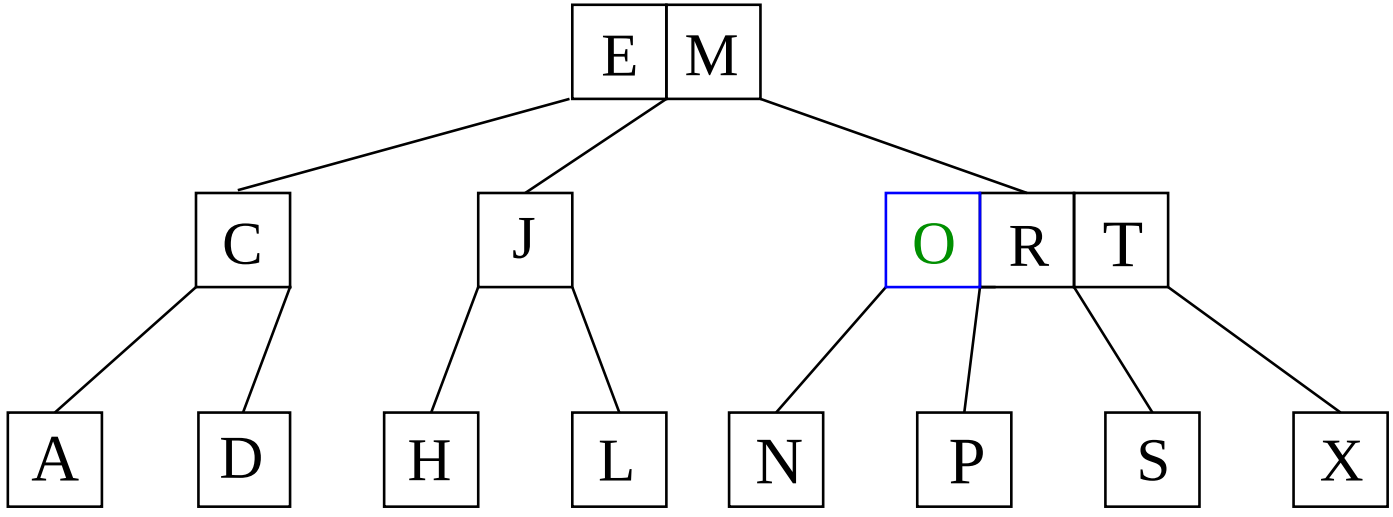
Inserção em árvore 2-3

Inserção do O... Em seguinda, inserção do N.



Inserção em árvore 2-3

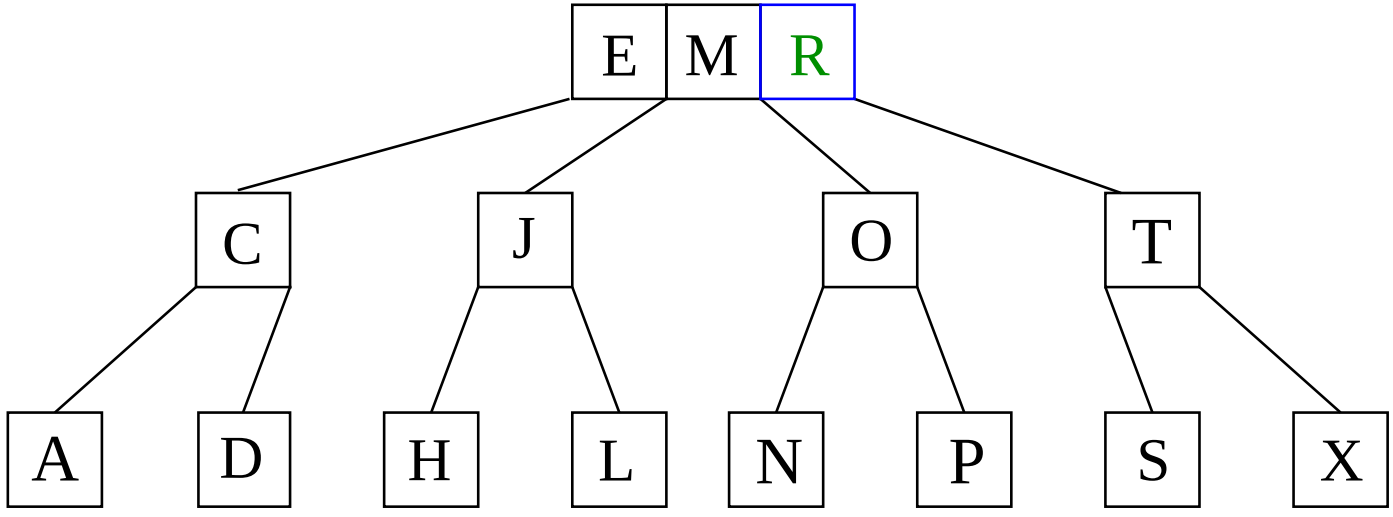
Inserção do O... Em seguinda, inserção do N.



Divide o 4-nó...

Inserção em árvore 2-3

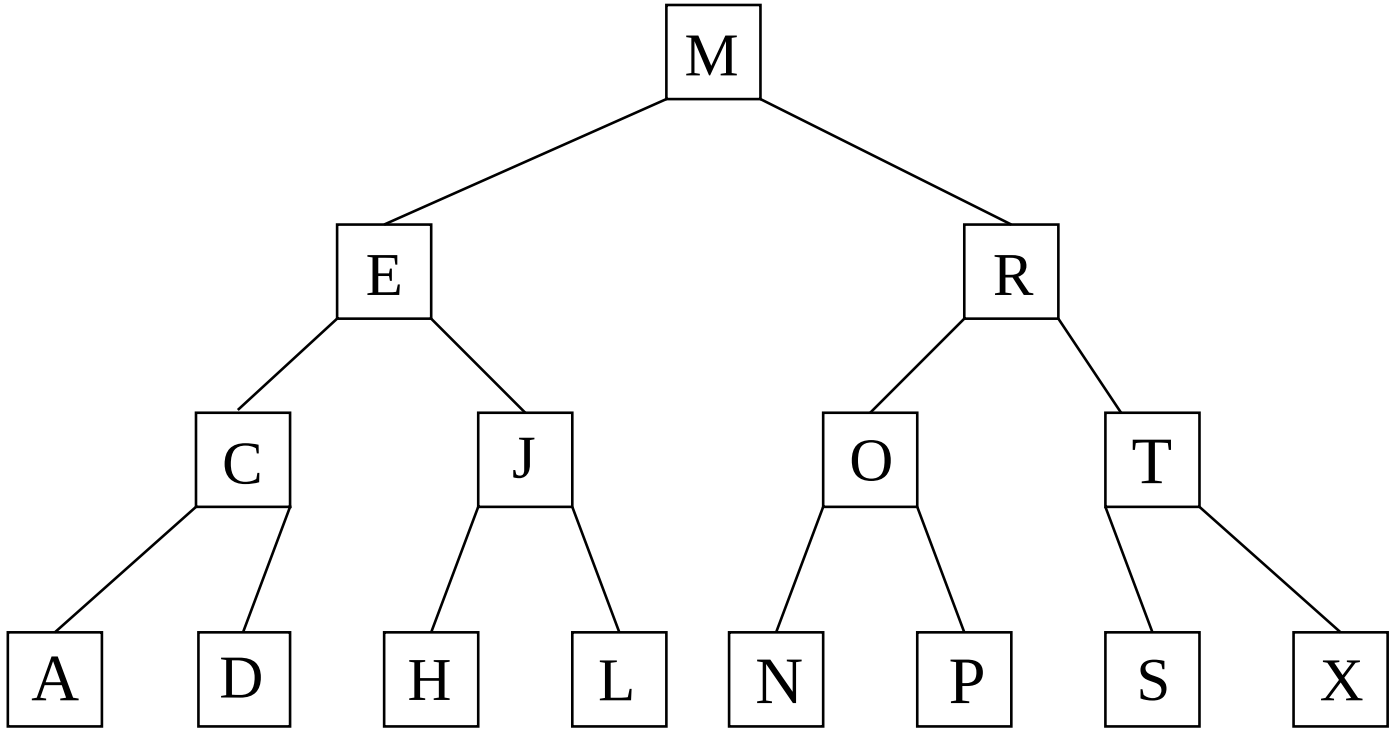
Inserção do O... Em seguinda, inserção do N.



Divide o 4-nó... Divide esse outro 4-nó...

Inserção em árvore 2-3

Inserção do O... Em seguinda, inserção do N.

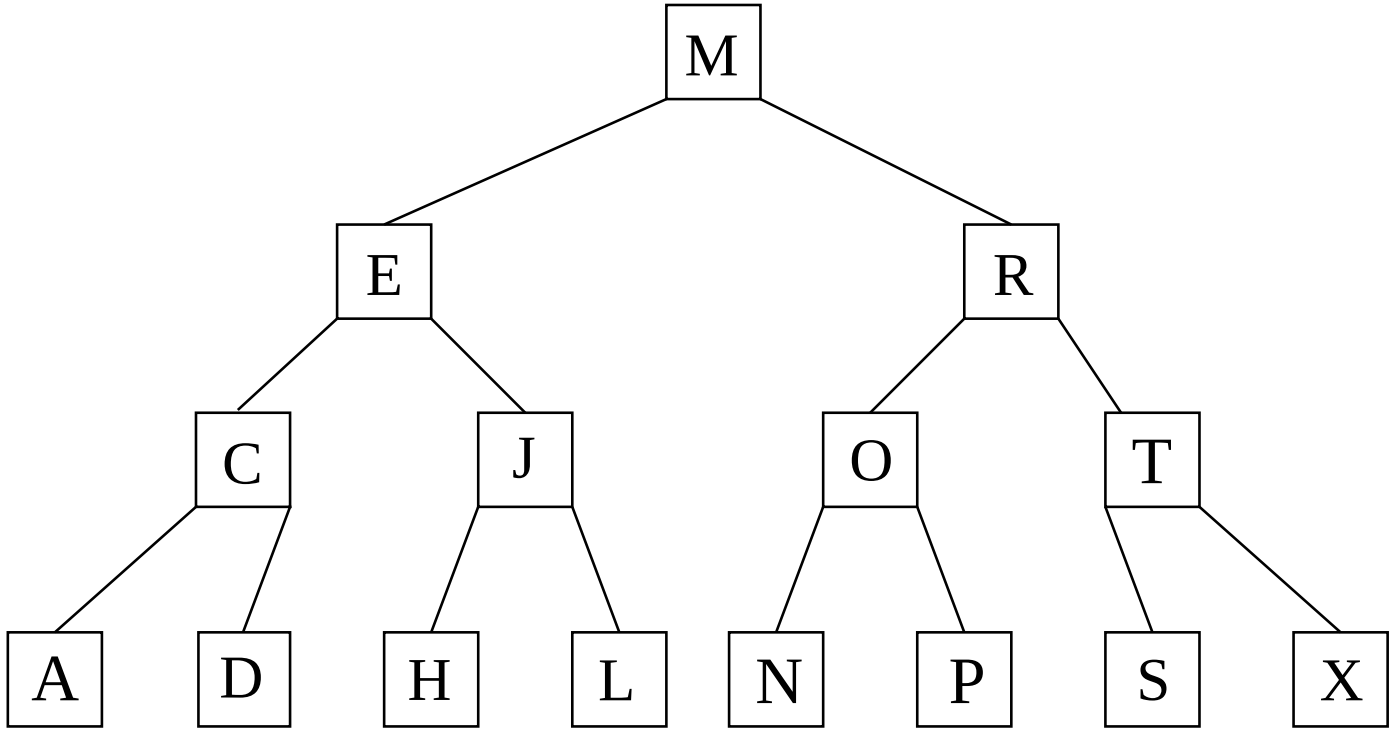


Divide o 4-nó... Divide esse outro 4-nó...

Divide a raiz!

Inserção em árvore 2-3

Inserção do O... Em seguinda, inserção do N.



Divide o 4-nó... Divide esse outro 4-nó...

Divide a raiz! É assim que as árvores 2-3 crescem!

Árvores 2-3 e ABBs rubro-negras

Implementar árvores 2-3 diretamente envolve lidar com nós de tamanhos diferentes, e por isso pode ser não trivial.

Árvores 2-3 e ABBs rubro-negras

Implementar árvores 2-3 diretamente envolve lidar com nós de tamanhos diferentes, e por isso pode ser não trivial.

Árvores rubro-**negras** são
árvores binárias que representam árvores 2-3.

Árvores 2-3 e ABBs rubro-negras

Implementar árvores 2-3 diretamente envolve lidar com nós de tamanhos diferentes, e por isso pode ser não trivial.

Árvores rubro-**negras** são
árvores binárias que representam árvores 2-3.

Árvores binárias são como árvores 2-3 com apenas 2-nós.
Sua manipulação é mais simples.

Árvores 2-3 e ABBs rubro-negras

Implementar árvores 2-3 diretamente envolve lidar com nós de tamanhos diferentes, e por isso pode ser não trivial.

Árvores rubro-**negras** são
árvores binárias que representam árvores 2-3.

Árvores binárias são como árvores 2-3 com apenas 2-nós.
Sua manipulação é mais simples.

Por serem representações de árvores 2-3,
elas são balanceadas. (Tem altura logarítmica.)

Árvores 2-3 e ABBs rubro-negras

Implementar árvores 2-3 diretamente envolve lidar com nós de tamanhos diferentes, e por isso pode ser não trivial.

Árvores rubro-**negras** são
árvores binárias que representam árvores 2-3.

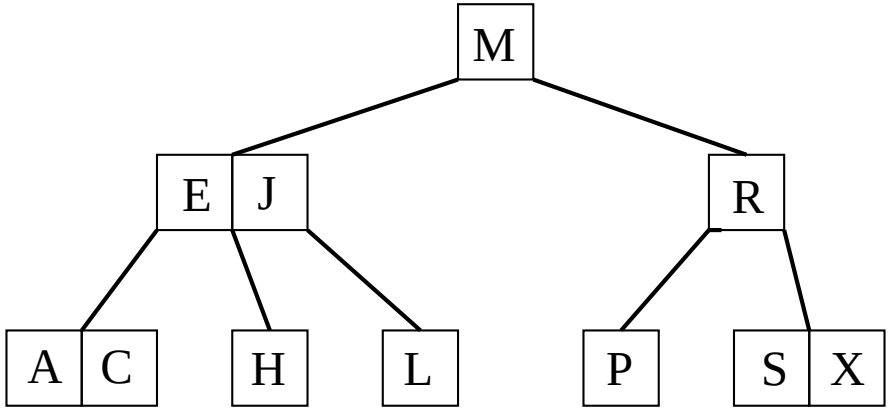
Árvores binárias são como árvores 2-3 com apenas 2-nós.
Sua manipulação é mais simples.

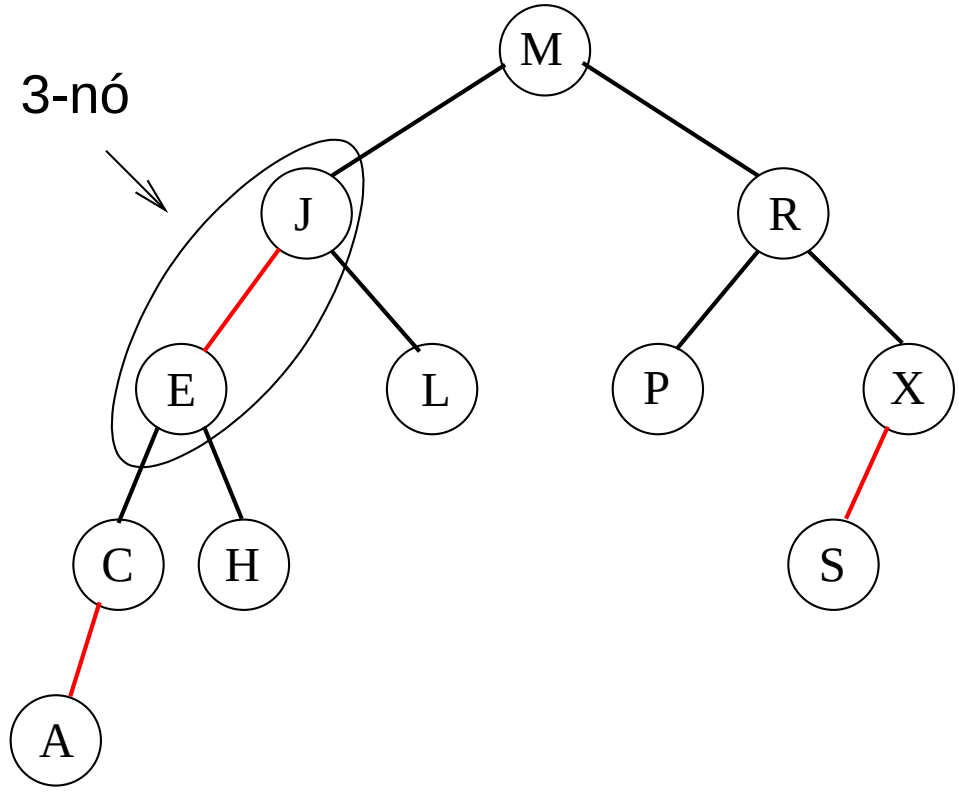
Por serem representações de árvores 2-3,
elas são balanceadas. (Tem altura logarítmica.)

As operações em árvores rubro-**negras**
consomem tempo no máximo proporcional a $\lg n$,
onde n é o número de chaves na árvore.

Árvores 2-3 e ABBs rubro-negras

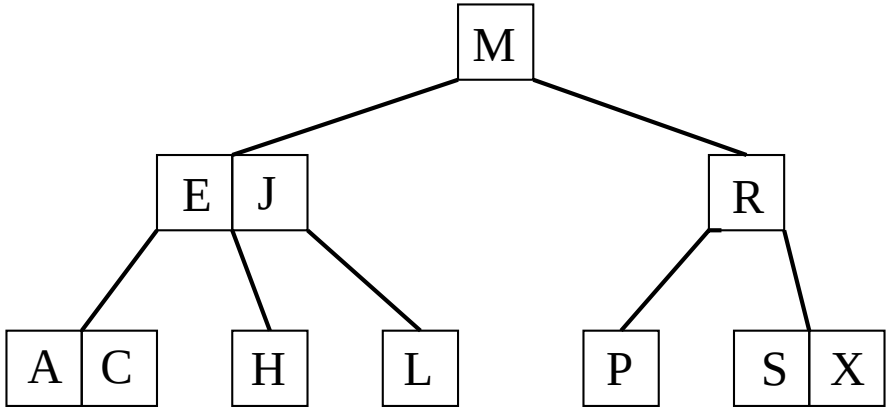
3-nós são representados por:
dois 2-nós, ligados por uma aresta vermelha.

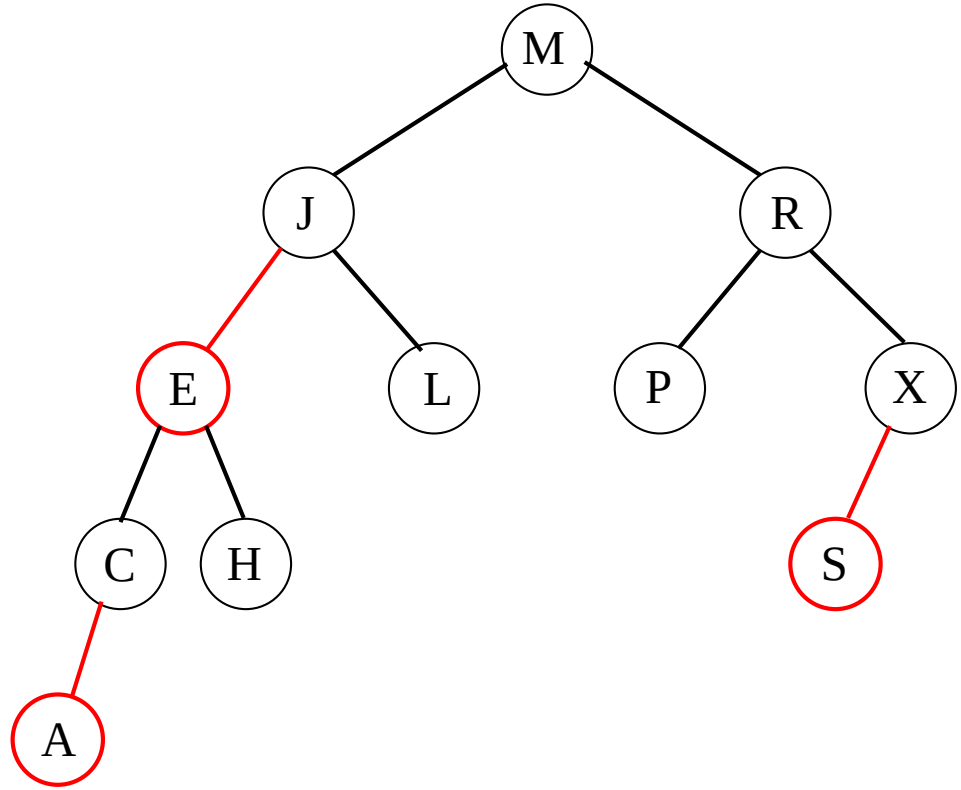




Árvores 2-3 e ABBs rubro-negras

3-nós são representados por:
dois 2-nós, ligados por uma aresta vermelha.

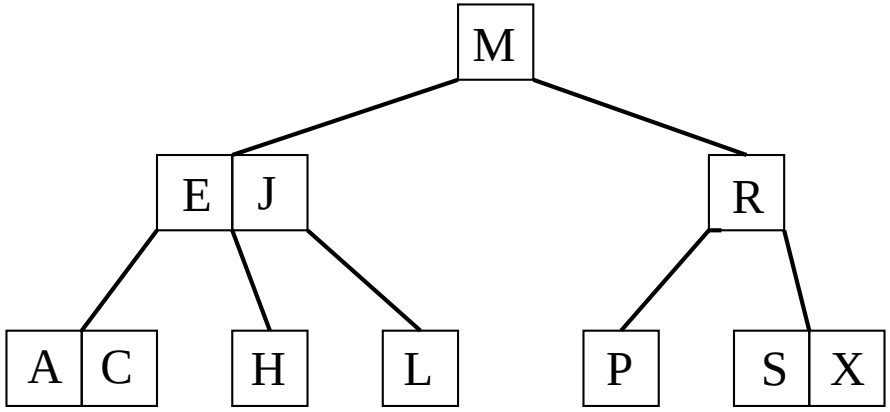


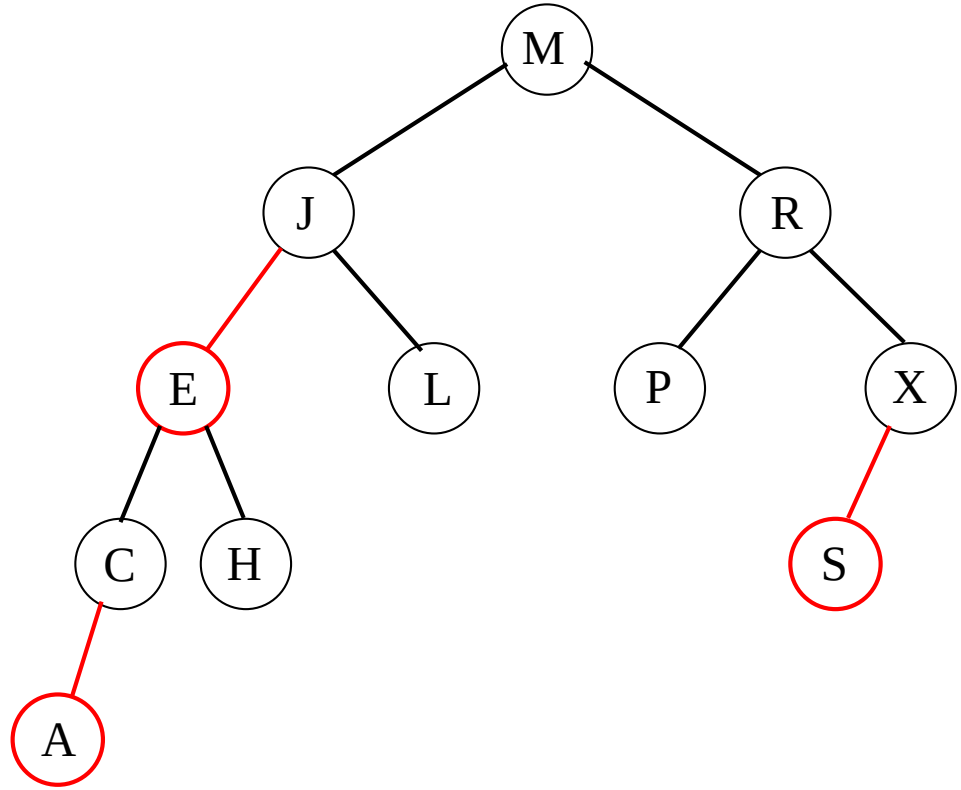


Nós são rubros ou **negros**.

Árvores 2-3 e ABBs rubro-negras

3-nós são representados por:
dois 2-nós, ligados por uma aresta vermelha.





Nós são rubros ou **negros**.

Bijeção entre árvores 2-3 e ABBs rubro-negras!

ABBs rubro-negras

Uma ABB é rubro-**negra** se

1. todo nó é rubro ou **negro**
2. toda folha (NIL) é **negra**
3. se um nó é rubro,
   então é filho esquerdo de um nó **negro**
4. todo caminho de um nó x até uma folha sua descendente tem o mesmo número de nós **negros**.

ABBs rubro-negras

Uma ABB é rubro-**negra** se

1. todo nó é rubro ou **negro**
2. toda folha (NIL) é **negra**
3. se um nó é rubro,
   então é filho esquerdo de um nó **negro**
4. todo caminho de um nó x até uma folha sua descendente tem o mesmo número de nós **negros**.

$rn(x)$: número de nós **negros** no caminho de um filho de x até uma folha descendente de x.

ABBs rubro-negras

Uma ABB é rubro-**negra** se

1. todo nó é rubro ou **negro**
2. toda folha (NIL) é **negra**
3. se um nó é rubro,
   então é filho esquerdo de um nó **negro**
4. todo caminho de um nó x até uma folha sua descendente tem o mesmo número de nós **negros**.

$rn(x)$: número de nós **negros** no caminho de um filho de x até uma folha descendente de x.

Lema: Uma ABB rubro-**negra** com n nós internos tem altura no máximo $2\lg(n+1)$.

ABBs rubro-negras

Busca

Como fazer busca numa ABB rubro-negra?

ABBs rubro-negras

Busca

Como fazer busca numa ABB rubro-negra?

Igual à busca em ABB!

ABBs rubro-negras

Busca

Como fazer busca numa ABB rubro-negra?

Igual à busca em ABB!

Idem para mínimo, e percursos em geral.

ABBs rubro-negras

Busca

Como fazer busca numa ABB rubro-negra?

Igual à busca em ABB!

Idem para mínimo, e percursos em geral.

O que muda então?

ABBs rubro-negras

Busca

Como fazer busca numa ABB rubro-negra?

Igual à busca em ABB!

Idem para mínimo, e percursos em geral.

O que muda então?

Inserção e remoção.

ABBs rubro-negras

Busca

Como fazer busca numa ABB rubro-negra?

Igual à busca em ABB!

Idem para mínimo, e percursos em geral.

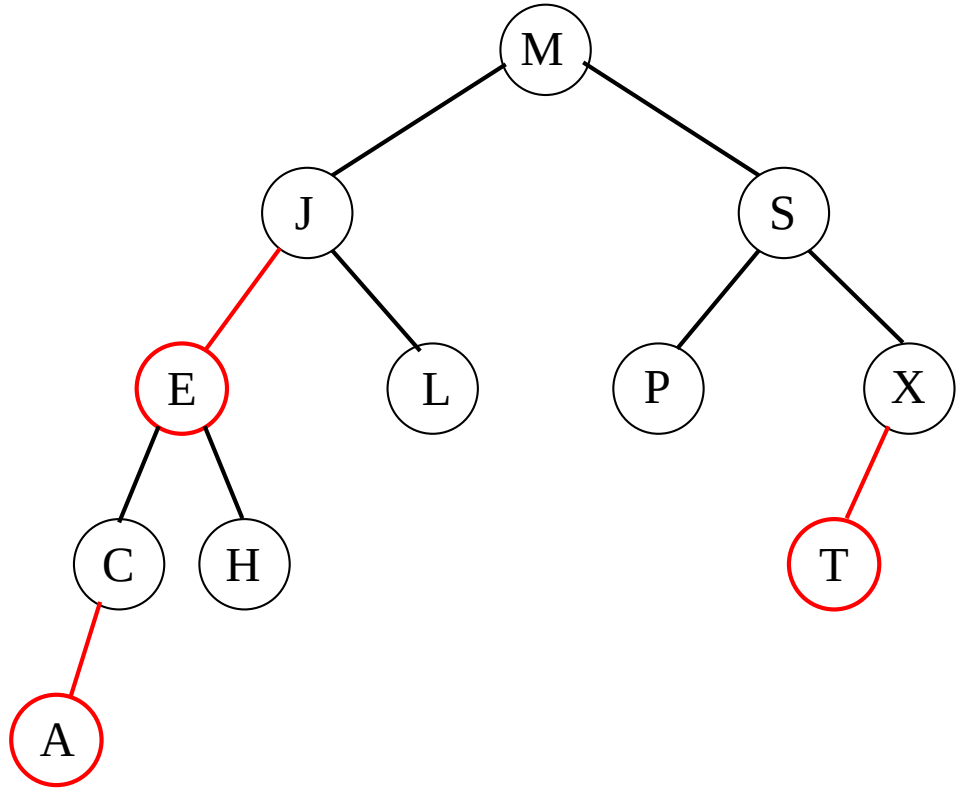
O que muda então?

Inserção e remoção.

Ideia: Simular inserção e remoção em árvores 2-3!

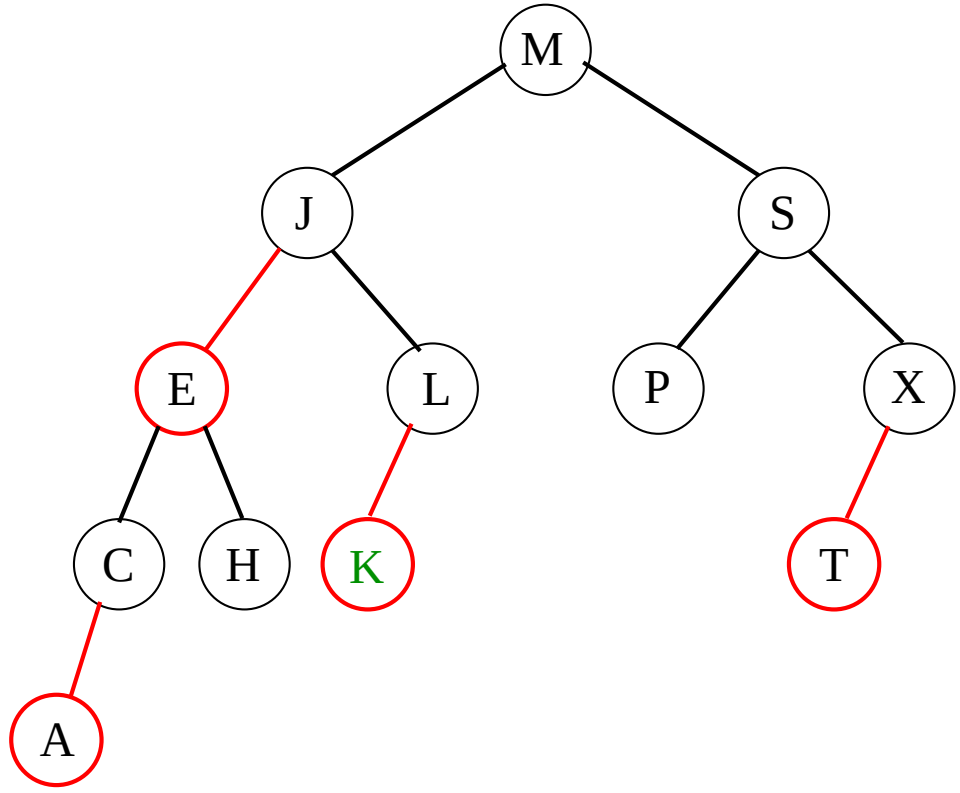
Inserção em ABB rubro-negra

Inserir o K.



Inserção em ABB rubro-negra

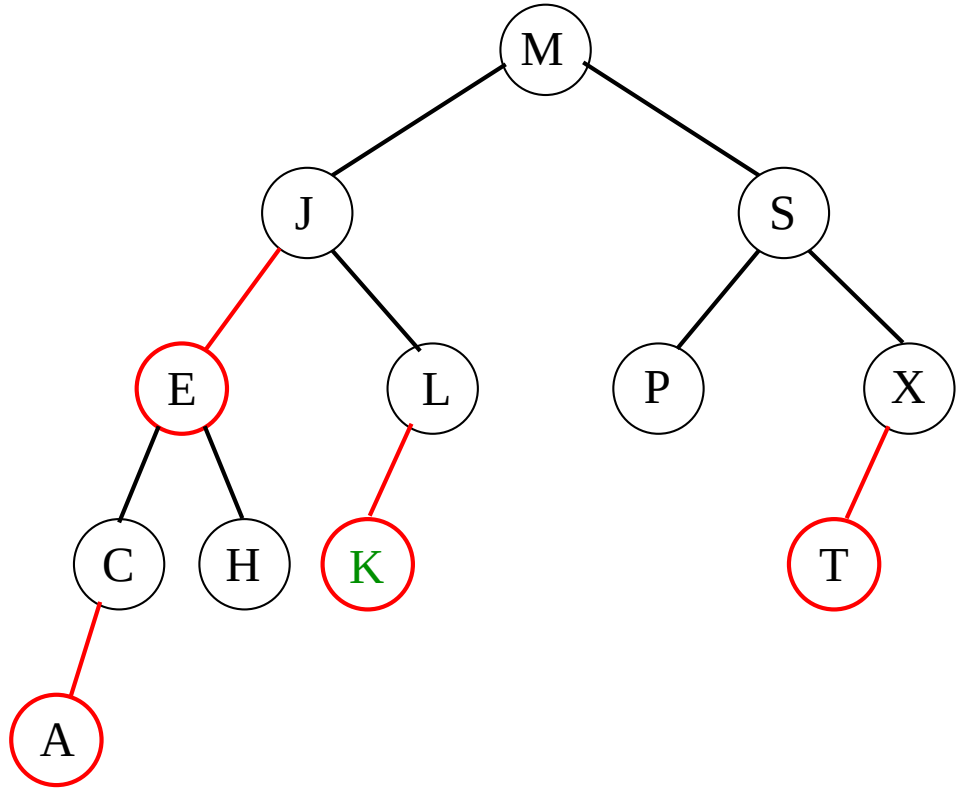
Inserir o K.



Insere K como um nó vermelho.

Inserção em ABB rubro-negra

Inserir o K.

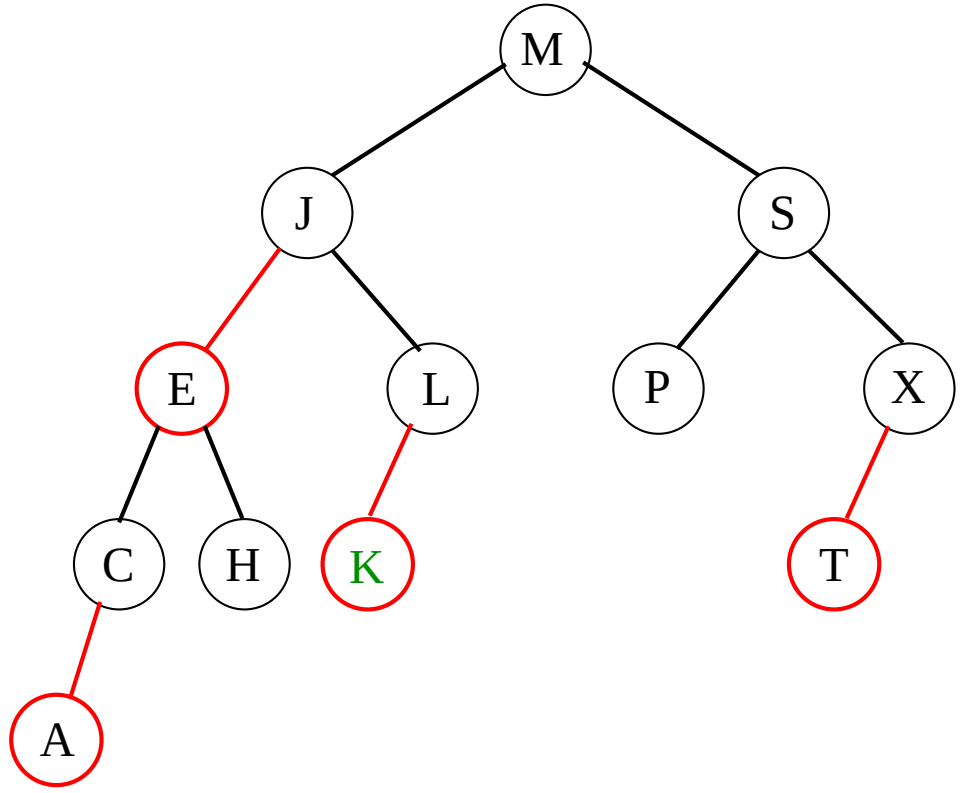


Insere K como um nó vermelho.

Se o pai de K não é vermelho, terminou!

Inserção em ABB rubro-negra

Inserir o K.

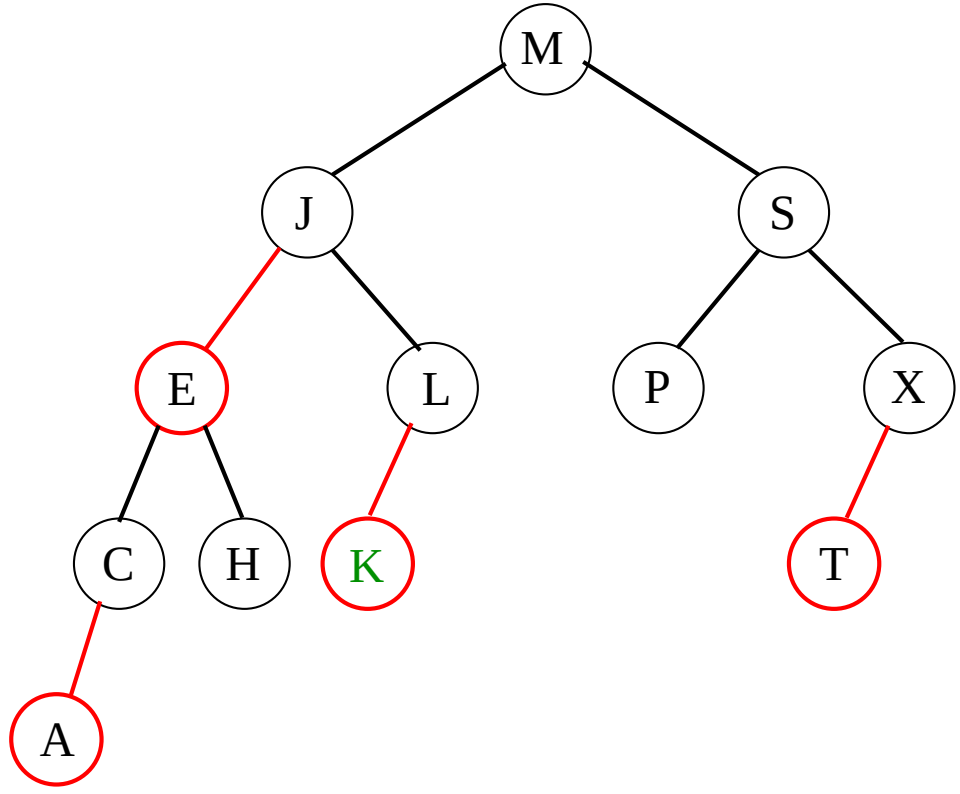


Insere K como um nó vermelho.

Se o pai de K não é vermelho, terminou! Nem sempre...

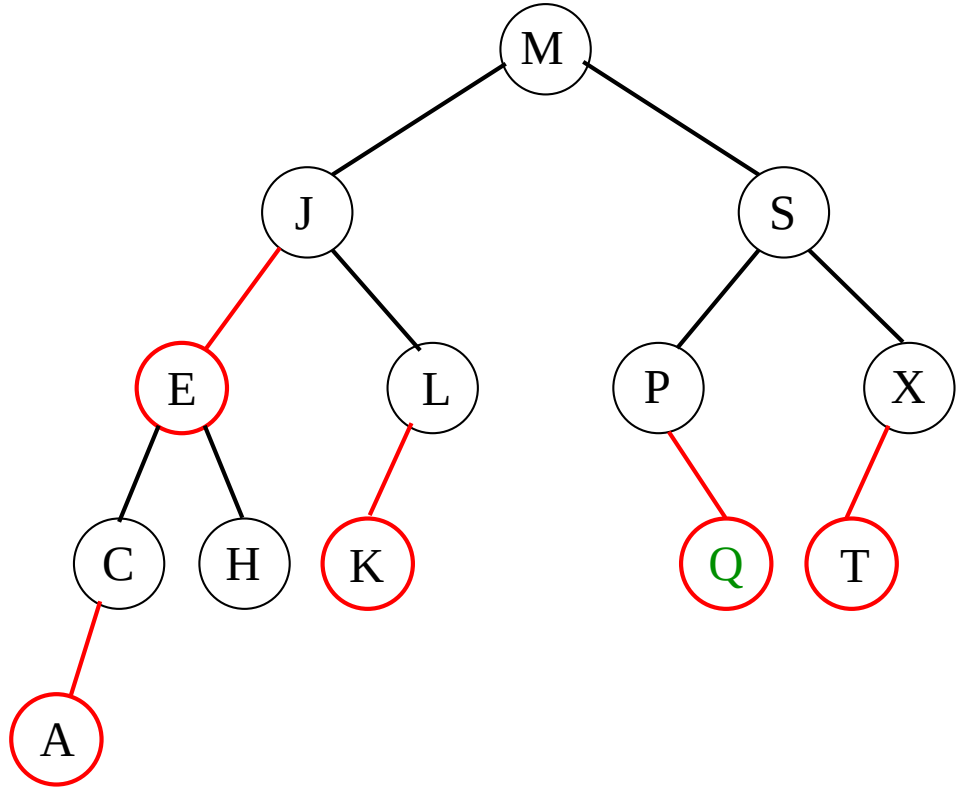
Inserção em ABB rubro-negra

Inserir o Q.



Inserção em ABB rubro-negra

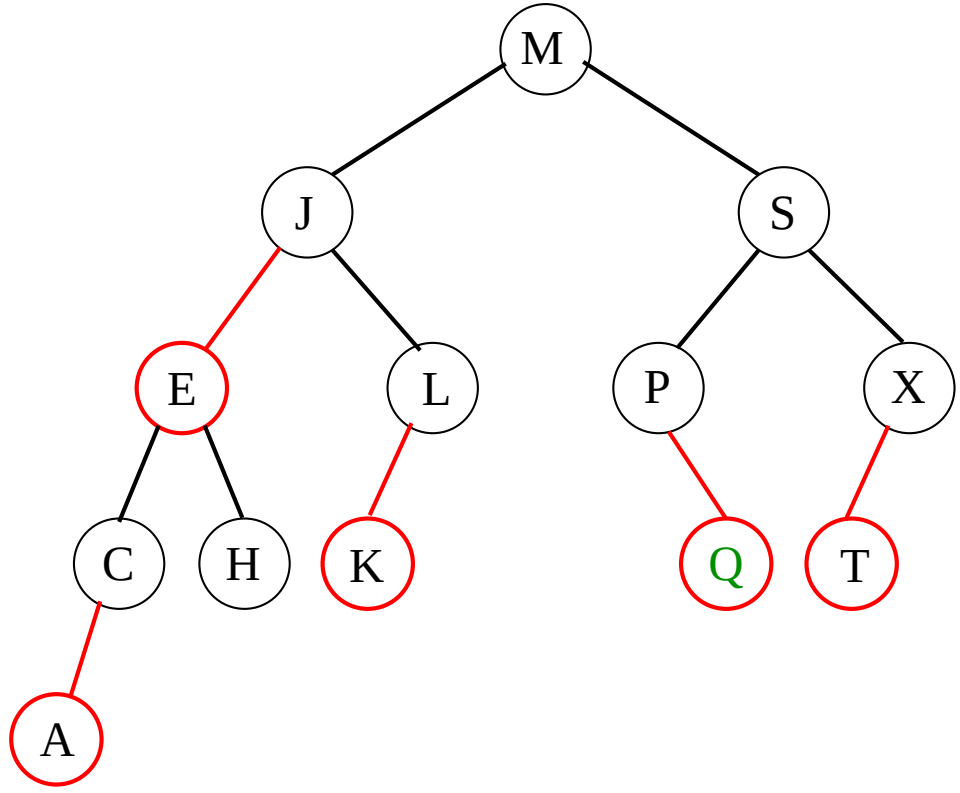
Inserir o Q.



Insere Q como um nó vermelho.

Inserção em ABB rubro-negra

Inserir o Q.

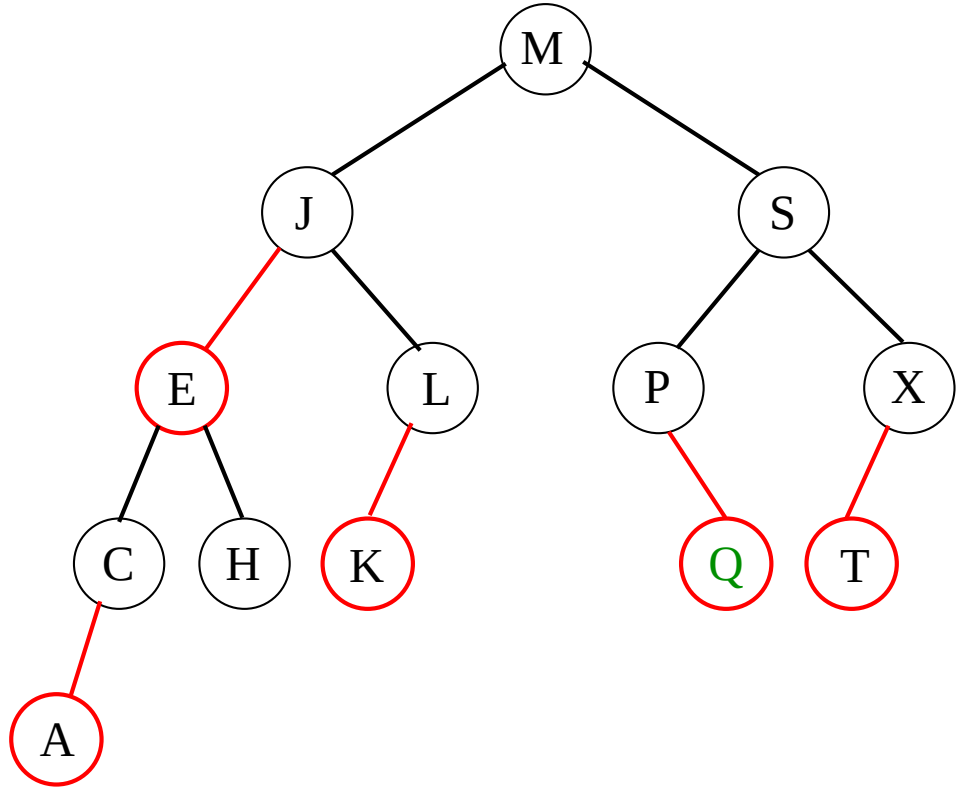


Insere Q como um nó vermelho.

Se o pai de Q não é vermelho,
mas Q é filho direito... temos que fazer algo...

Inserção em ABB rubro-negra

Inserir o Q.

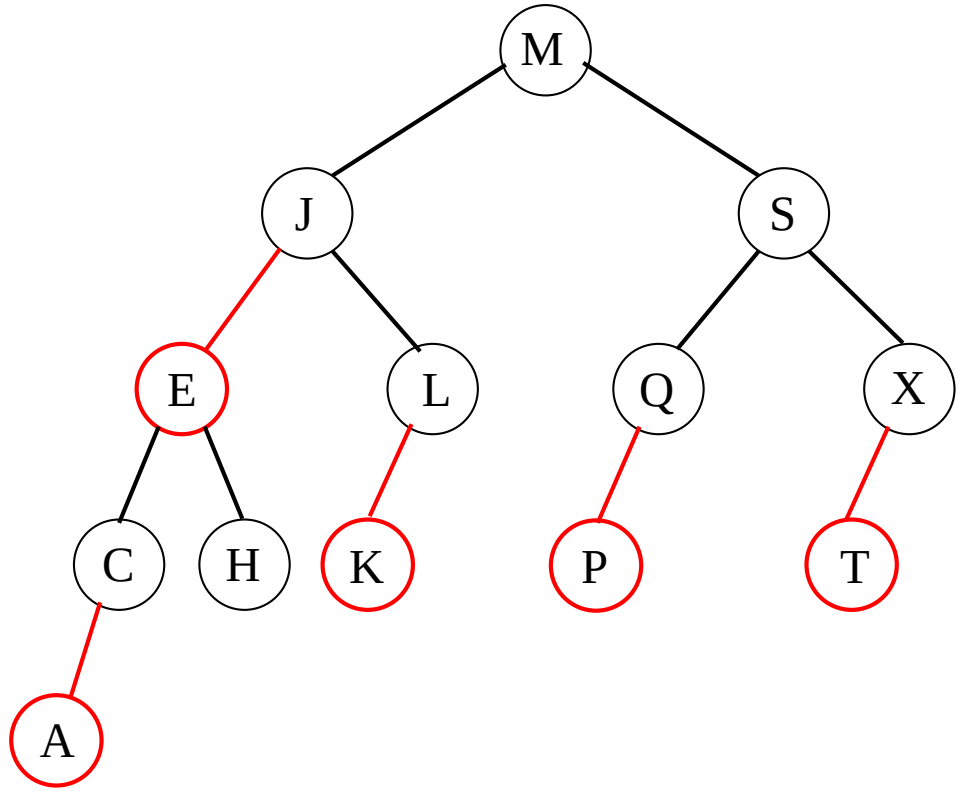


Insere Q como um nó vermelho.

Se o pai de Q não é vermelho,
mas Q é filho direito... temos que fazer algo...

Inserção em ABB rubro-negra

Inserir o Q.



Insere Q como um nó vermelho.

Se o pai de Q não é vermelho,
mas Q é filho direito, gire para a esquerda!

Rotações

O nó p é tal que $dir(p) \neq \text{NIL}$ e $cor(dir(p)) = \text{RUBRO}$.

GIREESQ (p)
1 $q \leftarrow dir(p)$
2 $dir(p) \leftarrow esq(q)$
3 $esq(q) \leftarrow p$
4 $cor(q) \leftarrow cor(p)$
5 $cor(p) \leftarrow$ RUBRO
6 **devolva** q

Rotações

O nó p é tal que $dir(p) \neq \text{NIL}$ e $cor(dir(p)) = \text{RUBRO}$.

```
GIREESQ (p)
1  q ← dir(p)
2  dir(p) ← esq(q)
3  esq(q) ← p
4  cor(q) ← cor(p)
5  cor(p) ← RUBRO
6  devolva q
```

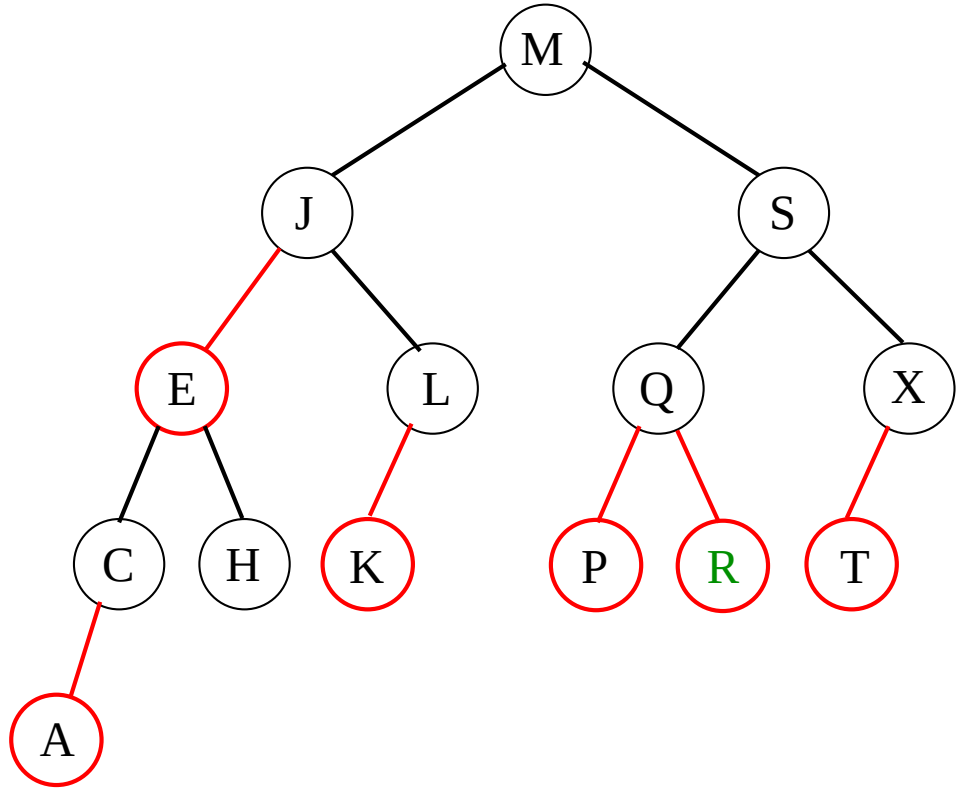
Exercício: Escreva o GIREDIR.

Inserção em ABB rubro-negra

Inserir o R.

Inserção em ABB rubro-negra

Inserir o R.

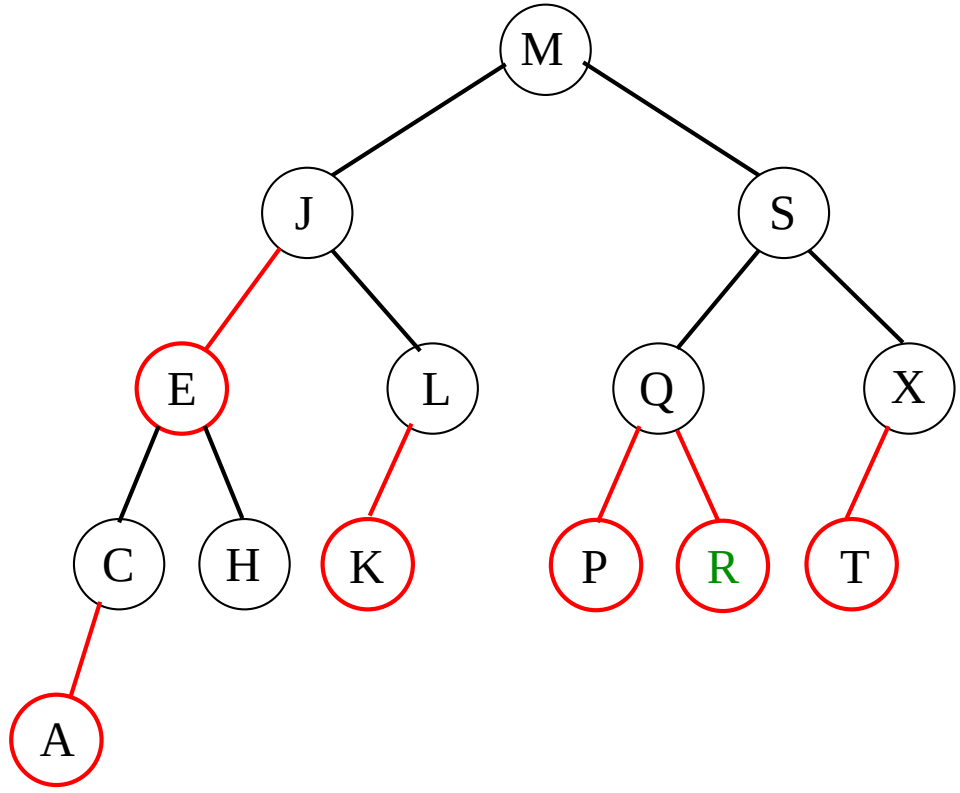


Insere R como um nó vermelho.

O pai de R é negro e R é filho direito.
Giro para a esquerda?

Inserção em ABB rubro-negra

Inserir o R.

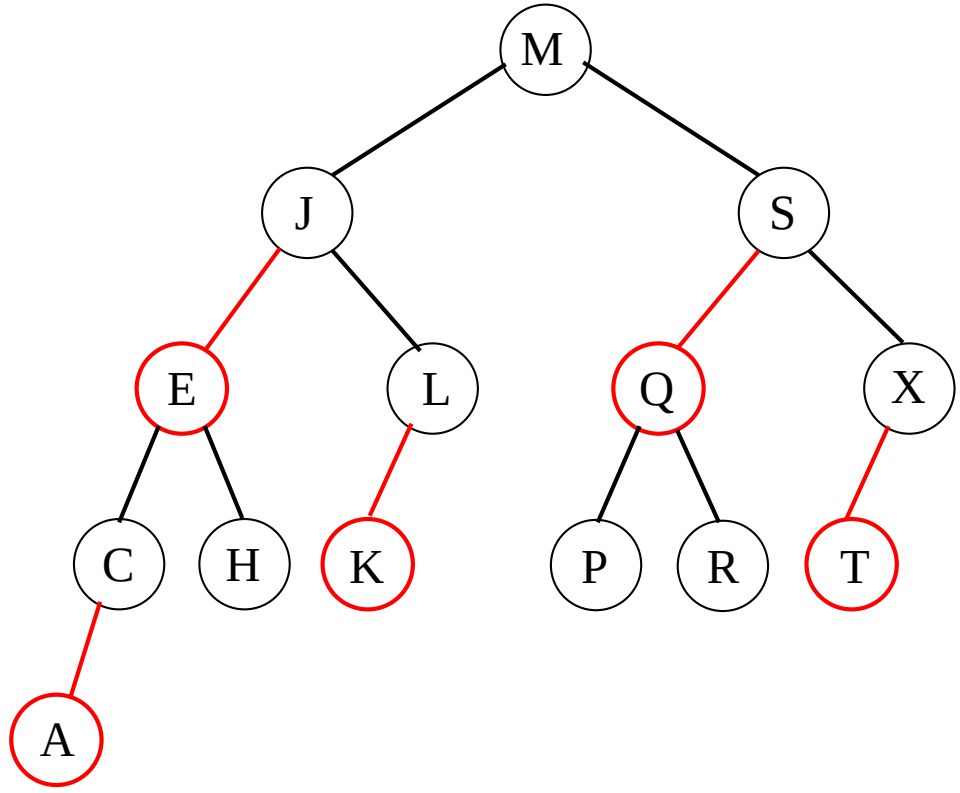


Insere R como um nó vermelho.

O pai de R é negro e R é filho direito.
Giro para a esquerda? Não...

Inserção em ABB rubro-negra

Inserir o R.

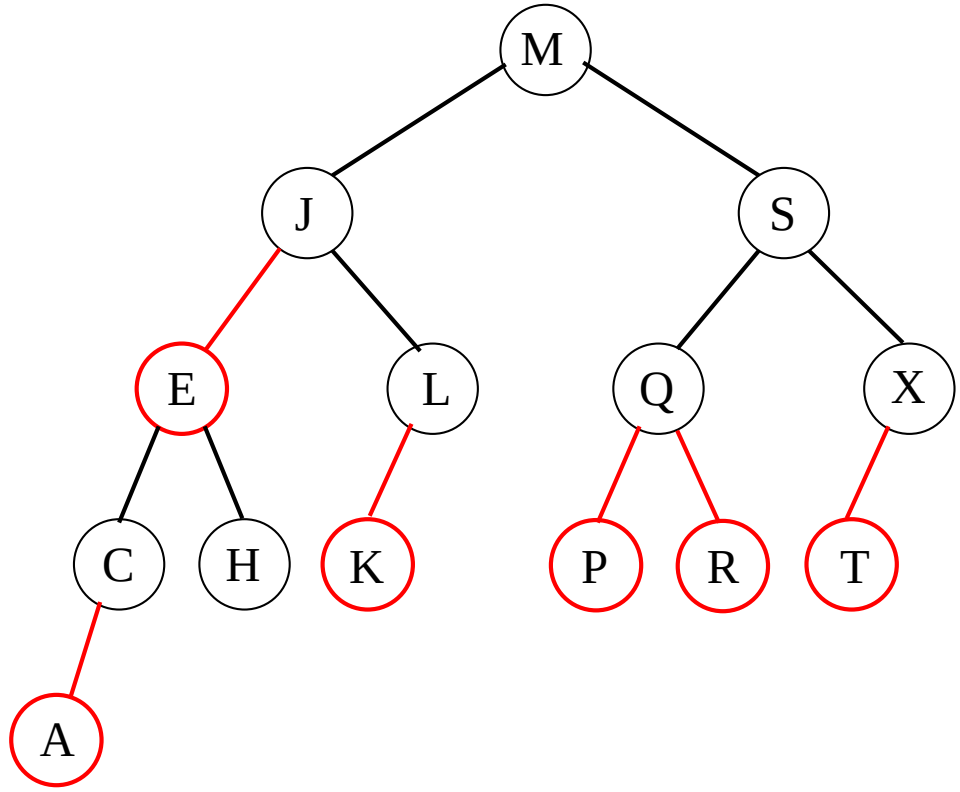


Insere R como um nó vermelho.

O pai de R é negro e R é filho direito.
Giro para a esquerda? Não... Troque cores!

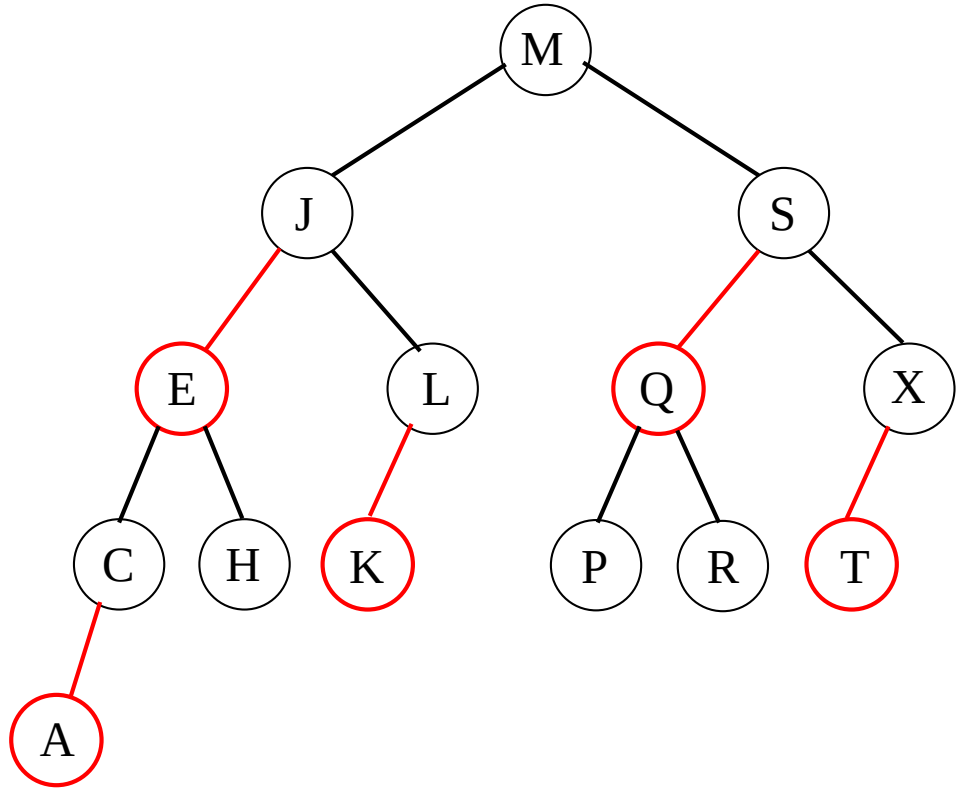
Inserção em ABB rubro-negra

Ao que corresponde a troca de cores em árvores 2-3?



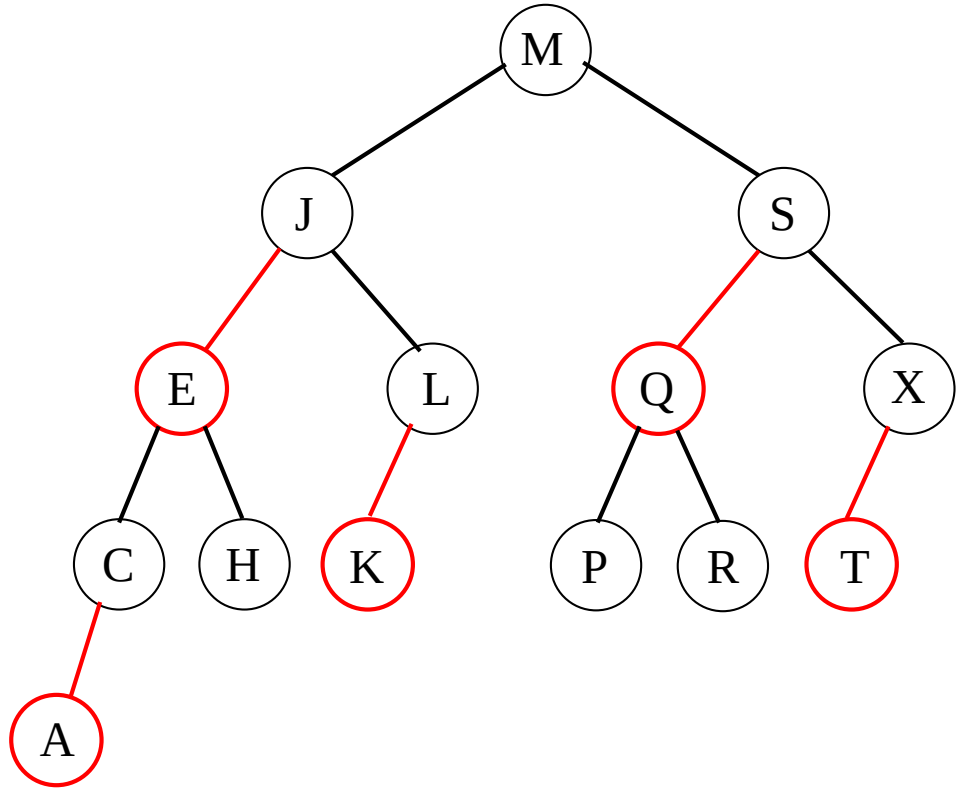
Inserção em ABB rubro-negra

Ao que corresponde a troca de cores em árvores 2-3?



Inserção em ABB rubro-negra

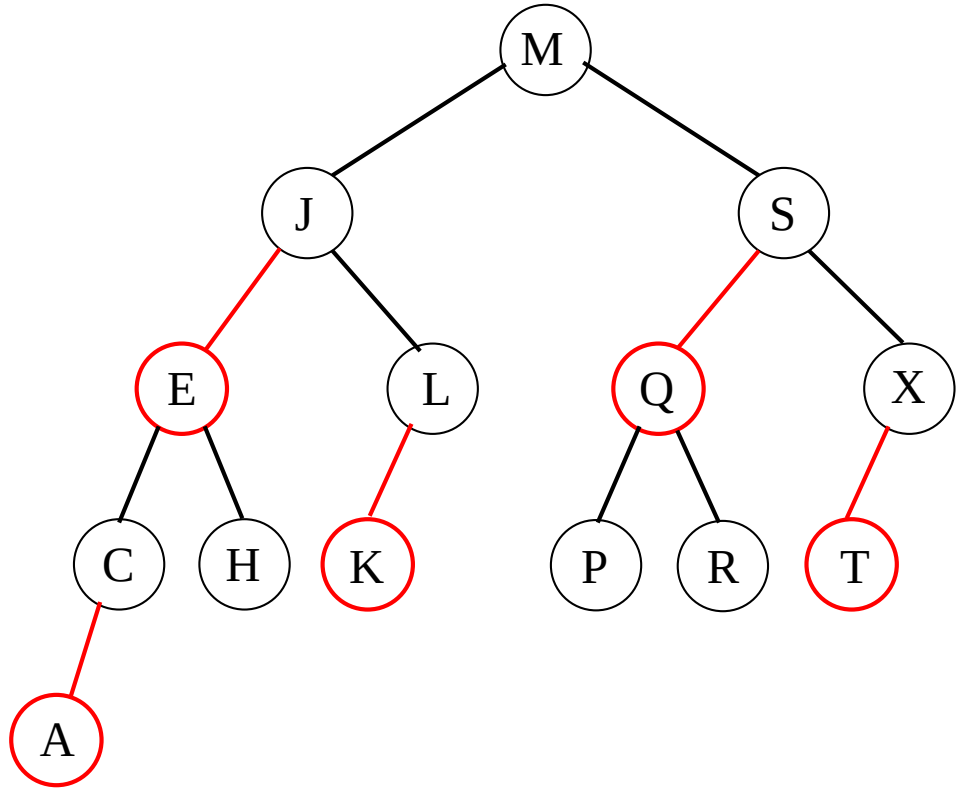
Ao que corresponde a troca de cores em árvores 2-3?



A quebrar um 4-nó mandando sua chave para cima!

Inserção em ABB rubro-negra

Ao que corresponde a troca de cores em árvores 2-3?



A quebrar um 4-nó mandando sua chave para cima!
Eventualmente é necessário repetir o processo...

Ajusta cores

O nó p é interno.

TROQUECORES (p)

1 $cor(p) \leftarrow$ OUTRACOR$(cor(p))$
2 $cor(esq(p)) \leftarrow$ OUTRACOR$(cor(esq(p)))$
3 $cor(dir(p)) \leftarrow$ OUTRACOR$(cor(dir(p)))$

Ajusta cores

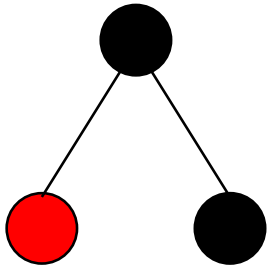
O nó p é interno.

TROQUECORES (p)
1 $cor(p) \leftarrow$ OUTRACOR$(cor(p))$
2 $cor(esq(p)) \leftarrow$ OUTRACOR$(cor(esq(p)))$
3 $cor(dir(p)) \leftarrow$ OUTRACOR$(cor(dir(p)))$

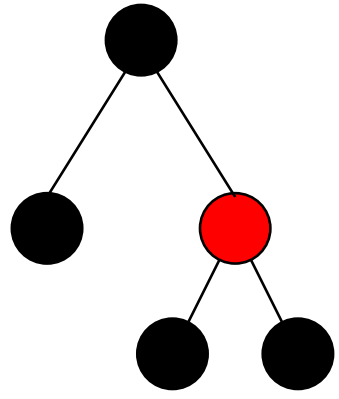
OUTRACOR (c)
1 **se** $c =$ RUBRO
2 **então devolva** NEGRO
3 **senão devolva** RUBRO

Inserção em ABB rubro-negra

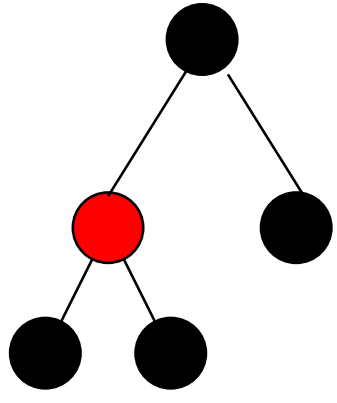
Resumos dos casos analisados até agora:

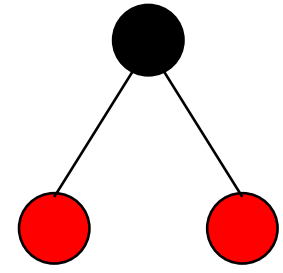


ok!

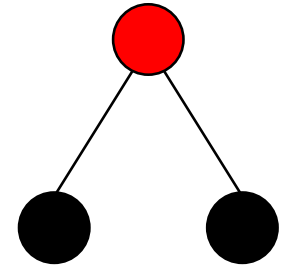


GIREESQ



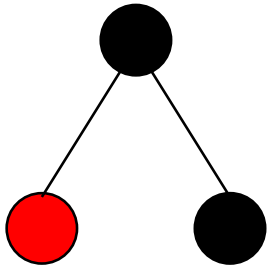


TROQUECORE

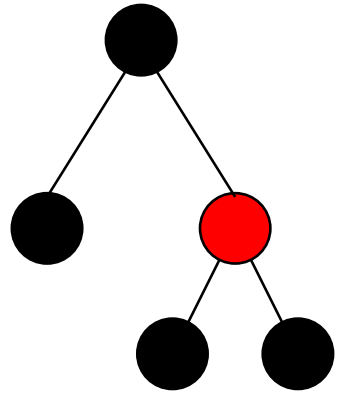


Inserção em ABB rubro-negra

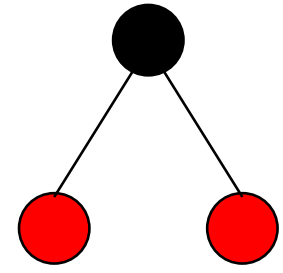
Resumos dos casos analisados até agora:



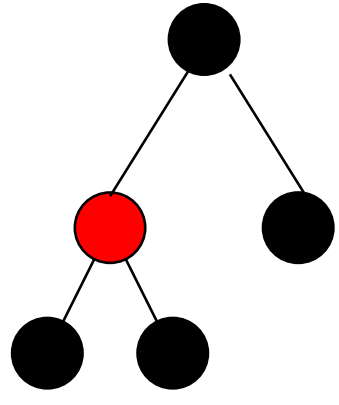
ok!

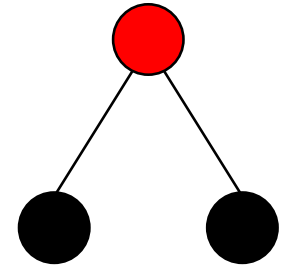


GIREESQ



TROQUECORE

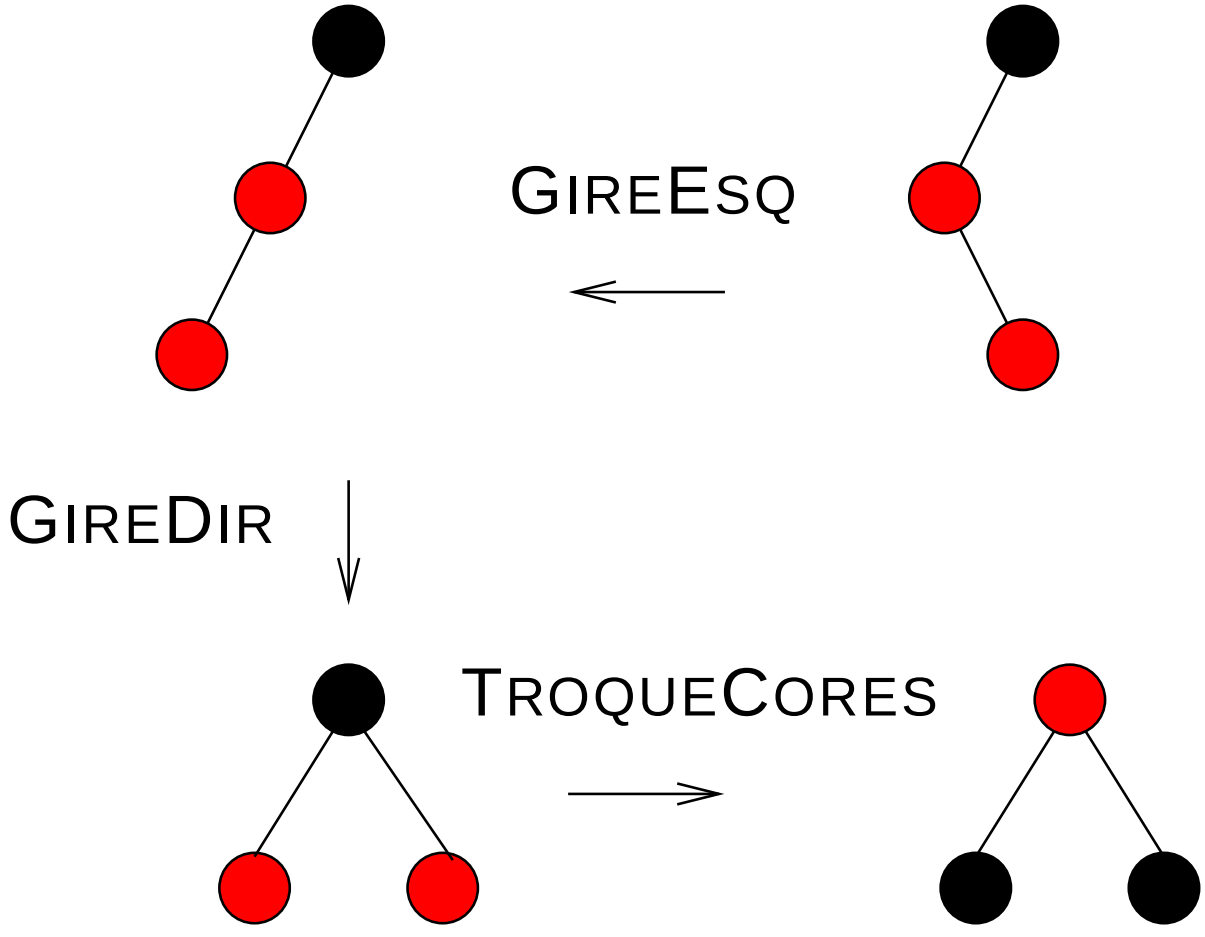




No terceiro caso,
repita o processo!

Inserção em ABB rubro-negra

Casos que faltam:



Inserção em ABB rubro-negra

```
RUBRO (p)
1  se p = NIL
2     então devolva FALSO
3     senão se cor(p) = RUBRO
4                 então devolva VERDADE
5                 senão devolva FALSO

NEGRO (p)
1  devolva não RUBRO(p)
```

Inserção em ABB rubro-negra

```
RUBRO (p)
1  se p = NIL
2     então devolva FALSO
3     senão se cor(p) = RUBRO
4                 então devolva VERDADE
5                 senão devolva FALSO

NEGRO (p)
1  devolva não RUBRO(p)

INSIRA (T, x)
 1  T ← INSIRAREC(T, x)
 2  cor(T) ← NEGRO     ▷ a raiz é sempre negra
```

Inserção em ABB rubro-negra

```
INSIRAREC (T, x)
 1  se T = NIL
 2     então q ← NOVACÉLULA(x, NIL, NIL, RUBRO)
 3           devolva q
 4  se x < info(T)
 5     então esq(T) ← INSIRAREC(esq(T), x)
 6     senão dir(T) ← INSIRAREC(dir(T), x)
 7  se RUBRO(dir(T)) e NEGRO(esq(T))
 8     então T ← GIREESQ(T)
 9  se RUBRO(esq(T)) e RUBRO(esq(esq(T)))
10     então T ← GIREDIR(T)
11  se RUBRO(esq(T)) e RUBRO(dir(T))
12     então TROQUECORES(T)
13  devolva T
```

Comentários finais

Exercício: Simule a inserção em uma árvore rubro-negra inicialmente vazia das seguintes chaves:

20, 17, 38, 40, 53, 10, 6, 16, 23, 14, 11, 50, 45.

Quarta edição do livro do Sedgewick (com Wayne):
`http://www.cs.princeton.edu/algs4/`

Próxima aula: remoção de árvores rubro-negras e skip lists.