

# Unidade VIII:

## Árvores 2.3.4

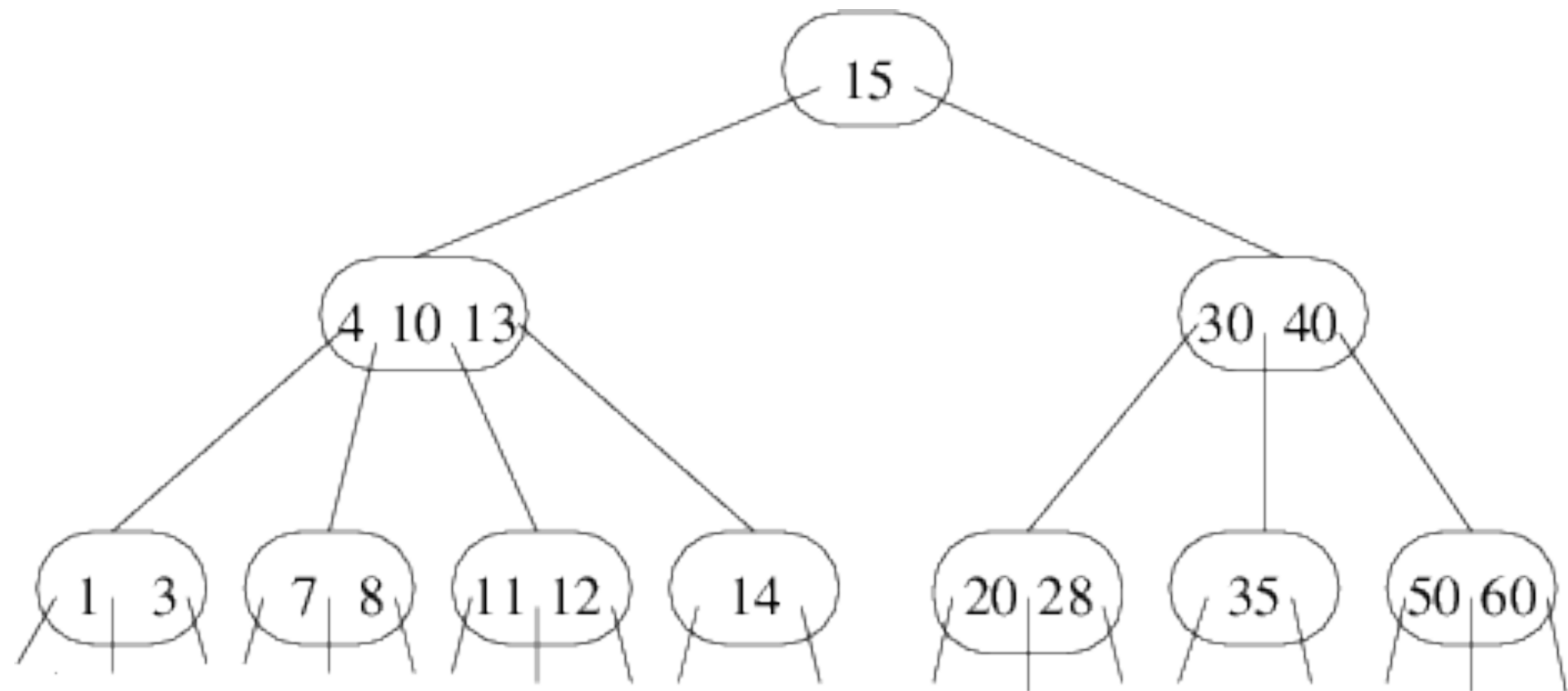


**PUC Minas**

Instituto de Ciências Exatas e Informática  
Departamento de Ciência da Computação

# Introdução

- Árvore de pesquisa cujos nós são de três tipos (2-nó, 3-nó ou 4-nó) e as folhas estão situadas no mesmo nível



# Propriedade das Árvore 2.3.4

- A altura  $h(n)$  de uma árvore-2.3.4 <sup>1</sup> contendo  $n$  itens é  $\Theta(\lg n)$
- Por questões de simplificação, este material considera que todos os elementos da árvore são distintos

<sup>1</sup>Prova: Temos que  $\log_4(n+1) \leq h(n) \leq \log_2(n+1)$ . Além disso, a árvore-2.3.4 que tem menos itens é a que tem só tem 2-nós. Como todas as folhas estão no mesmo nível, o nº de itens da árvore é  $2^0 + 2^1 + \dots + 2^h = 2^{h+1} - 1$ . Da mesma forma, a árvore-2.3.4 que tem mais itens é a que só tem 4-nós e seu nº de itens é  $4^0 + 4^1 + \dots + 4^h = 4^{h+1} - 1$ . Deduz-se que, para toda árvore-2.3.4 contendo  $n$  itens e de altura  $h(n)$ , temos a relação  $2^{h+1} - 1 \leq n \leq 4^{h+1} - 1$ , de onde se tira a relação acima.

# Pesquisa em Árvores 2.3.4

- Igual ao das árvores binárias
- Funcionamento básico:
  - (1) Verificar se o elemento procurado  $x$  está no nó raiz
  - (2) Se estiver, tem-se uma **resposta positiva**
  - (3) Senão, se  $x < x_1$ , verificar na subárvore da esquerda
  - (4) Senão, se  $x < x_2$ , na do meio à esquerda
  - (5) Senão, se  $x < x_3$ , na do meio à direita
  - (6) Senão, na da direita
  - (7) Se a subárvore for nula, tem-se uma **resposta negativa**

# Análise de Complexidade da Pesquisa

- Número de comparações em uma pesquisa com sucesso:
  - Melhor Caso:  $\Theta(1)$
  - Pior e Caso Médio:  $\Theta(\lg(n))$

# Inserção

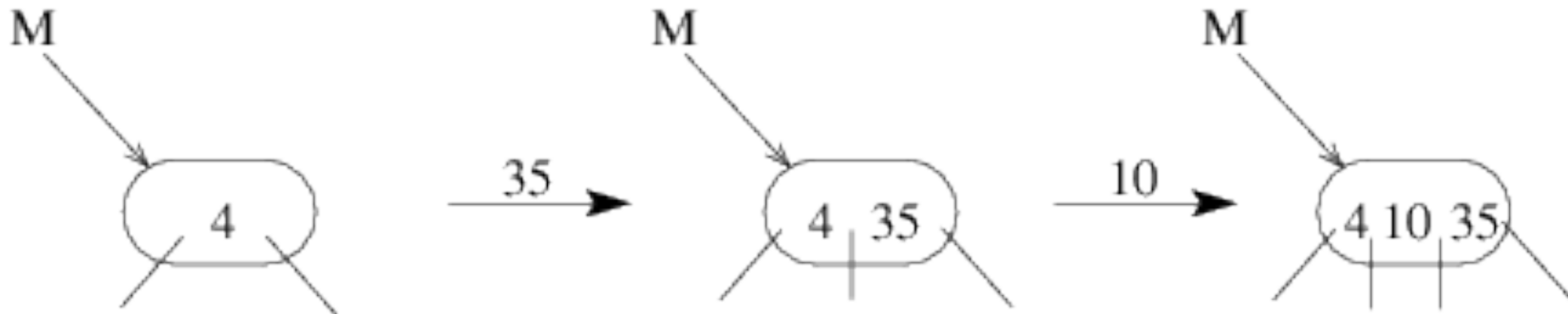
- Este material considera a inserção sempre nas folhas
- Se a folha tiver 1 ou 2 elementos, insere-se nela
- Senão, insere-se um novo nó na árvore e a reorganiza
- O processo de reorganização pode ser feito por ascensão ou na descida

# Inserção com Fragmentação na Ascensão

- O exemplo a seguir mostra a construção de uma árvore-2.3.4 por inserções sucessivas nas folhas a partir de uma árvore vazia
- Insere-se sucessivamente os elementos 4, 35, 10, 13, 3, 30, 15, 12, 7, 40, 20, 11 e 6

# Inserção com Fragmentação na Ascensão

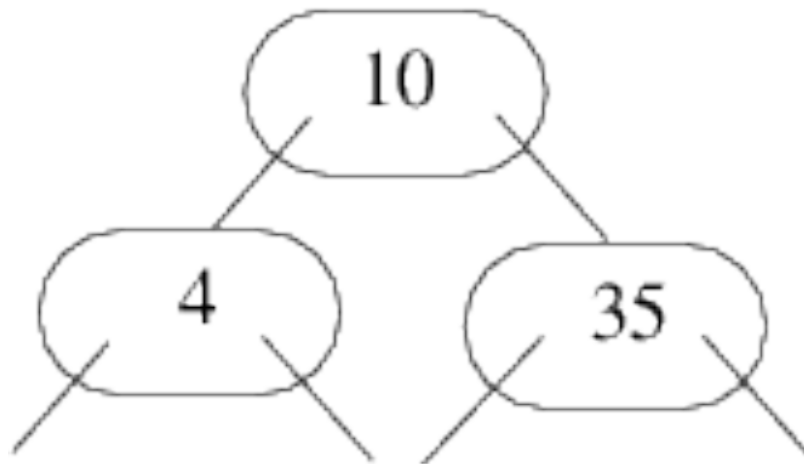
- A inserção do 4 leva à criação de um 2-nó, que se transforma em 3-nó com a inserção do 35 e, depois, em 4-nó com a do 10





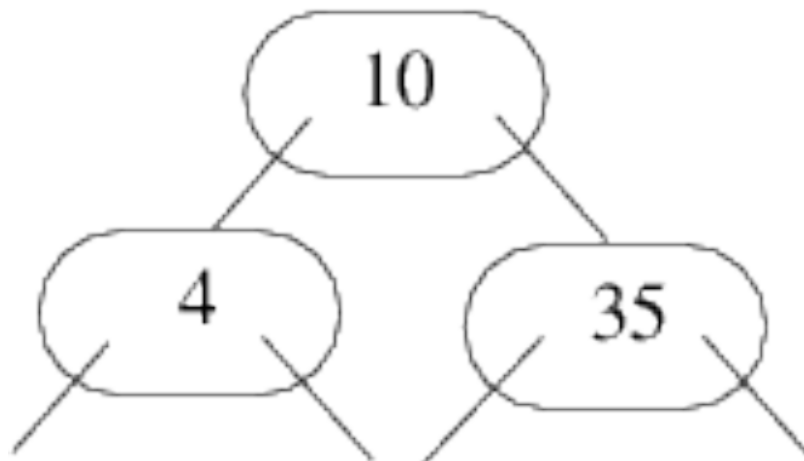
# Inserção com Fragmentação na Ascensão

- Neste ponto, o nó da raiz não comporta mais elementos e, do ponto de vista da pesquisa, a árvore equivalente à binária abaixo:



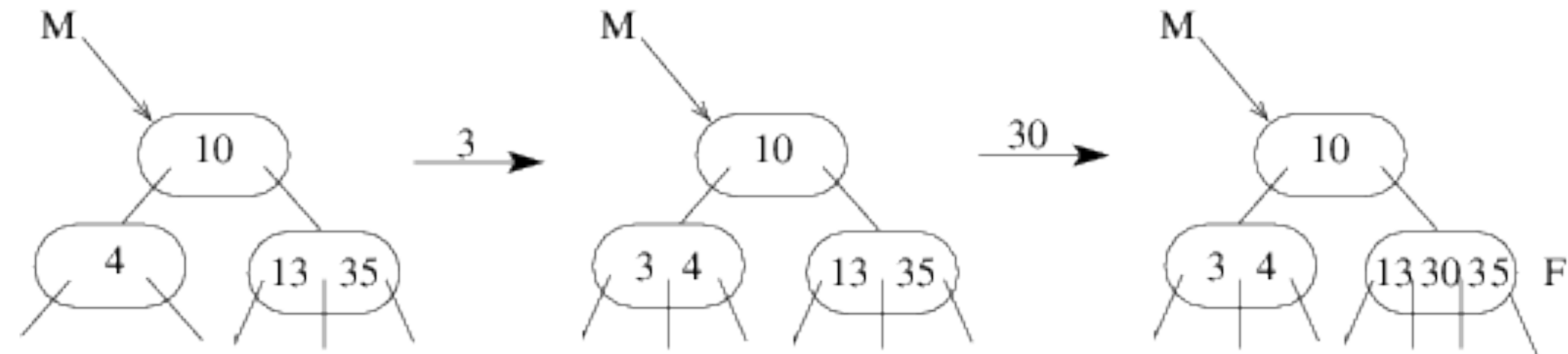
# Inserção com Fragmentação na Ascensão

- Assim, para inserir o quarto elemento, fragmenta-se o nó corrente, ascendendo seu elemento médio e transformando a árvore em uma binária – que também é uma 2.3.4
- Na árvore corrente, existe lugar nas folhas para acomodar novos elementos, permitindo as inserções do 13, 3 e 30



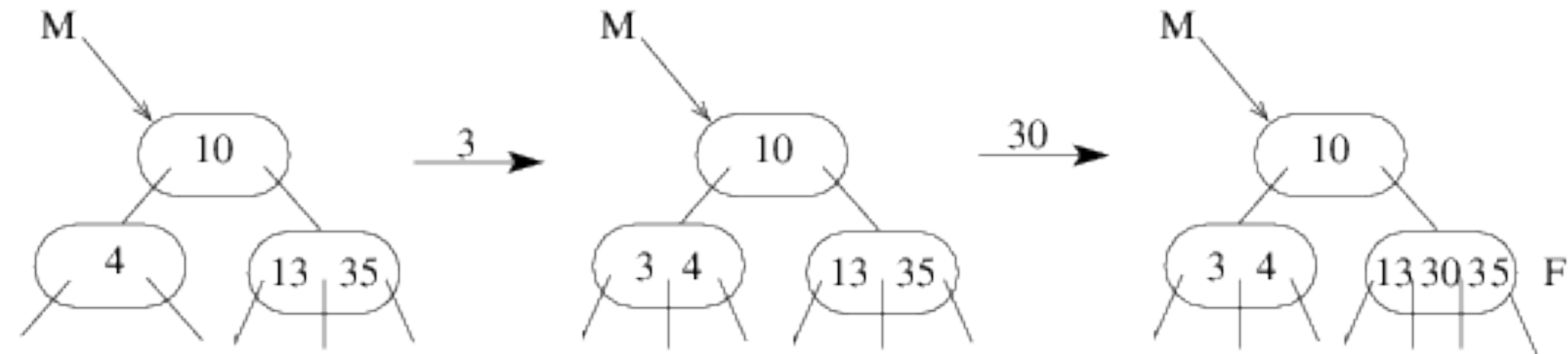
# Inserção com Fragmentação na Ascensão

- Árvore-2.3.4 após as inserções do 13, 3 e 30



# Inserção com Fragmentação na Ascensão

- Como inserir o 15?



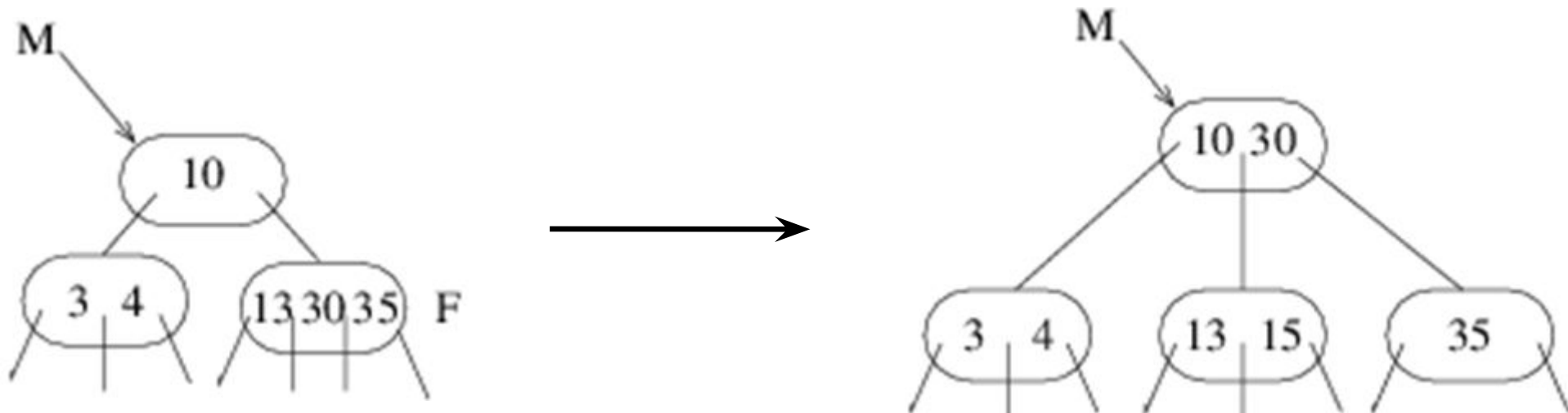
# Inserção com Fragmentação na Ascensão

- Para inserir o 15, fragmenta-se o nó da seguinte forma:
  - Faz-se a ascensão do elemento médio de F (30) para o pai
  - Fragmenta-se F em dois 2-nós, um com o menor elemento e outro com o maior
  - Como o 15 é menor que o 30, insere-se o 15 no nó que contém o 13



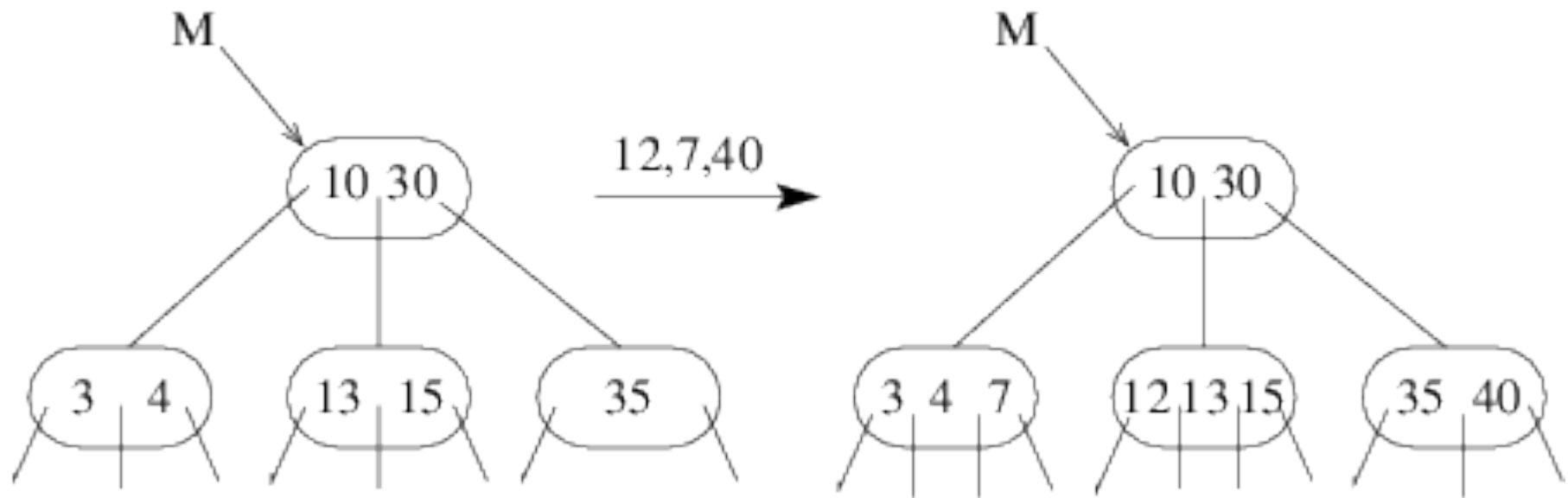
# Inserção com Fragmentação na Ascensão

- Para inserir o 15, fragmenta-se o nó da seguinte forma:
  - Faz-se a ascensão do elemento médio de F (30) para o pai
  - Fragmenta-se F em dois 2-nós, um com o menor elemento e outro com o maior
  - Como o 15 é menor que o 30, insere-se o 15 no nó que contém o 13



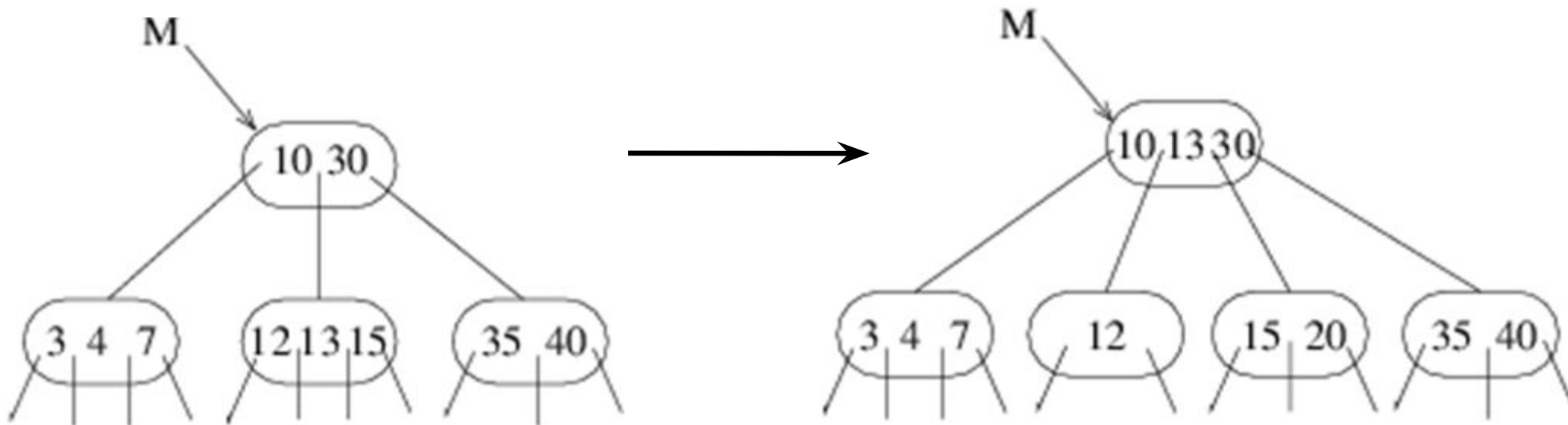
# Inserção com Fragmentação na Ascensão

- A inserção do 12, 7, 40 é fácil



# Inserção com Fragmentação na Ascensão

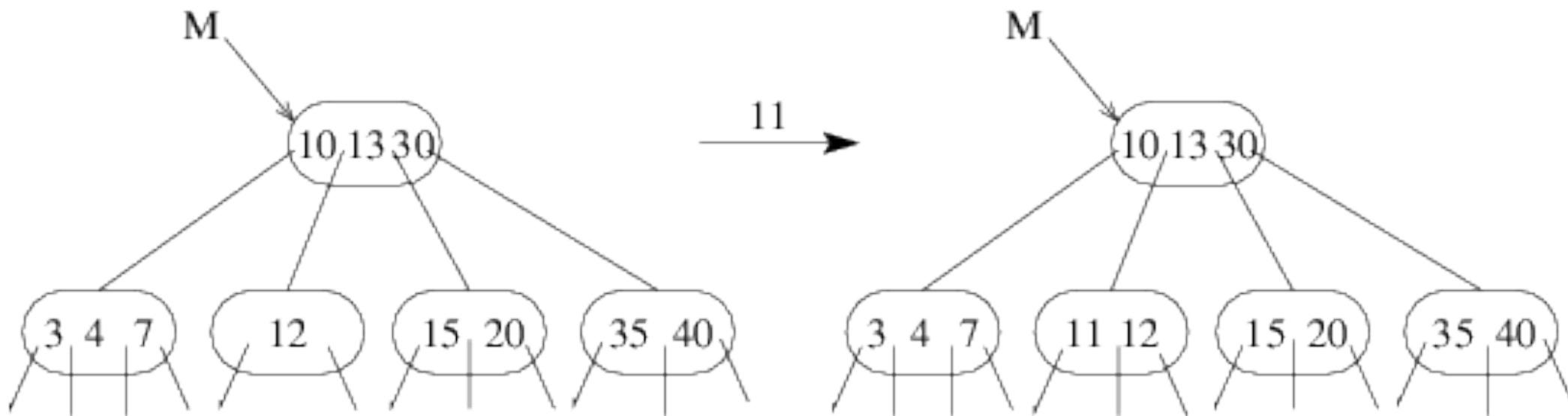
- A inserção do 20 leva a uma fragmentação em dois da folha que contém o 12, 13 e 15, com ascensão do 13 para o pai





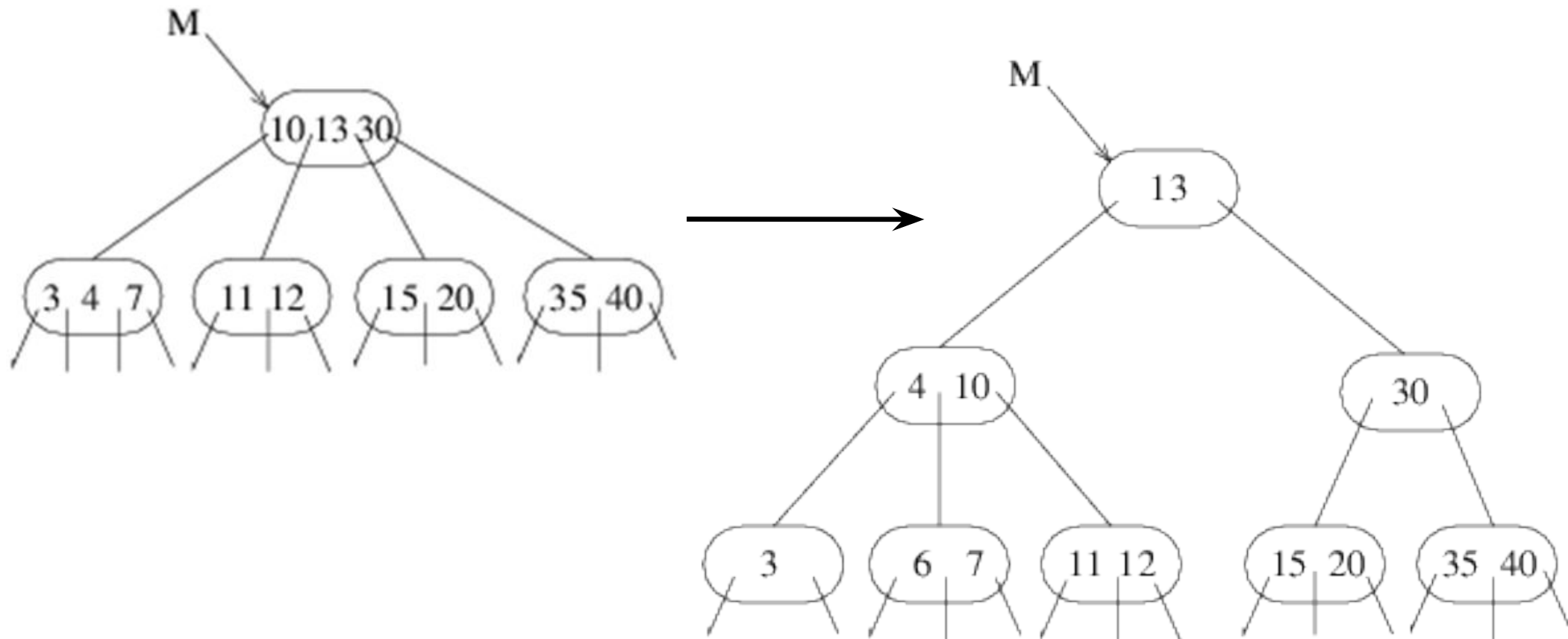
# Inserção com Fragmentação na Ascensão

- A inserção de 11 se faz sem problemas



# Inserção com Fragmentação na Ascensão

- A inserção de 6 causa a fragmentação da folha (3,4,7) cujo pai contém três elementos. Por sua vez, a ascensão do 4 fragmenta a raiz em dois nós e cria um novo nó raiz para receber o elemento médio 13



# Inserção com Fragmentação na Ascensão

- Um efeito colateral da fragmentação ascendente é que ela pode provocar uma cascata de fragmentações
- Quando o caminho da inserção na árvore for formado somente por 4-nós, teremos fragmentações em toda a altura da árvore

# Inserção com Fragmentação na Descida

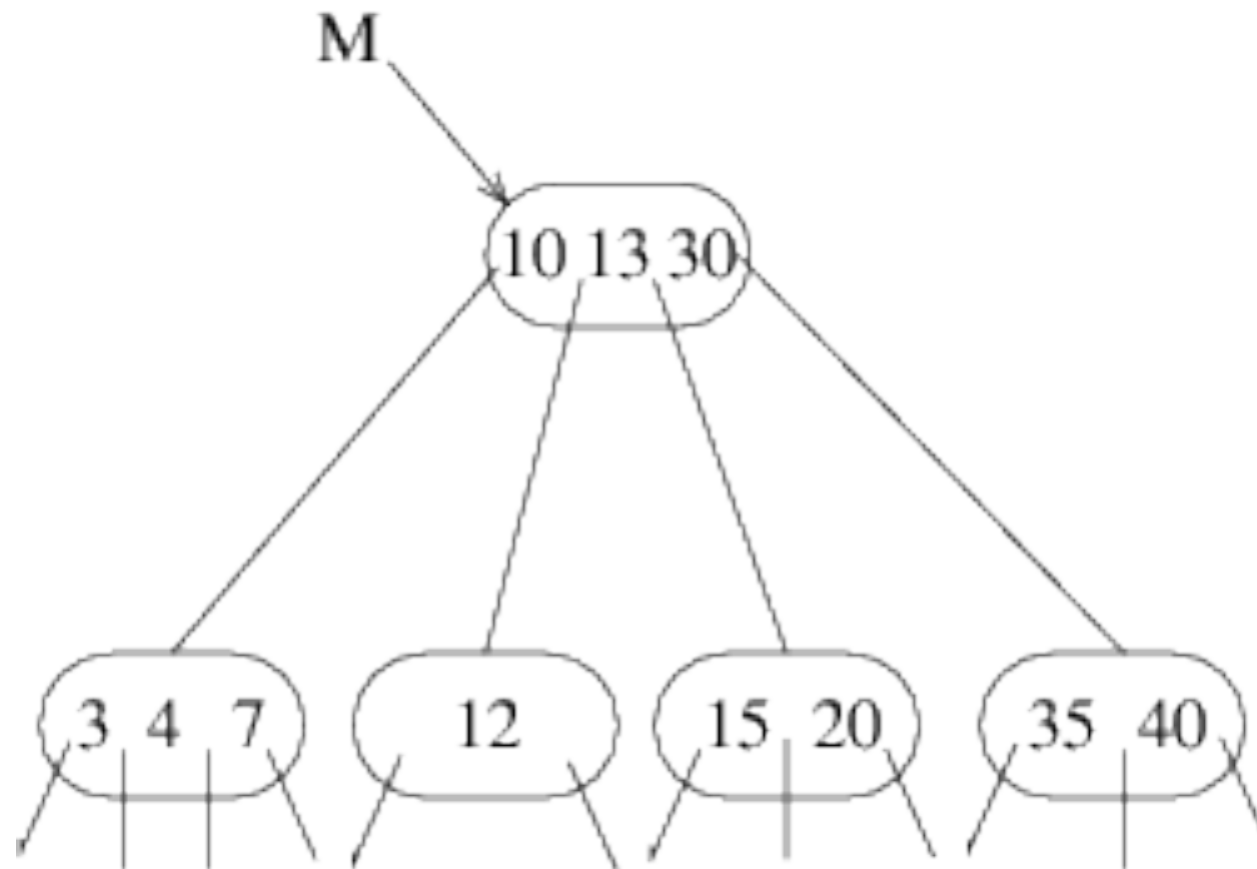
- Para evitar a propagação das fragmentações, basta proibir que a árvore tenha dois 4-nós seguidos
- Toda inserção causará no máximo uma fragmentação
- Pode-se implementar essa restrição fragmentando os 4-nós na descida (durante a pesquisa do lugar para adicionar o novo elemento)

# Inserção com Fragmentação na Descida

- A fragmentação na descida é preventiva dado que qualquer 4-nó será fragmentado antes de qualquer inserção
- Uma desvantagem são as fragmentações “inúteis”

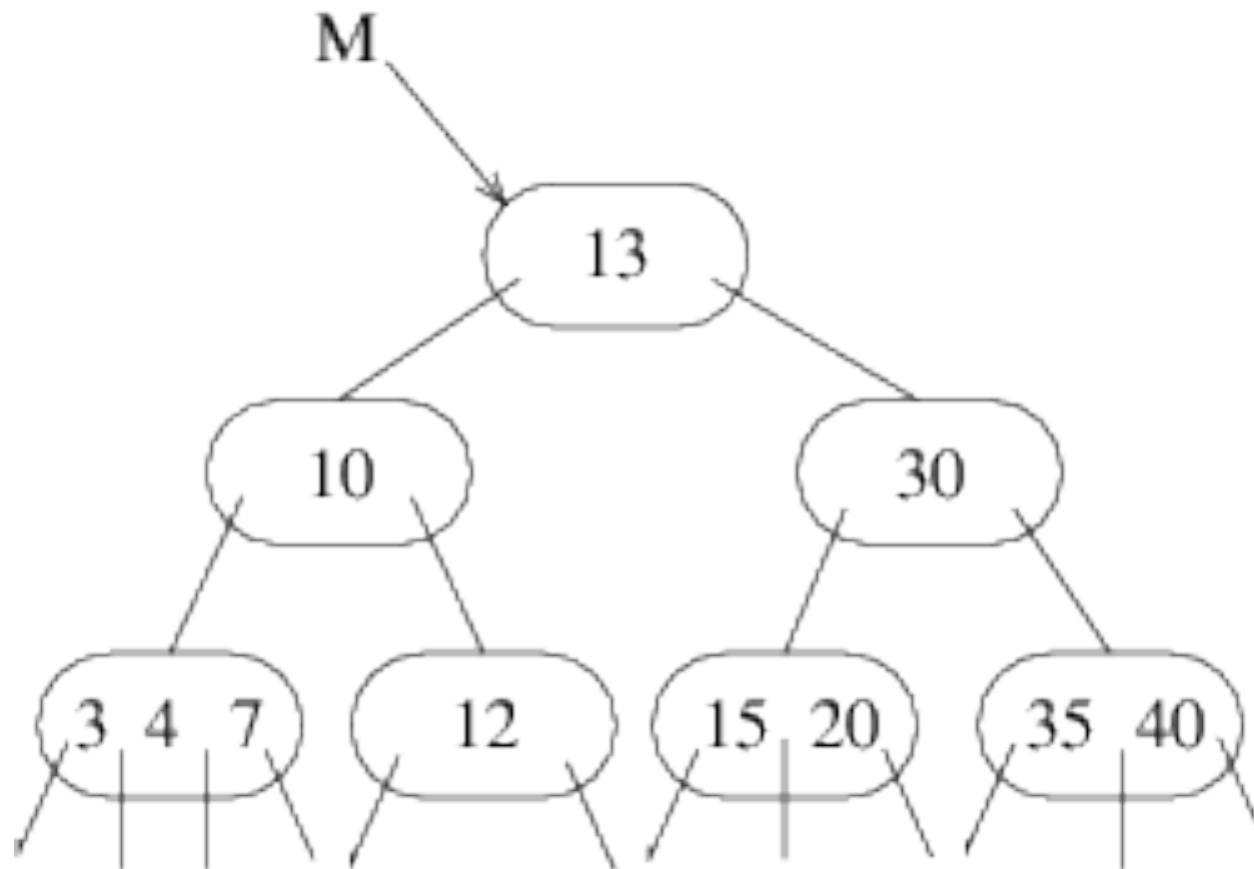
# Inserção com Fragmentação na Descida

- Exemplo: Seja a árvore-2.3.4 obtida com as inserções sucessivas de 4, 35, 10, 13, 3, 30, 15, 12, 7, 40 e 20:



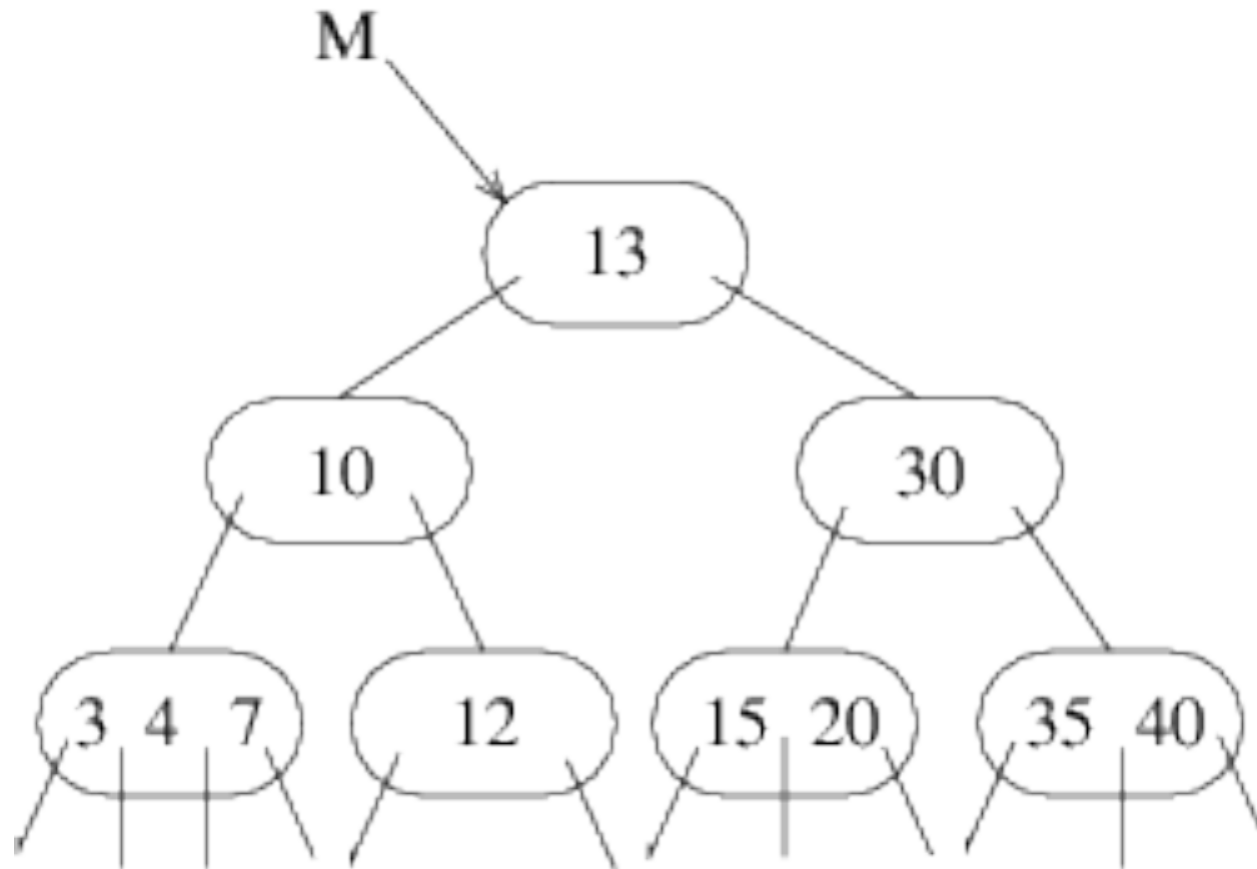
# Inserção com Fragmentação na Descida

- Neste ponto, como a inserção de qualquer elemento fragmenta o nó da raiz, de forma pró-ativa, ele é fragmentado



# Inserção com Fragmentação na Descida

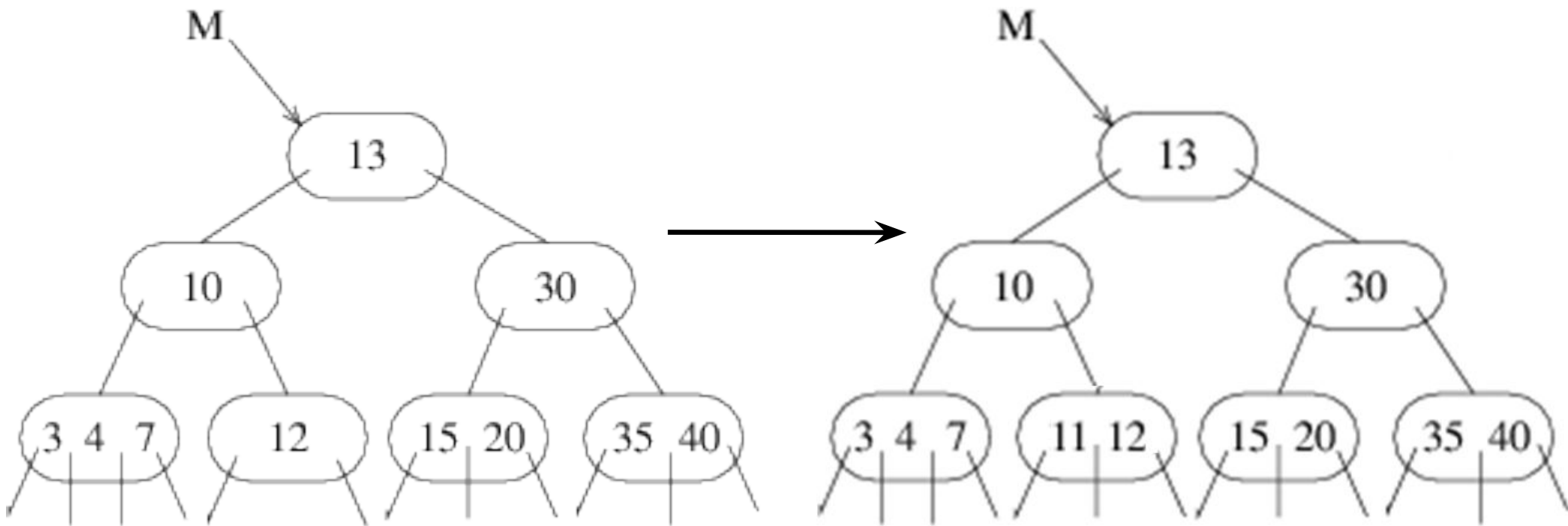
- Neste ponto, como a inserção de qualquer elemento fragmenta o nó da raiz, de forma pró-ativa, ele é fragmentado





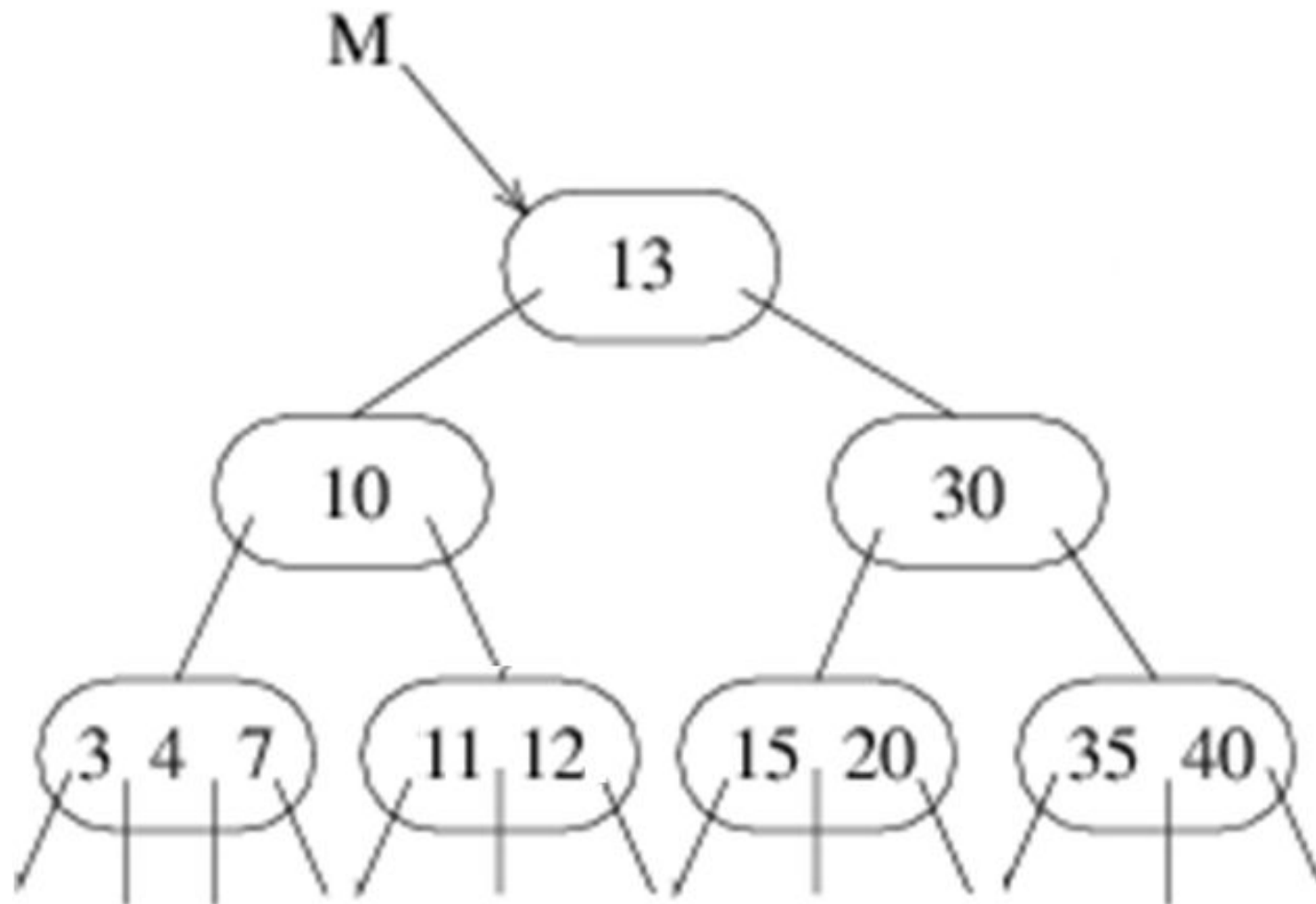
# Inserção com Fragmentação na Descida

- Agora, a inserção, por exemplo, do 11 pode ser feita no nó contendo o 12



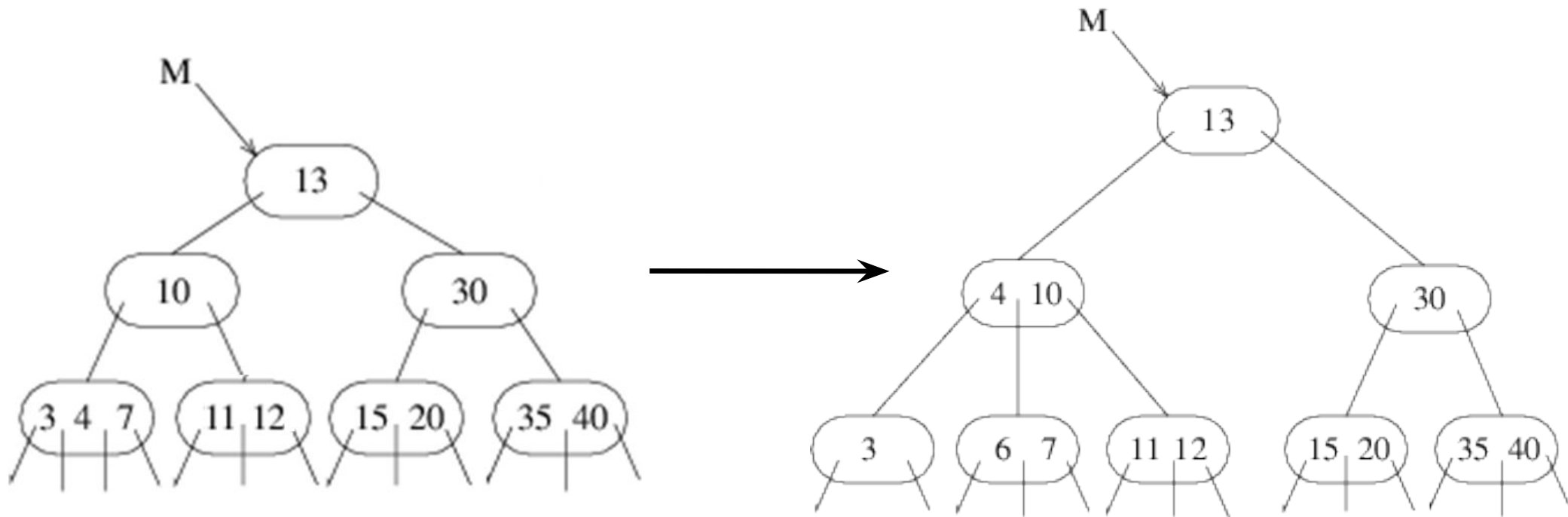
# Inserção com Fragmentação na Descida

- Neste ponto, observa-se que a altura da árvore corrente é maior que a obtida pelo método anterior



# Inserção com Fragmentação na Descida

- Neste ponto, observa-se que a altura da árvore corrente é maior que a obtida pelo método anterior, contudo, inserindo o 6, fragmenta-se o nó (3,4,7) e obtém-se a árvore final do método anterior

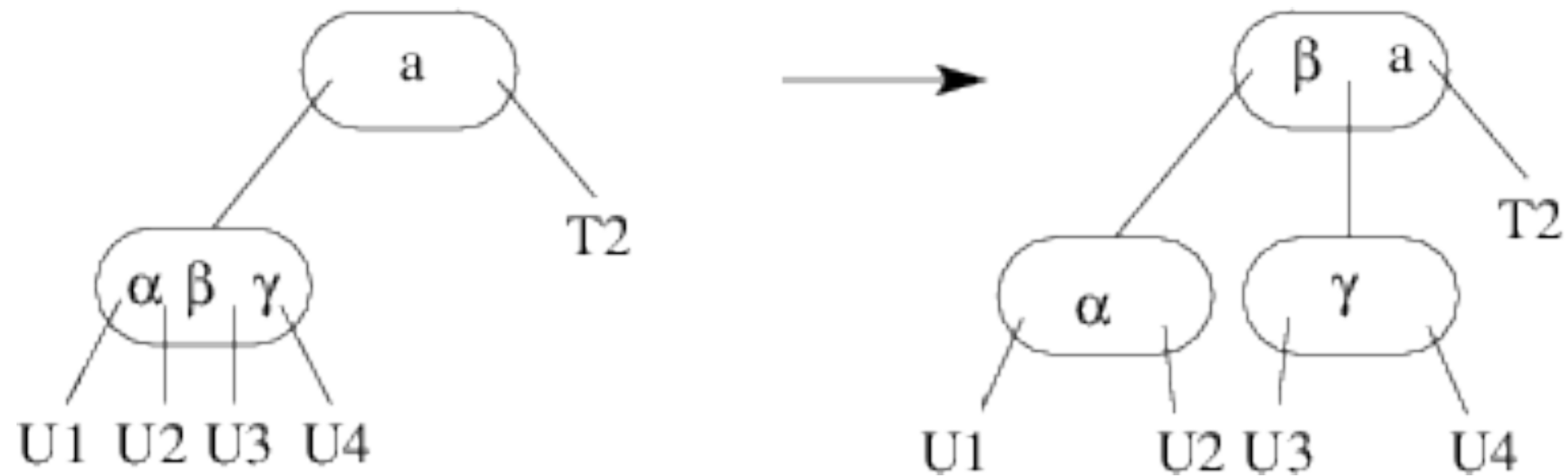


# Inserção com Fragmentação na Descida

- Observa-se que:
  - Esta técnica faz todos os reequilíbrios durante a descida na árvore e a inserção quando se chega em uma folha
  - Os reequilíbrios na descida são puramente locais

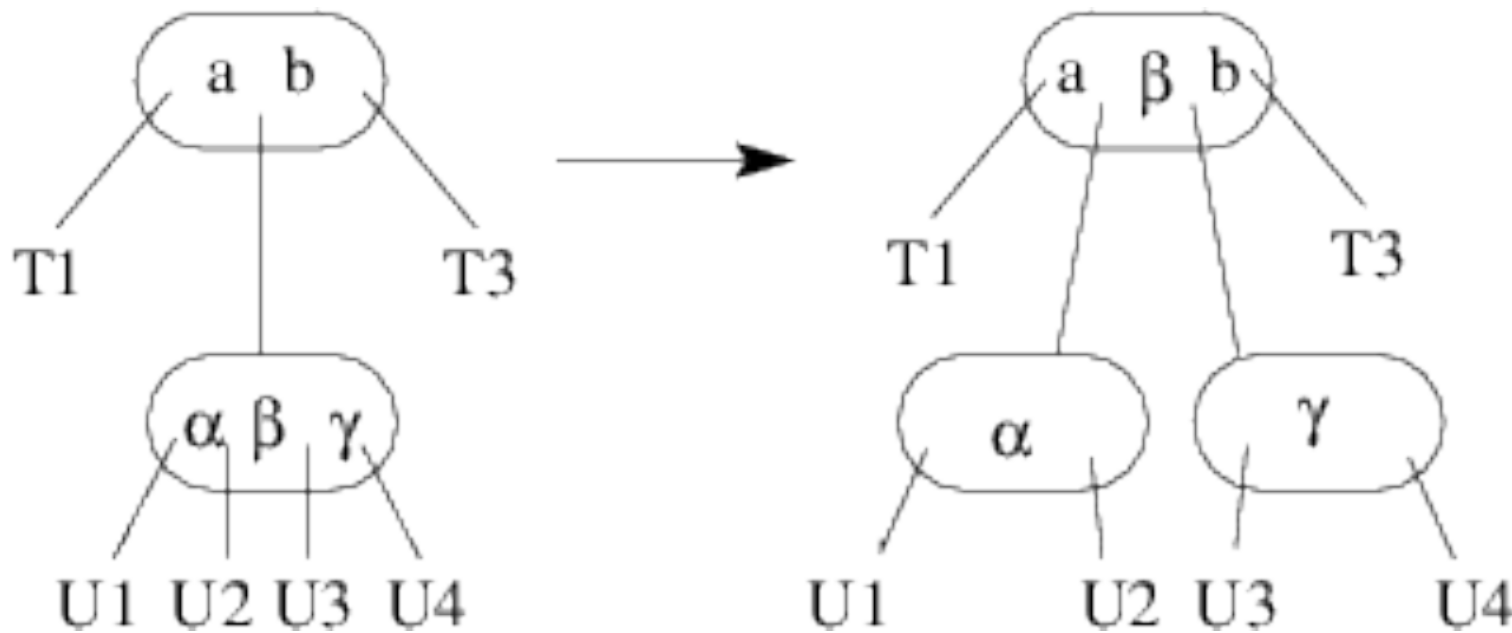
# Inserção com Fragmentação na Descida

- Observa-se que:
  - Quando se fragmenta um nó e seu pai é um 2-nó, esse vira um 3-nó como no exemplo abaixo:



# Inserção com Fragmentação na Descida

- Observa-se que:
  - Quando se fragmenta um nó e seu pai é um 3-nó, esse vira um 4-nó como no exemplo abaixo:



# Inserção com Fragmentação na Descida

- Observa-se que:
  - O pai de um nó nunca será um 4-nó, pois, nesse caso, ele seria fragmentado anteriormente
  - Esse processo não modifica a profundidade das folhas, exceto quando se fragmenta a raiz em que a altura da árvore aumenta em 1 unidade
  - A inserção conserva as propriedades das árvores-2.3.4

# Inserção com Fragmentação na Descida

• Observa-se que:

- Os dois métodos de inserção sempre operam em um caminho da raiz a uma folha da árvore-2.3.4
- A complexidade para o número de comparações dos dois métodos no pior e no caso médio é  $\Theta(\lg n)$



# Inserção com Fragmentação na Descida

- Vantagens do segundo método de inserção sobre o primeiro:
  - O primeiro precisa de uma pilha para restaurar o equilíbrio da árvore repassando o caminho inverso de pesquisa no caso de fragmentações
  - A árvore do segundo pode ser acessada paralelamente por vários usuários com uma sincronização mínima
- A desvantagem do segundo método é que ele normalmente consome mais espaço de memória porque sua taxa de ocupação dos nós é menor do que no primeiro. Isso também pode aumentar a altura da árvore do segundo

# Técnica Pró-ativa vs. Reativa

- A inserção com fragmentação por ascensão é reativa, a na descida, pró-ativa
- Técnicas reativas esperam um problema para atuar
- Técnicas pró-ativas se adiantam e tentam resolver/minimizar o problema antes que o mesmo aconteça
- Várias soluções na Computação são classificadas como pró-ativas e reativas

## Exercício

- Insira os números 1 a 20 em uma 2.3.4 usando fragmentação por ascensão
- Insira os números 1 a 20 em uma 2.3.4 usando fragmentação na descida
- Insira os números 20 a 1 em uma 2.3.4 usando fragmentação por ascensão
- Insira os números 20 a 1 em uma 2.3.4 usando fragmentação na descida