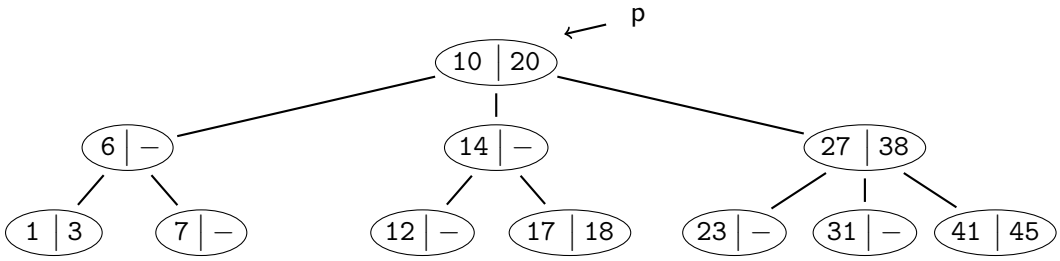
Estruturas de Dados

lista de exercícios 17

1. Imprimindo a árvore 2-3

   Imagine que você tem uma árvore 2-3 apontada pelo ponteiro `p`

   Por exemplo,



   A tarefa consiste em

   → *Imprimir os elementos da árvore em ordem crescente*

   a) Apresente um algoritmo recursivo que resolve esse problema.

   b) Apresente um algoritmo não recursivo que resolve esse problema.

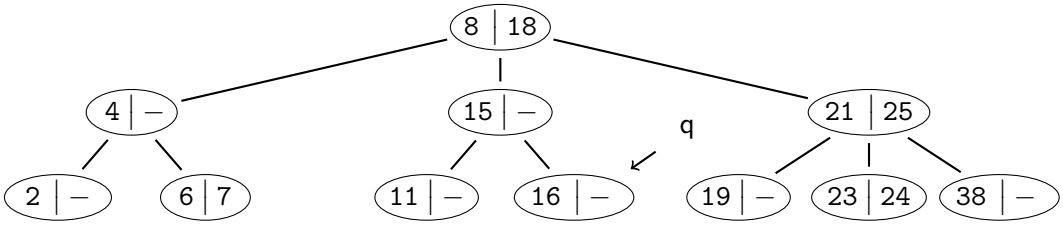
   c) Analise o tempo de execução dos seus algoritmos.

2. Navegando a árvore 2-3

   Imagine que você tem uma árvore 2-3.

   E imagine que você tem um ponteiro `q` apontando para o elemento que contém o valor `k`.

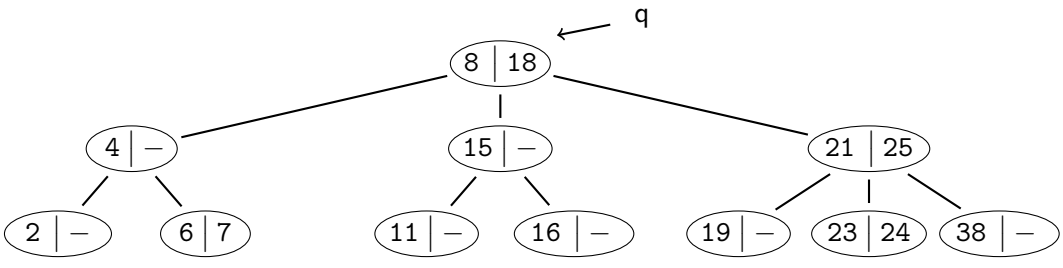
   Por exemplo, para `k = 16`



   A tarefa consiste em

   → *Reposicionar o ponteiro* `q` *sobre o elemento que contém o menor valor maior que* `k`

   No exemplo acima, isso nos daria



a) Apresente um algoritmo que resolve esse problema

Nota: Você pode assumir que os elementos possuem um ponteiro para o pai.

b) Analise o tempo de execução do seu algoritmo.

3. Fazendo contas

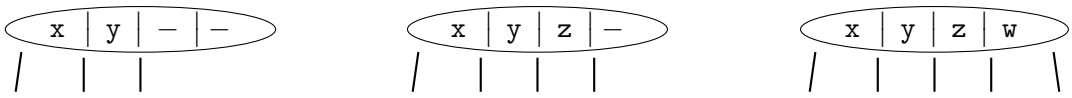
a) Qual é o menor número de elementos que uma árvore 2-3 com altura `k` pode ter ?

b) Qual é o maior número de elementos que uma árvore 2-3 com altura `k` pode ter ?

c) Imagine que você tem uma árvore 2-3 onde todos os elementos são do tipo 2.
É possível reconstruir essa árvore (utilizando elementos do tipo 3) de modo a reduzir a altura da árvore ?
Explique.

4. A árvore 3-4-5

A árvore 3-4-5 é uma árvore de busca onde os elementos podem ter 3, 4 ou 5 filhos



O detalhe interessante dessa árvore é que todos os seus elementos possuem ao menos 2 valores — (*com a possível exceção da raiz*)

Como usual, a tentativa de inserir um valor em um elemento que já está cheio provoca uma explosão

e o valor do meio `c` é inserido no nível de cima — (*ou se torna a nova raiz*)

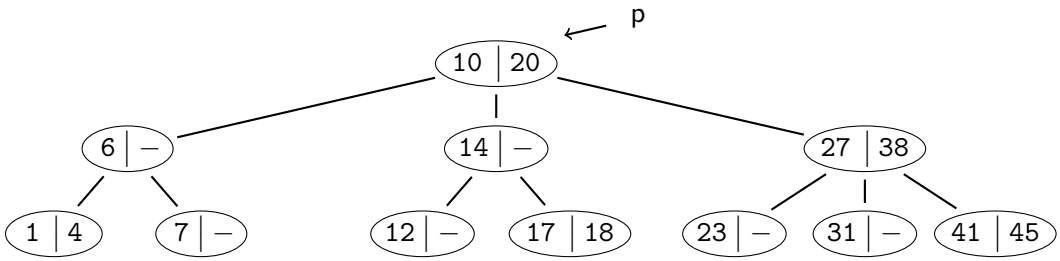
a) Apresente o algoritmo de inserção na árvore 3-4-5.

b) Analise o tempo de execução do seu algoritmo.

5. Inserção esperta (folha)

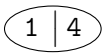
Imagine que você tem uma árvore 2-3 apontada pelo ponteiro p

Por exemplo,



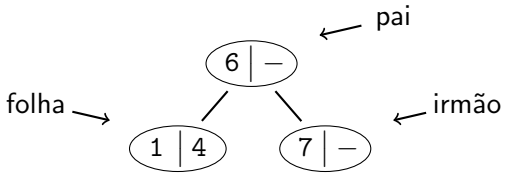
E imagine que nós queremos inserir o número 2 nessa árvore

Então nós sabemos que isso iria provocar a explosão da folha



Mas dessa vez nós podemos ser um pouco mais espertos.

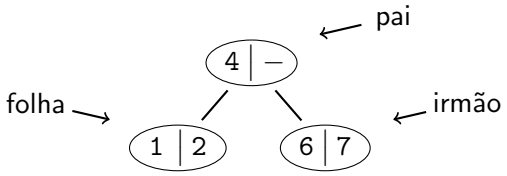
Quer dizer, note que o irmão dessa folha tem uma posição vazia



Daí que, nós podemos

- passar o 4 para o pai
- e o pai pode passar o 6 para o irmão

e resolver a situação da seguinte maneira



Quer dizer, a ideia é

→ *não fazer a explosão da folha,*
quando ela possui um irmão que tem uma posição vazia

a) Apresente um algoritmo de inserção para a árvore 2-3 que implementa essa ideia.

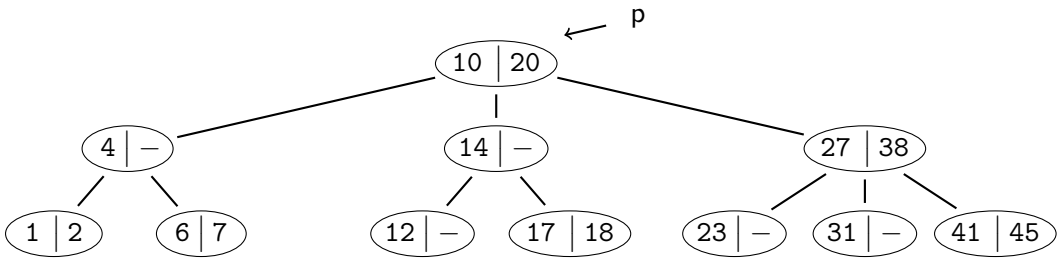
Nota: Você pode assumir que os elementos possuem um ponteiro para o pai.

b) Analise o tempo de execução do seu algoritmo.

6. Inserção esperta 2.0 (folha)

Agora nós vamos levar a ideia do exercício anterior um pouquinho mais longe.

Quer dizer, imagine agora que nós queremos inserir o 3 na árvore



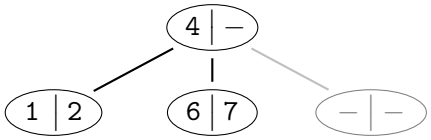
Novamente não há espaço na folha.

E dessa vez não há irmão com uma posição vazia.

Mas a folha não tem todos os irmãos possíveis.

Quer dizer, o pai é do tipo 2, e ele tem apenas 2 filhos.

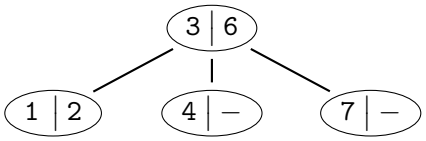
A esperteza então consiste em imaginar que existe um irmão imaginário (com duas posições vazias)



Daí, nós podemos

- passar o 3 para o pai
- e o pai passa o 4 para o irmão
- e o pai passa o 7 para o irmão imaginário

E a coisa fica assim



Quer dizer, a ideia dessa vez é

→ *não fazer a explosão da folha quando ainda existe espaço na família*

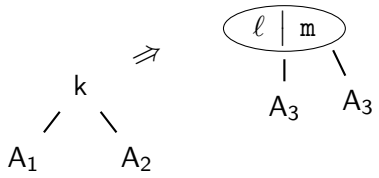
a) Apresente um algoritmo de inserção para a árvore 2-3 que implementa essa ideia.

b) Analise o tempo de execução do seu algoritmo.

7. **Inserção esperta 3.0 (caso geral)**

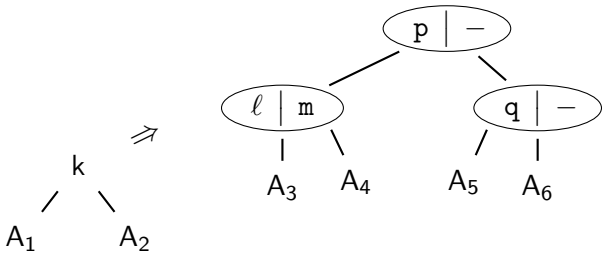
Agora nós vamos generalizar a ideia dos exercícios anteriores, para tentar evitar a explosão de um elemento interno da árvore.

Quer dizer, o cenário é aquele onde a inserção que ocorre na subárvore traz um elemento de volta

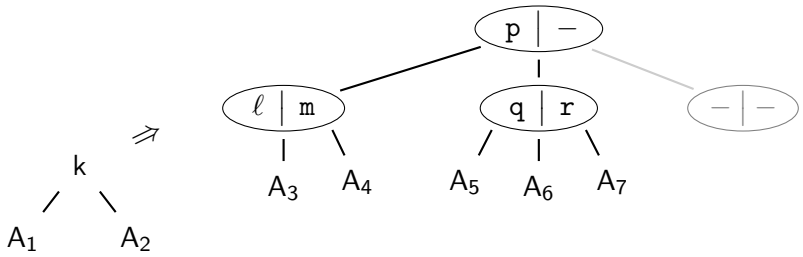


mas não há lugar para ele ali.

Daí, o primeiro caso é aquele onde o irmão tem uma posição vazia



E o segundo caso é aquele onde o irmão não tem posição vazia, mas o pai ainda tem espaço para um irmão imaginário



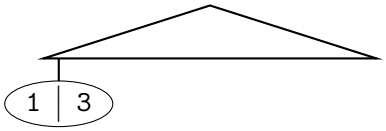
a) Analise qual é a solução adequada em cada caso para evitar a explosão.

b) Apresente um algoritmo de inserção para a árvore 2-3 que implementa essa ideia.

c) Analise o tempo de execução do seu algoritmo.

8. **Remoção do menor esperta**

O caso mais fácil da remoção do menor valor na árvore 2-3 é aquele onde ele se encontra em um registro do tipo 3



Porque daí, não há quase nada a fazer

A esperteza desse exercício é que esse pode ser o único caso que existe.

Quer dizer, a ideia é a seguinte

→ *No momento da descida pela esquerda,*
sempre que a gente encontra um elemento tipo 2,
a gente transforma ele em um elemento do tipo 3 (ou 4)

Daí que, ao chegar na folha que contém o menor elemento, nós sempre temos que ela é um elemento do tipo 3 (ou 4).

Como já foi indicado, essa solução utiliza elementos do tipo 4 temporários — (*que só existem durante o procedimento de remoção*)

Os elementos do tipo 4 são criados por operações de colapso que acontecem durante a descida.

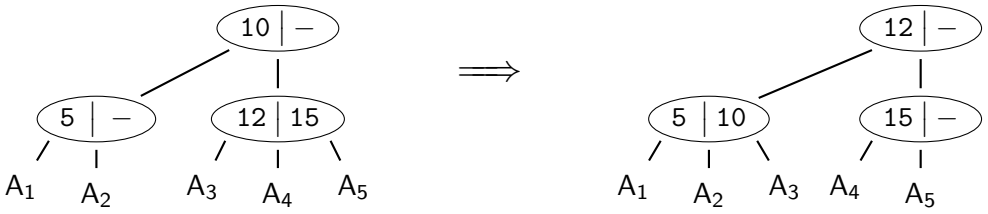
Daí na hora da subida, nós explodimos todos os elementos do tipo 4 que nós encontramos no caminho.

Certo.

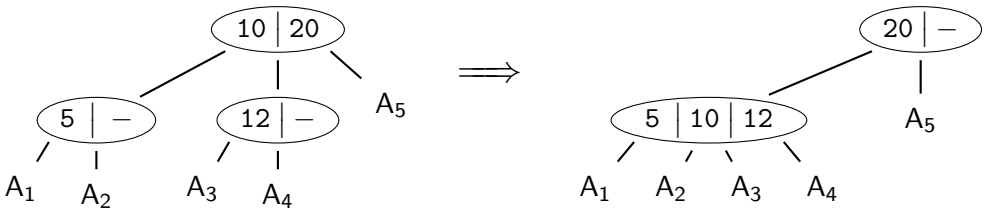
A lógica da descida pela esquerda fica assim

- Se o filho esquerdo é um elemento do tipo 3, nós apenas descemos até ele
- Senão, existem basicamente dois casos

Caso 1: (irmão do tipo 3)



Caso 2: (colapso)



Nota: Na raiz o colapso pode acontecer com um pai do tipo 2 — (*e isso diminui a altura da árvore*)

Em todos os outros casos, o colapso ocorre com um pai do tipo 3 ou 4 — (*porque?*)

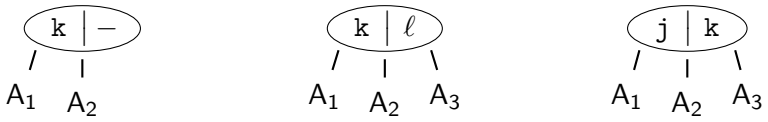
a) Apresente um algoritmo que implementa essa solução para a remoção do menor valor na árvore 2-3.

b) Analise o tempo de execução do seu algoritmo.

9. **Remoção esperta (caso geral)**

Agora nós vamos aplicar a ideia de exercício anterior para remover um elemento `k` qualquer da árvore.

Suponha primeiramente que `k` se encontra na raiz



Nesse caso, nós adaptamos o procedimento do exercício anterior para remover o menor valor `m` de A_2 (ou A_3).

E depois nós colocamos o `m` no lugar do `k`.

Agora suponha que o valor `k` não está na raiz.

Nesse caso,

- primeiro nós fazemos a descida até o `k`, transformando todo elemento do tipo 2 por onde a gente passa em um elemento do tipo 3 (ou 4)

- daí, nós removemos o menor elemento `m` de A_2 (ou A_3), e o colocamos no lugar de `k`

a) Apresente um algoritmo que implementa essa solução para a remoção de um valor qualquer na árvore 2-3.

b) Analise o tempo de execução do seu algoritmo.