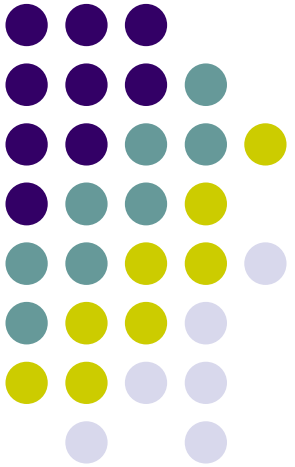
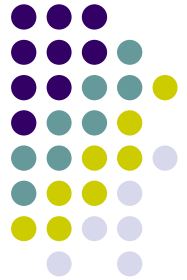
Árvores AVL

Prof. Leandro C. Fernandes

Estruturas de Dados

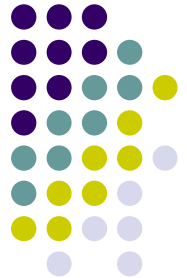


Revisão: Árvores de Busca Binária



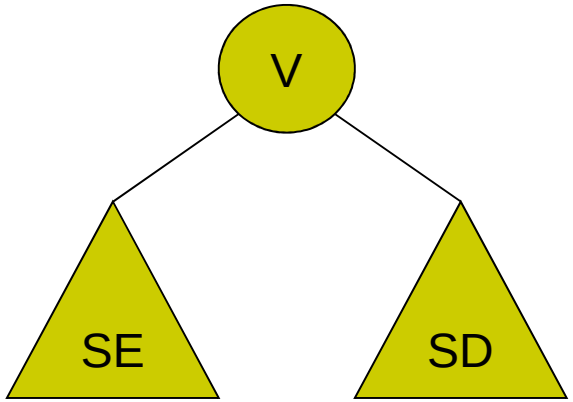
- Uma árvore de busca binária é uma árvore binária T tal que:
  - Cada nó interno *v* armazena um item (*k*,*e*), onde *e* é o elemento associado a chave *k*.
  - As chaves armazenadas em nós na sub-árvore esquerda de *v* são menores do que *k*
  - As chaves armazenadas em nós na sub-árvore direita de *v* são maiores do que *k*

Revisão: Árvores de Busca Binária

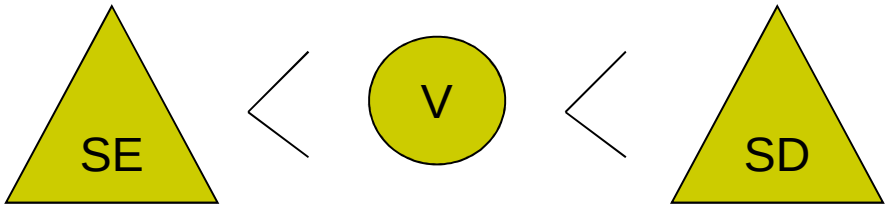


- Ou seja,...

Para qualquer nó:



É válida a relação:

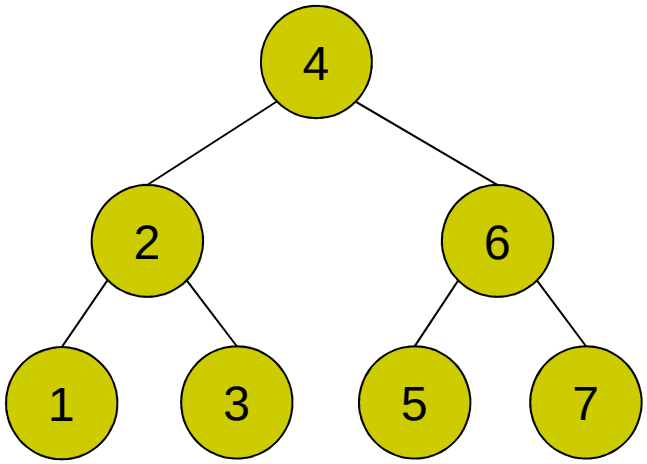


Busca em Árvores Bin. de Busca

- O tempo de execução da busca é dependente do formato das árvores, ou seja, da forma como os nó estão distribuídos

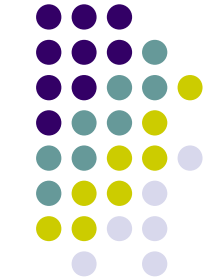
- Se for uma árvore completa (nós com sub-árvore vazias somente no último e penúltimo níveis), o tempo de computação, no pior caso, será $1+\log_2 n$ (altura da árvore)

- Inserir os valores:
  4, 2, 6, 1, 3, 5 e 7



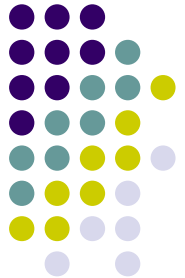
... ou ainda:
4, 6, 2, 5, 3, 1 e 7
4, 2, 6, 7, 1, 3 e 5
(...)

Problemas com as Árvores Binárias de Busca

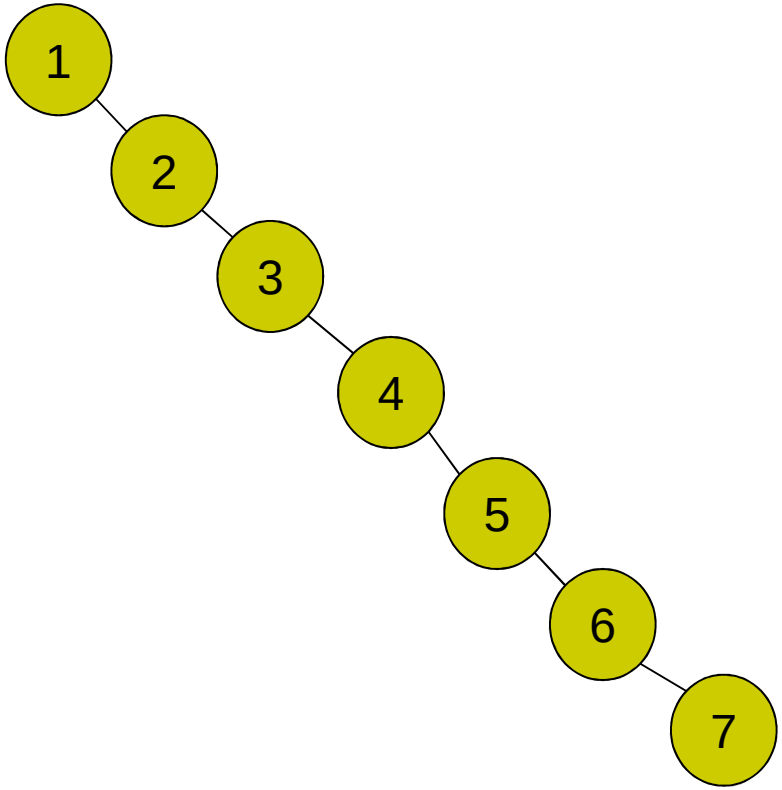


- Deterioração
  - quando inserimos utilizando a inserção simples, dependendo da distribuição de dados, pode haver deterioração.
- Uma má distribuição pode levar, no pior caso, a uma situação onde a árvore se torna uma lista linear
  - O pior caso corresponde a inserção de chaves completamente ordenadas
  - O tempo de busca, neste caso, é igual ao número de elementos da árvore, ou seja, O(*n*)

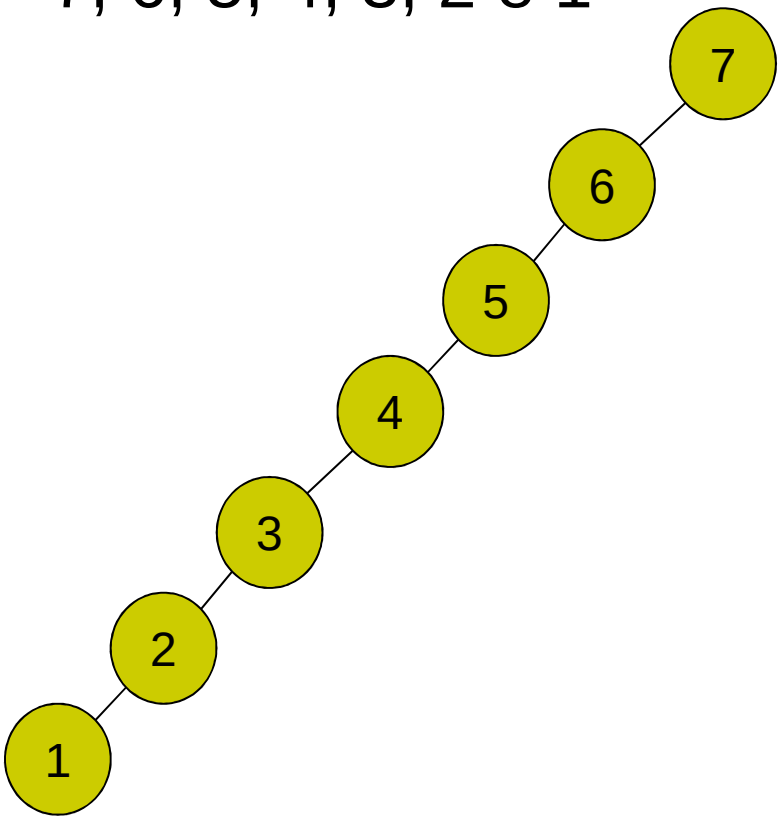
Problemas com as Árvores Binárias de Busca



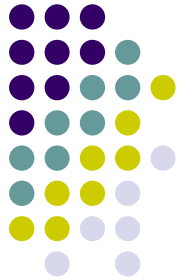
- Inserir os valores:
  1, 2, 3, 4, 5, 6 e 7



- Inserir os valores:
  7, 6, 5, 4, 3, 2 e 1



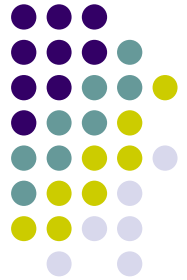
Problemas com as Árvores Binárias de Busca



Alternativa 1

- Se, após uma remoção ou inserção, a árvore resultante não for completa é possível utilizar um algoritmo de restabelecimento da estrutura
- Pode ser necessário percorrer todos os nós da árvore, apresentando complexidade O(*n*)
- O tempo para restabelecimento seria maior que o de inserções e remoções, apresentando complexidade O(log *n*). Assim, o custo de um algoritmo de restabelecimento é muito elevado

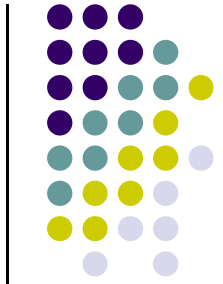
Problemas com as Árvores Binárias de Busca



Alternativa 2

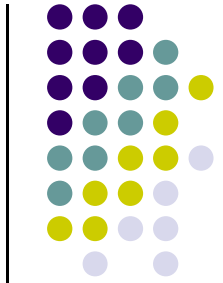
- Utilizar árvores binárias com altura da mesma ordem de grandeza ($O(\log_2 n)$) que a de uma árvore completa de mesmo número de nós ($1+\log_2 n$)
  - É desejável também que esta propriedade se estenda a todas as sub-árvores
  - Uma sub-árvore que satisfaz essa condição é denominada de árvore balanceada
- A forma de uma árvore balanceada é menos rígida que a de uma árvore completa
- Com isso, torna-se mais fácil o restabelecimento das condições de balanceamento, após inserções e remoções

Árvores AVL



- Proposta em 1962 pelos matemáticos russos G.M.Adelson-Velskki e E.M. Landis (AVL)
- Descreveram procedimentos para inserção e remoção dos nós nessas árvores, fundamentados no balanceamento (redistribuição da altura das sub-árvores).
- Assim, a árvore AVL é uma árvore com uma condição de balanço, porém não completamente balanceada.
- Árvores AVL permitem inserção/remoção e (re)balanceamento dos nós de maneira (aceitavelmente) rápida.

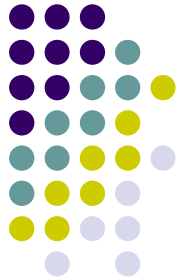
Árvores AVL



Definição: árvore binária de busca construída de tal modo que a altura de sua sub-árvore direita difere da altura da sub-árvore esquerda em no máximo 1 unidade

- Para sua compreensão, deve-se estudar:
  - O que pode acontecer quando um novo nó é inserido em uma árvore balanceada ?
  - O que pode acontecer quando um nó é removido de uma árvore balanceada ?

Inserção em Árvore AVL

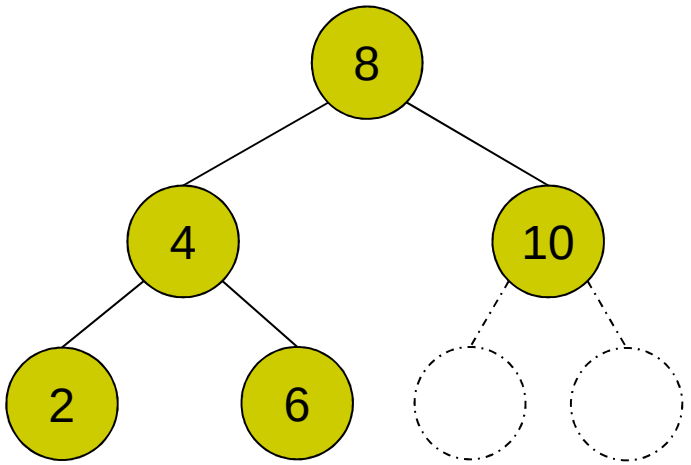


- Considere uma árvore AVL com raiz *v* e sub-árvores *L* (left) e *R* (right)
  - Suponha que a inserção deve ser feita na sub-árvore da esquerda.
  - Existem 3 casos possíveis de inserção:
    - Se $h_L = h_R$, então *L* e *R* ficam com alturas diferentes mas continuam balanceadas.
    - Se $h_L < h_R$, então *L* e *R* ficam com alturas iguais e balanceamento foi melhorado.
    - Se $h_L > h_R$, então *L* fica ainda maior e balanceamento foi violado.

Exemplo

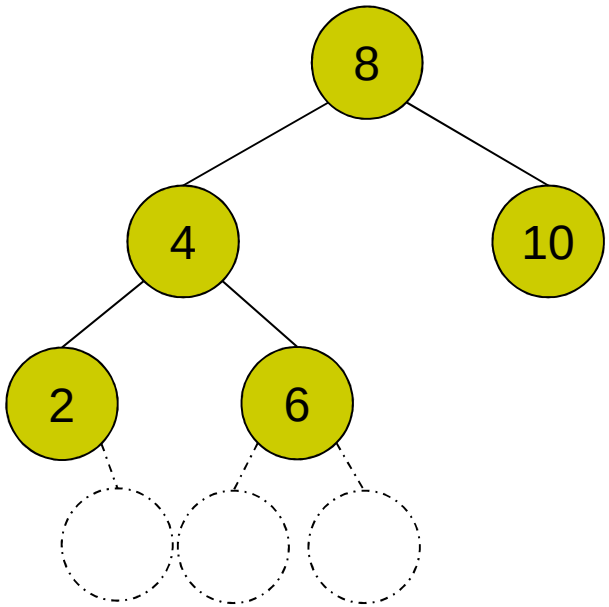
- Inserção de nós com chaves 9 e 11 pode ser feita sem a necessidade de balanceamento
  - Inclusive melhorando o balanceamento da árvore!

- Inserção de nós com chaves 3, 5 e 7 exige um re-balanceamento da árvore

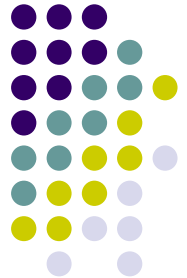


Exemplo

- Inserção de nós com chaves 9 e 11 pode ser feita sem a necessidade de balanceamento
  - Inclusive melhorando o balanceamento da árvore!
- Inserção de nós com chaves 3, 5 e 7 exige um re-balanceamento da árvore



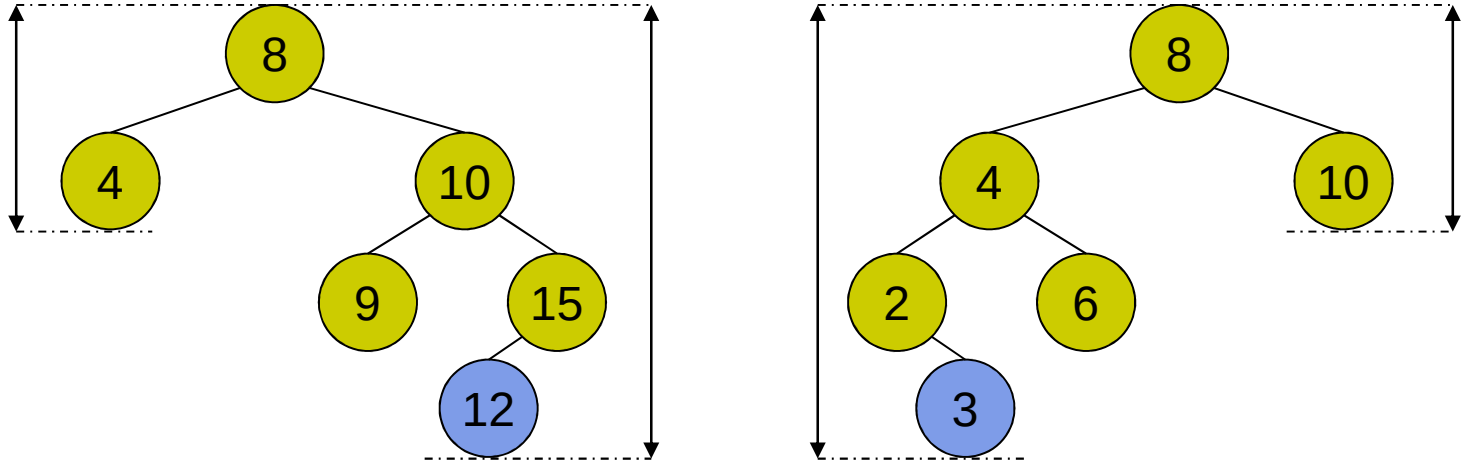
Balanceamento



- Fator de Balanceamento (FB) de um nó *v* pode ser definido como:
  - Altura da sub-árvore direita do nó menos a altura da sub-árvore esquerda do nó: h_R-h_L
- Os problemas de balanceamento das árvores AVL podem ser mapeados em 2 casos:
  - O nó raiz de uma sub-árvore tem FB=2 (-2) e tem um filho com FB = 1 (-1) o qual tem o mesmo sinal que o FB do nó pai.
  - O nó raiz de uma sub-árvore tem FB=2 (-2) e tem um filho com FB = -1 (1) o qual tem o sinal oposto ao FB do nó pai.

Caso 1

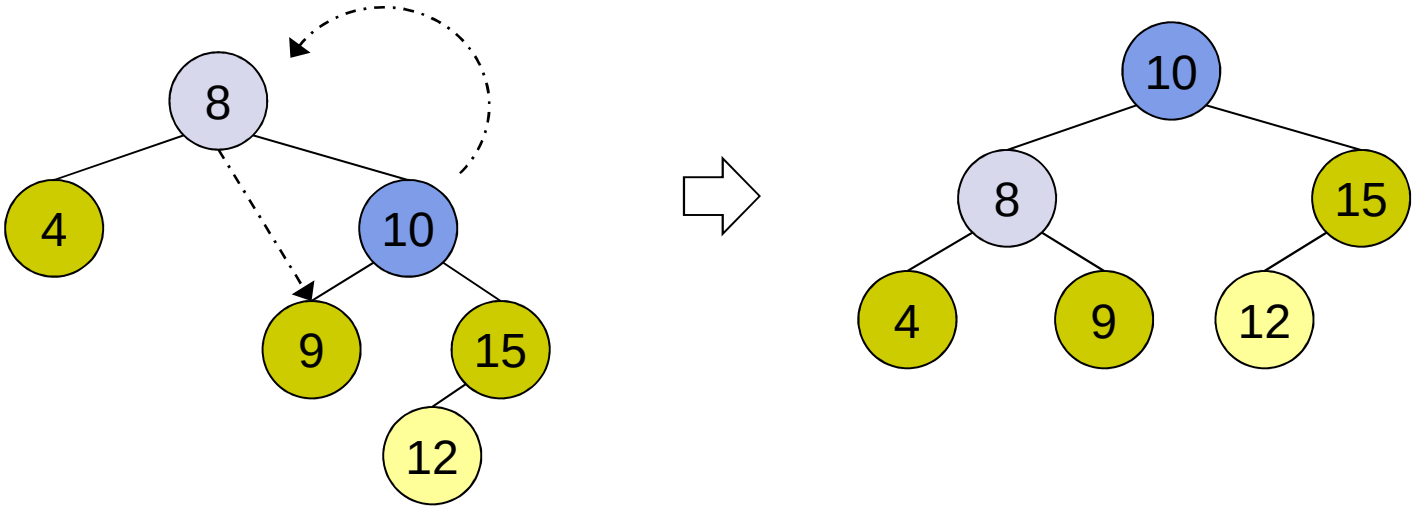
- Nó raiz da sub-árvore tem FB=2 (-2) e tem filho com FB=1 (-1) o qual tem o mesmo sinal que o FB do nó pai



- Solução: rotação simples sobre o nó de FB=2 (-2)
  - Rotações são feitas à esquerda quando FB é positivo e à direita quando FB é negativo.

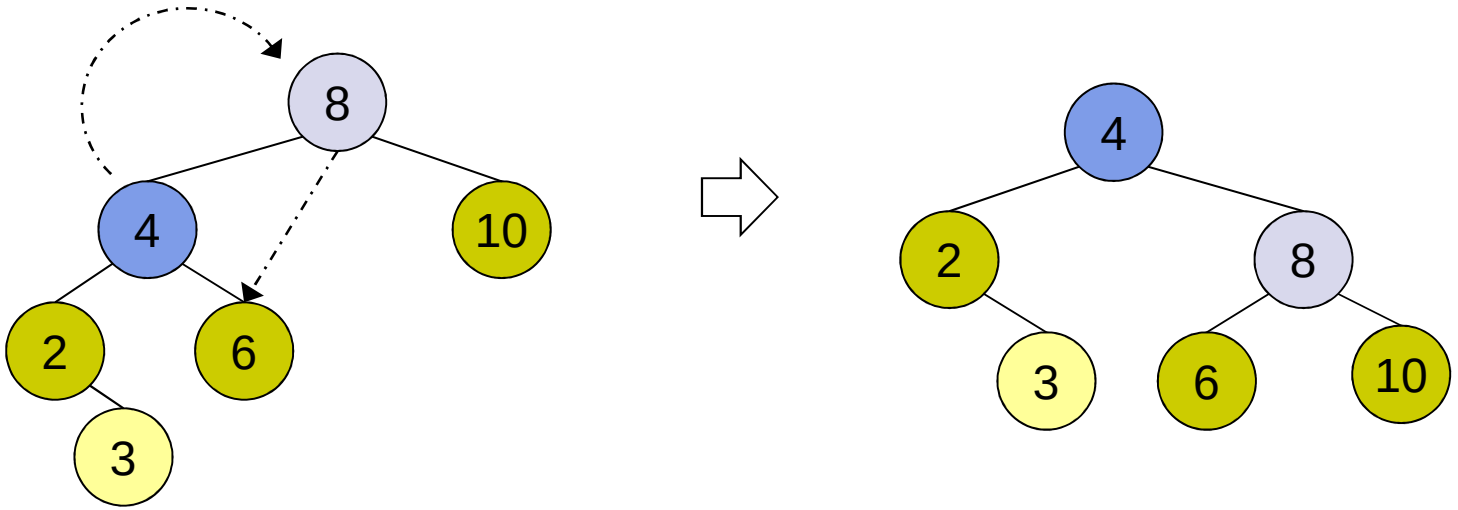
Caso 1 (rotação à esquerda)

- FB(8) = 3-1 = 2
- FB(10) = 1



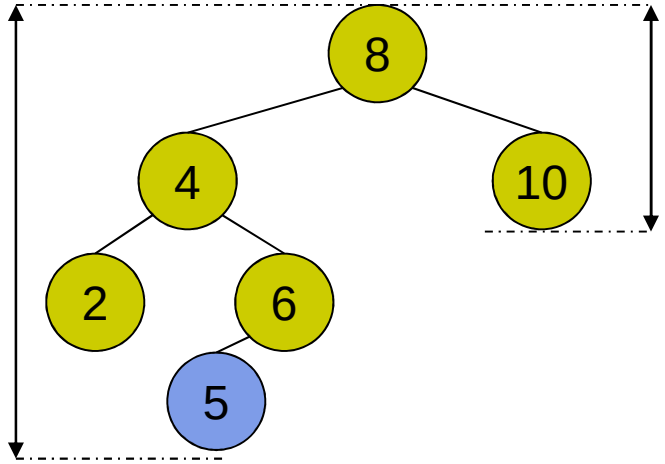
Caso 1 (rotação à direita)

- FB(8) = 1-3 = -2
- FB(4) = -1



Caso 2

- Nó raiz da sub-árvore tem FB=2 (-2) e tem filho com FB=-1 (1) o qual tem o sinal oposto que o FB do nó pai



- Solução: rotação dupla
  - Rotação sobre o nó com FB=-1 (-1) na direção apropriada
  - Rotação sobre o nó com FB=2 (2) na direção oposta

Caso 2 (rotação dupla)

- FB(8) = 1-3 = -2
- FB(4) = 1

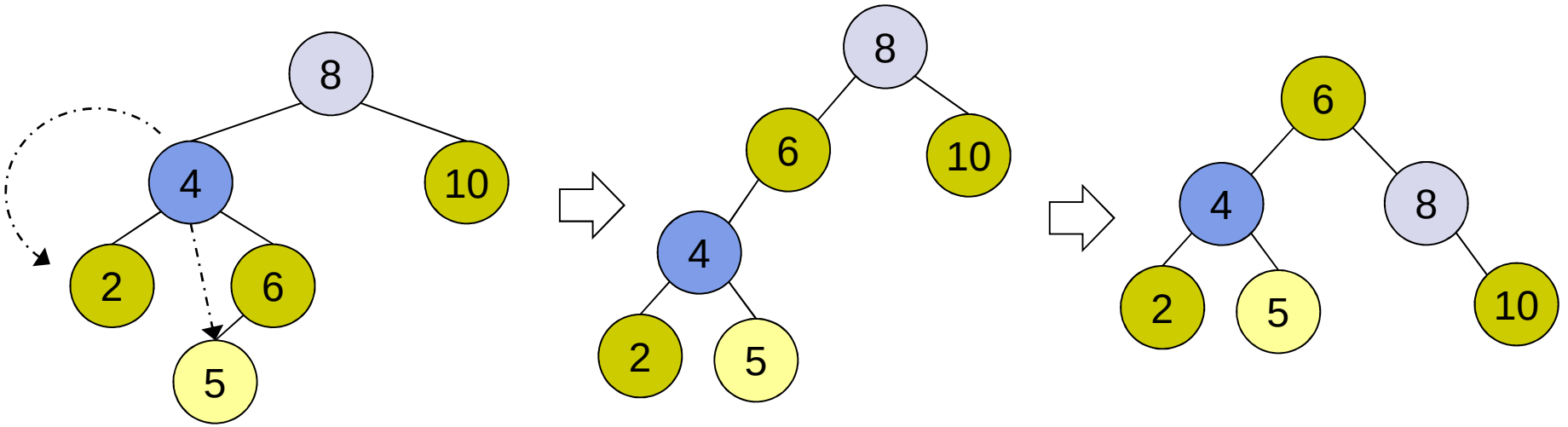


Ilustração esquemática das rotações

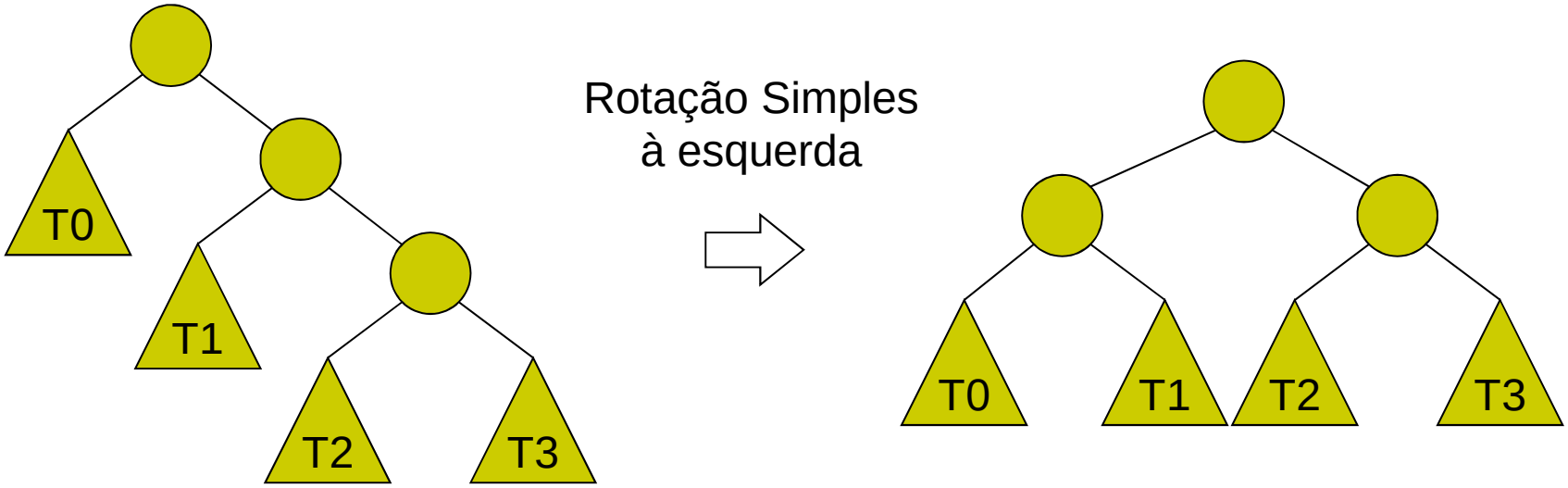


Ilustração esquemática das rotações

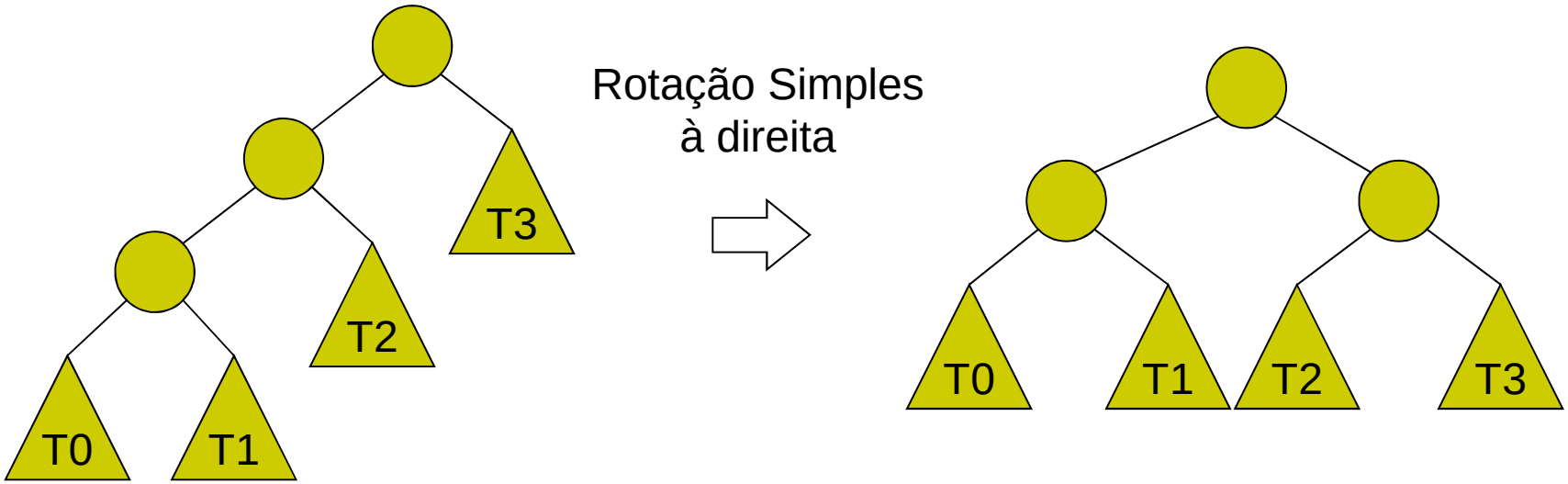


Ilustração esquemática das rotações

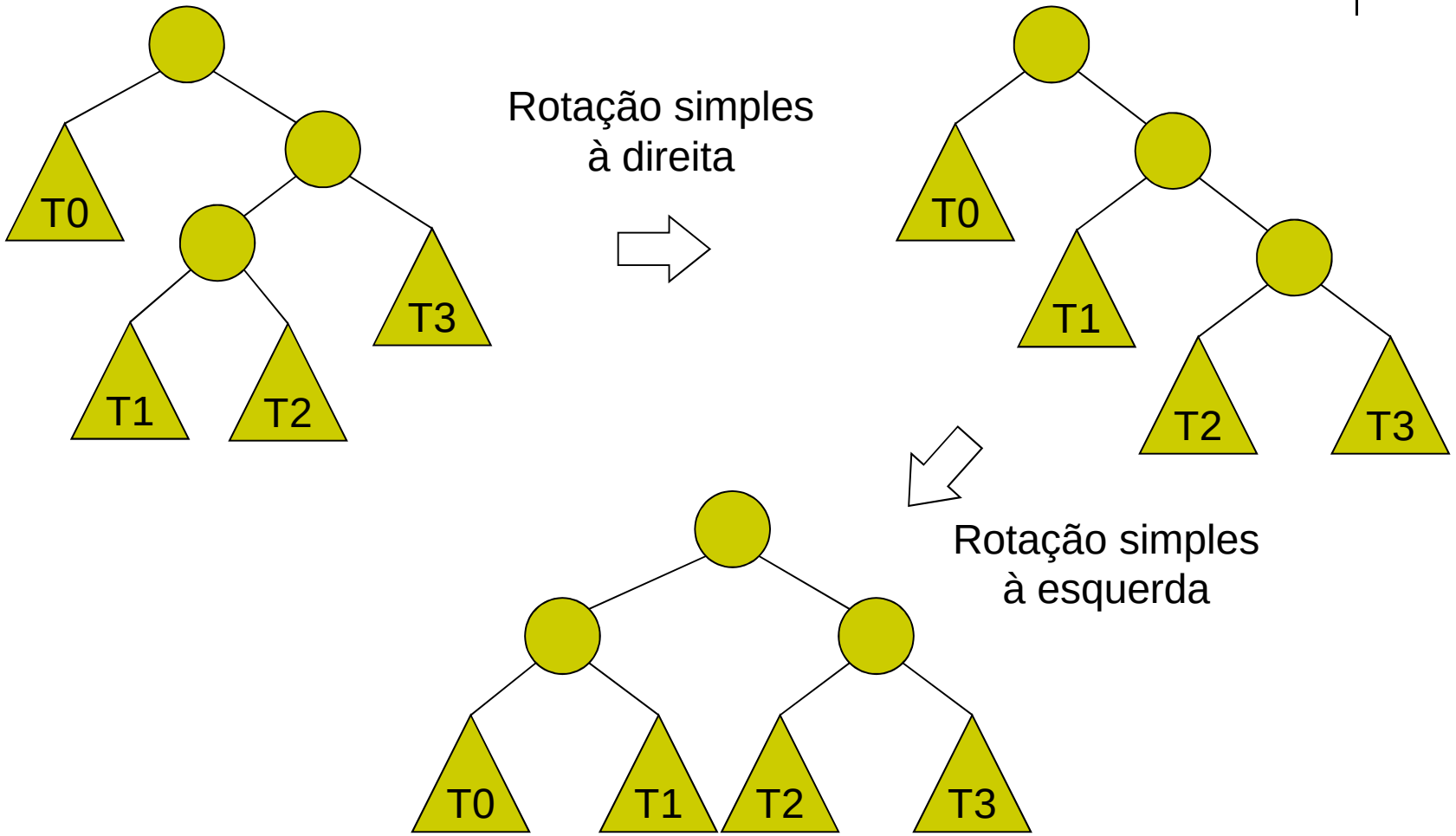
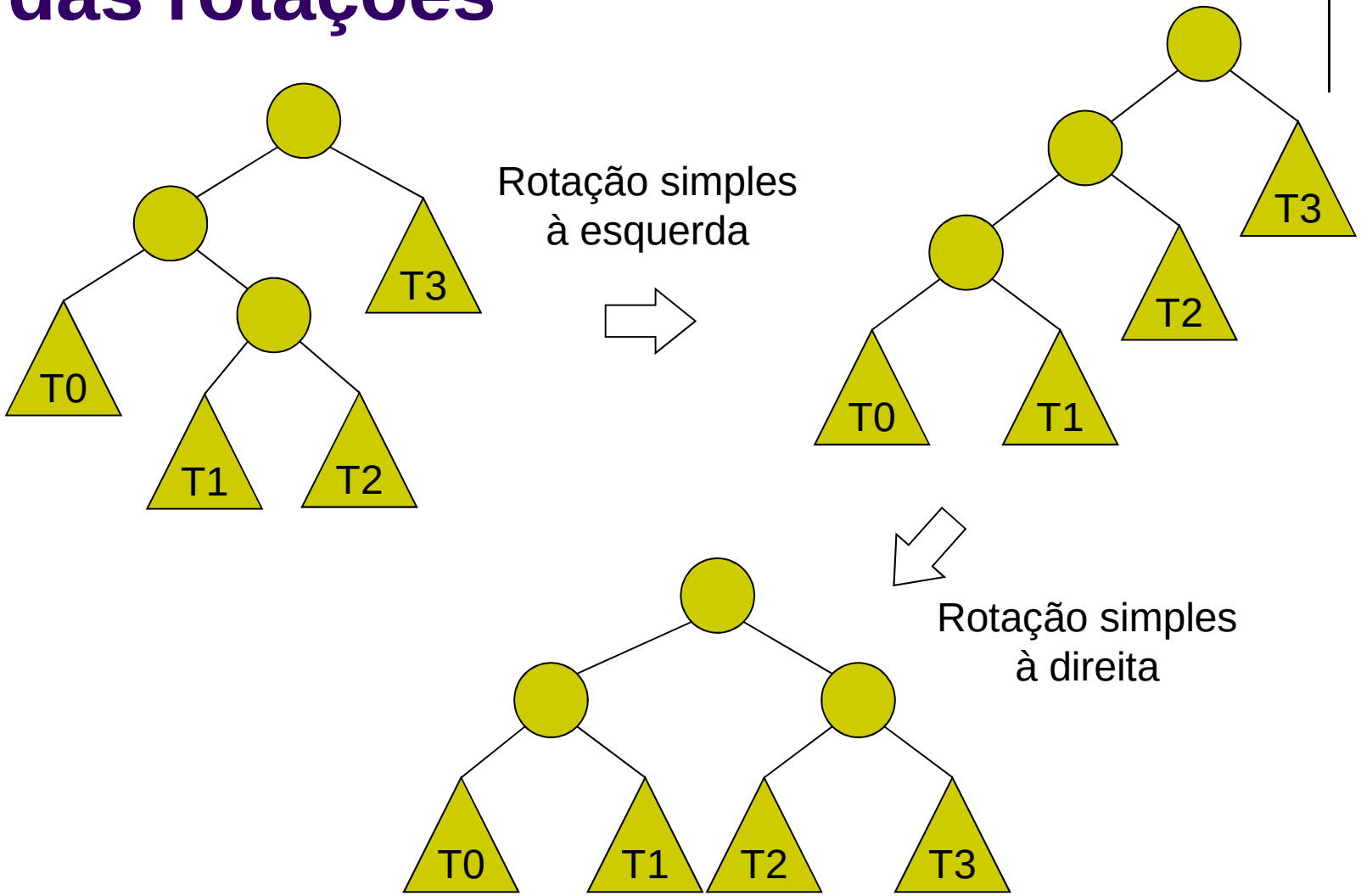
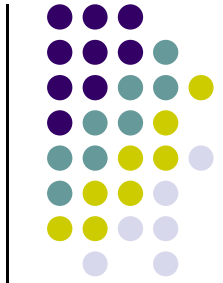


Ilustração esquemática das rotações



Algoritmos para rotação



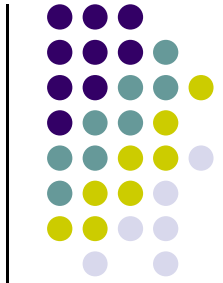
```
algoritmo rotacao_esquerda(p)
início
   q = p->dir    /* q filho de p */
   temp = q->esq
   q->esq = p
   p->dir = temp
   p = q
fim

algoritmo rotacao_direita (p)
início
   q = p->esq    /* q filho de p */
   temp = q->dir
   q->dir = p
   p->esq = temp
   p = q
fim
```

Implementação da Inserção

- Seja T uma árvore AVL e um novo nó q com chave x a ser incluído

```
Efetuar busca em T para verificar se x ∈ T
Se PERTENCER
então o processo deve ser encerrado
senão a busca encontra o local correto do novo nó
      chamar função de inserção
      verificar se existe algum nó desbalanceado
      se EXISTIR
      então efetuar o re-balanceamento
         /* Como visto, pelo valores de FB dos nós,
            determina-se a rotação apropriada */
      senão o processo termina
```

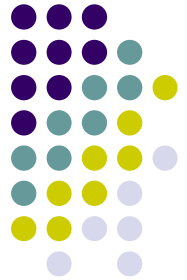
Implementação da Inserção



- Problema:
  - Como verificar se existe algum nó ficou desbalanceado após uma inserção?

- Solução:
  - Percorrer o caminho que vai do novo nó em direção à raiz, atualizando os FB´s dos nós deste caminho
  - Neste percurso, conferir o balanceamento de cada nó
  - Ocorrendo um nó desbalanceado, executar a operação de rotação adequada

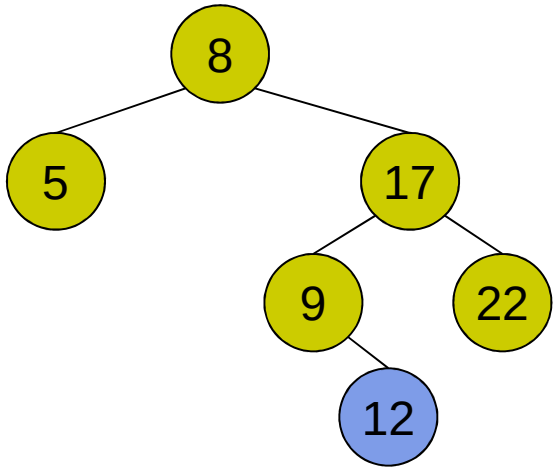
Remoção em Árvore AVL



- Processo semelhante ao de Inserção.
- Uma busca localiza o nó a ser removido
- Remove-se o nó desejado utilizando o algoritmo de remoção de árvores binárias
- Após a remoção, é verificado se a árvore tornou-se desbalanceada examinando os nós no caminho da raiz até a folha
- Para cada nó desbalanceado, utiliza-se a rotação apropriada. Ao balancear um nó, outros nós do caminho até a raiz podem se tornar desbalanceados
  - No pior caso, O(log *n*) rotações podem ser necessárias

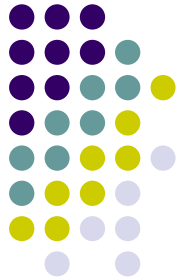
Exercícios

1. Executar o balanceamento apropriado para a árvore abaixo após a inserção do nó com chave 12



2. Implementar os algoritmos para as rotações do caso 2

Exercícios



3. Implementar o algoritmo de inserção para árvores AVL
4. Implementar o algoritmo de remoção para árvores AVL
5. Estudar o rebalanceamento para diferentes situações usando a applet de AVL

(http://www.site.uottawa.ca/~stan/csi2514/applets/avl/BT.html)