Árvores Splay

Mestrado em Ciência da Computação

Estruturas de Dados

Prof. Dr. Paulo Roberto Gomes Luzzardi

Aluna: Nelsi Warken

Características

- Inventada por Adel' son Vel' skii e Landis - 1962.
- Estudos e artigos: Daniel Sleator e Robert Tarjan.

- Também chamada de Árvores Auto-Ajustadas ou Árvore de Afunilamento.

- É um tipo de Árvore Binária de Pesquisa (BST)
  - grau de cada nó <= 2;
  - máximo 2 filhos: TE e TD;
  - todos nós à esquerda contém subárvores com valores menores ao nó raiz da subárvore;
  - todos nós à direita contém valores maiores ao nó raiz da subárvore.

Características

- Árvores mais simples que AVL:
- - não forçam o equilíbrio;
- - não mantem informação de altura.
- É uma árvore auto-ajustável, alterações tendem ao equilíbrio.
- É utilizada para aplicações específicas, onde se realizam uma sequência de operações em um universo ordenado.
- Possui três operações básicas: pesquisa, inserção e remoção. Em todas operações a árvore faz SPLAY.

O que é fazer SPLAY?

É trazer um elemento X para a raiz da árvore (BTT- Bring To Top), utilizando sucessivas rotações e tantas quanto necessárias.

Objetivos:

Minimizar o número de acessos para achar a chave requerida.

Otimizar a eficiência das operações, através da frequência com que cada nó é acessado, mantendo estes nós na parte superior da árvore.

Método Splaying

- Torna mais acessível o que é mais usado;
- ajusta a estrutura da árvore à frequência de acesso aos dados;
- junto à raiz estão os elementos:
  - mais usados
  - mais recentes
- os elementos mais inativos ficam mais "longe" da raiz;
- o recurso é implementado por meio de rotações nos nós.

Rotações geradas pelo Splay

Rotação Simples:

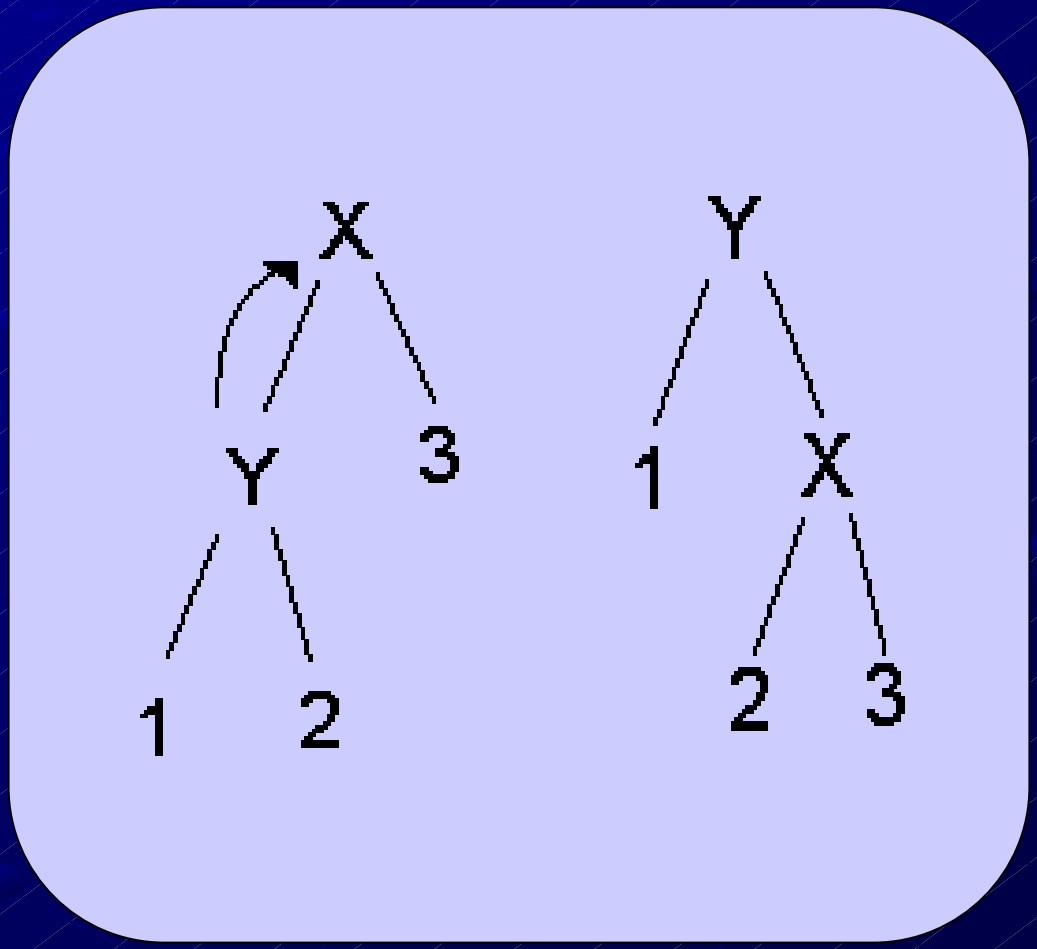
- Zig direita
- Zig esquerda

Rotação Dupla:

- Zig-zig direita;
- Zig-zig esquerda;
- Zig-zag esquerda e direita.

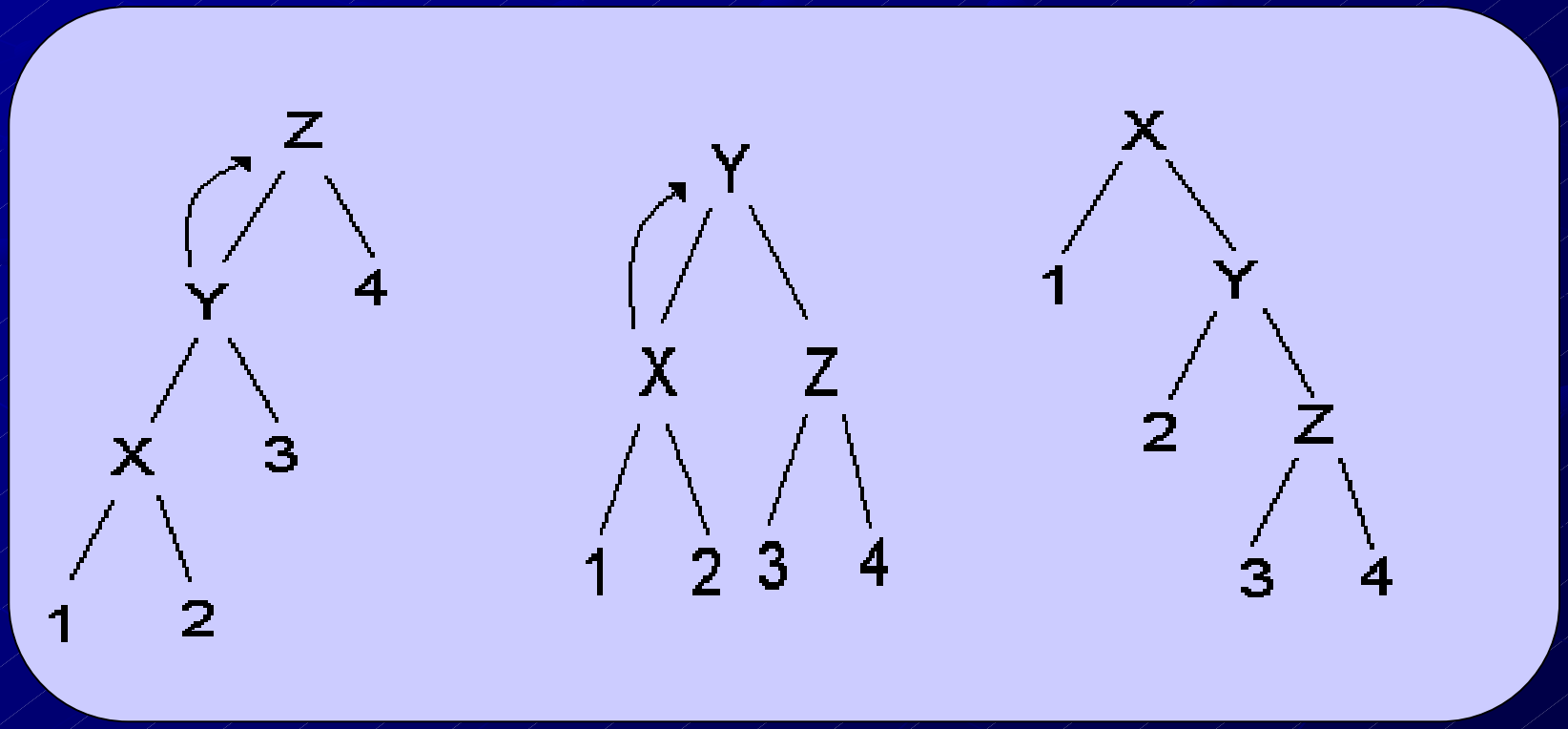
Rotação: ZIG

- Nesta rotação, o filho direito do elemento y, ficará o filho esquerdo do elemento x, que era pai de y.
- É a rotação de um nó sobre seu pai, permanecendo a árvore com a mesma altura.



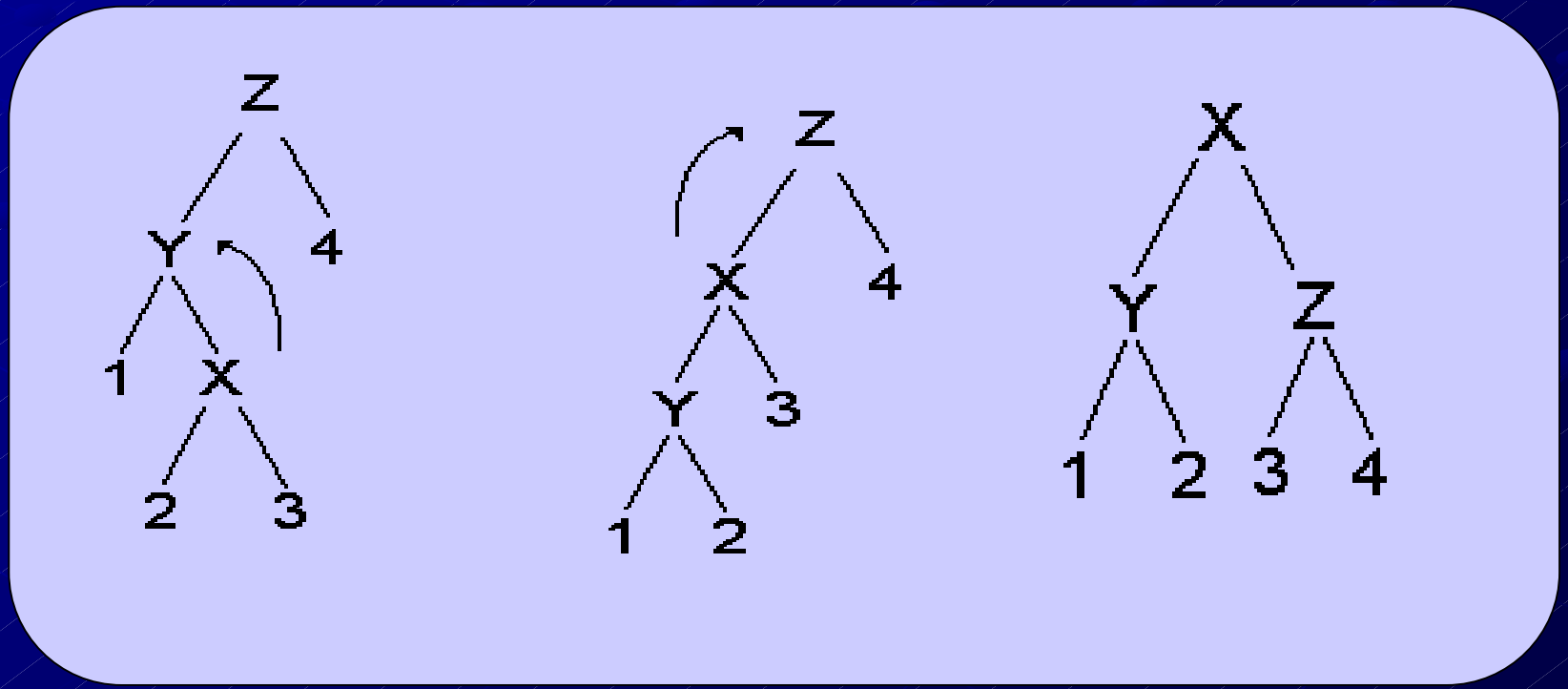
Splay Tree: zig-zig

- Para fazer o zig-zig de x, primeiro é realizado o zig do pai de x (que é y).
  - Após, é feito o zig de x.
  - Conclusão: no fundo é feito zig (y) e zig (x), respectivamente.



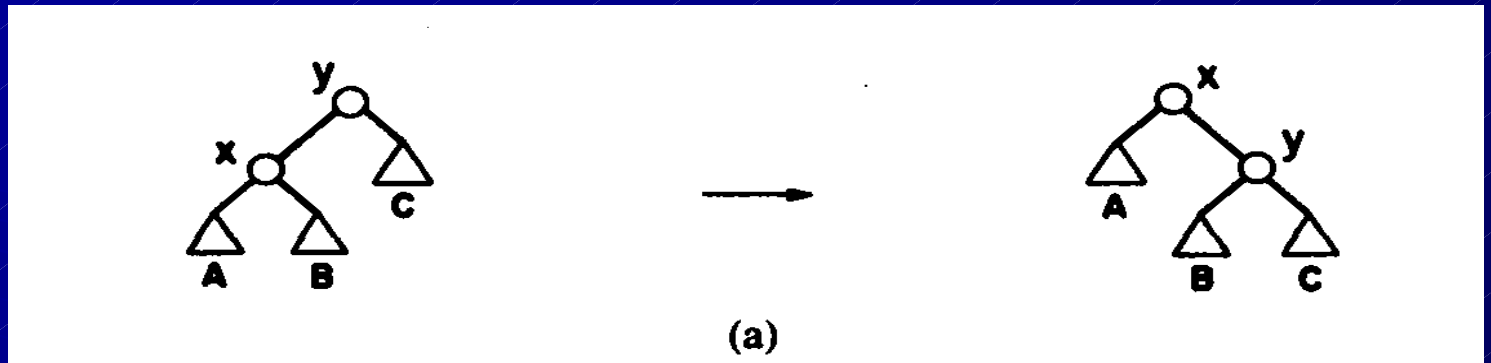
Splay Tree: zig-zag

- Ao contrário de ZIG-ZIG, primeiro realizar o ZIG de X com o pai de X (que é o Y).
- Depois fazer o ZIG de X com avô de X (que é o Z).
- Conclusão: No fundo corresponde a fazer ZIG (X) e ZIG (X).

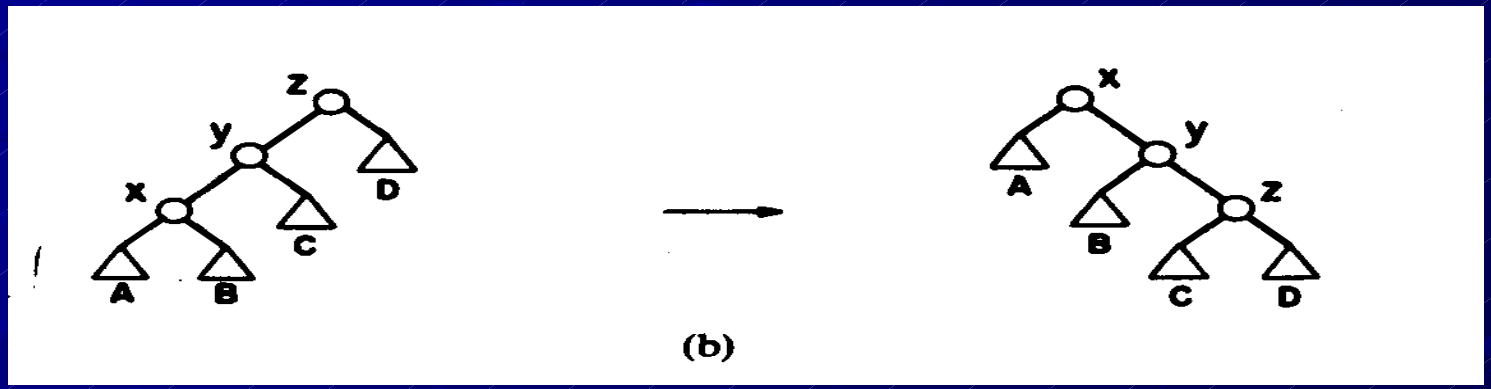


EXEMPLOS

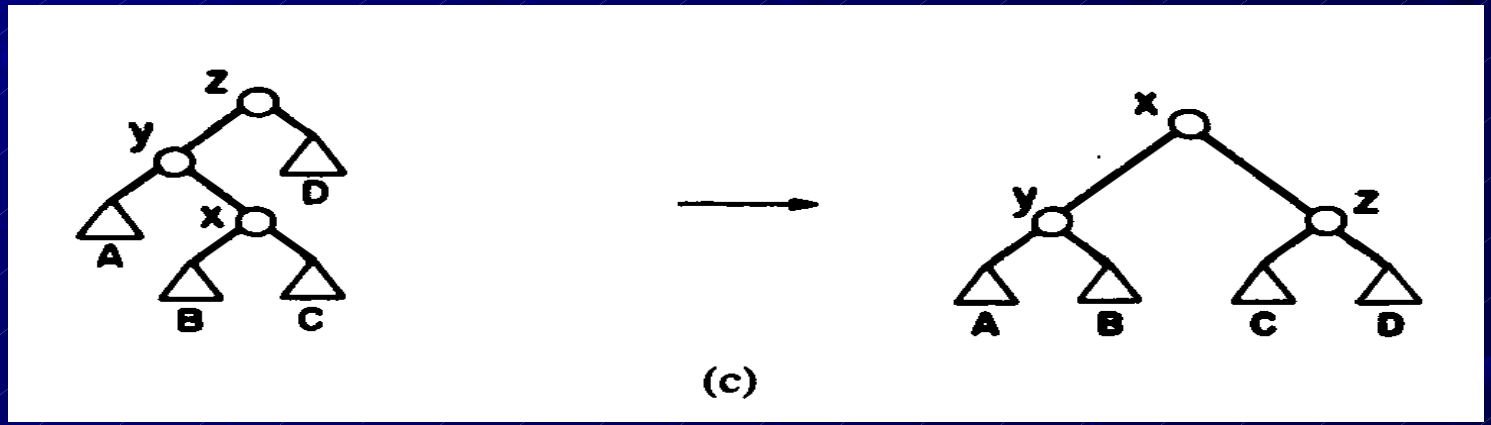
a) Zig: rotação simples.



(b) Zig-zig: duas rotações simples

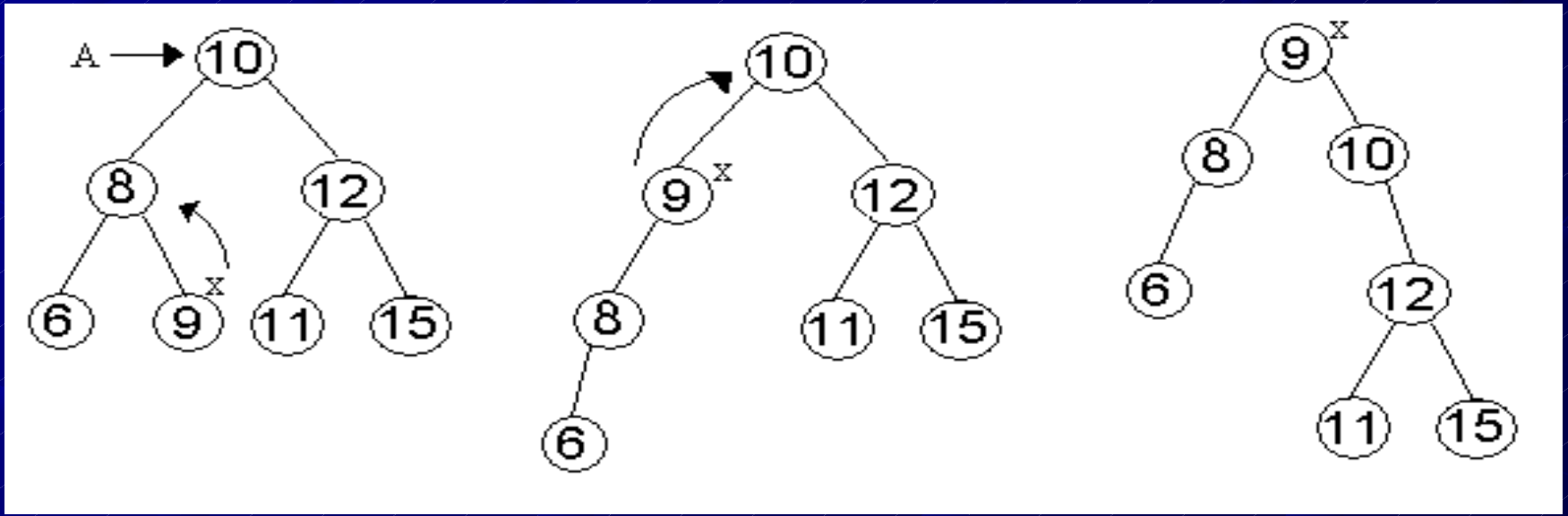


(c) Zig-zag: rotação dupla.

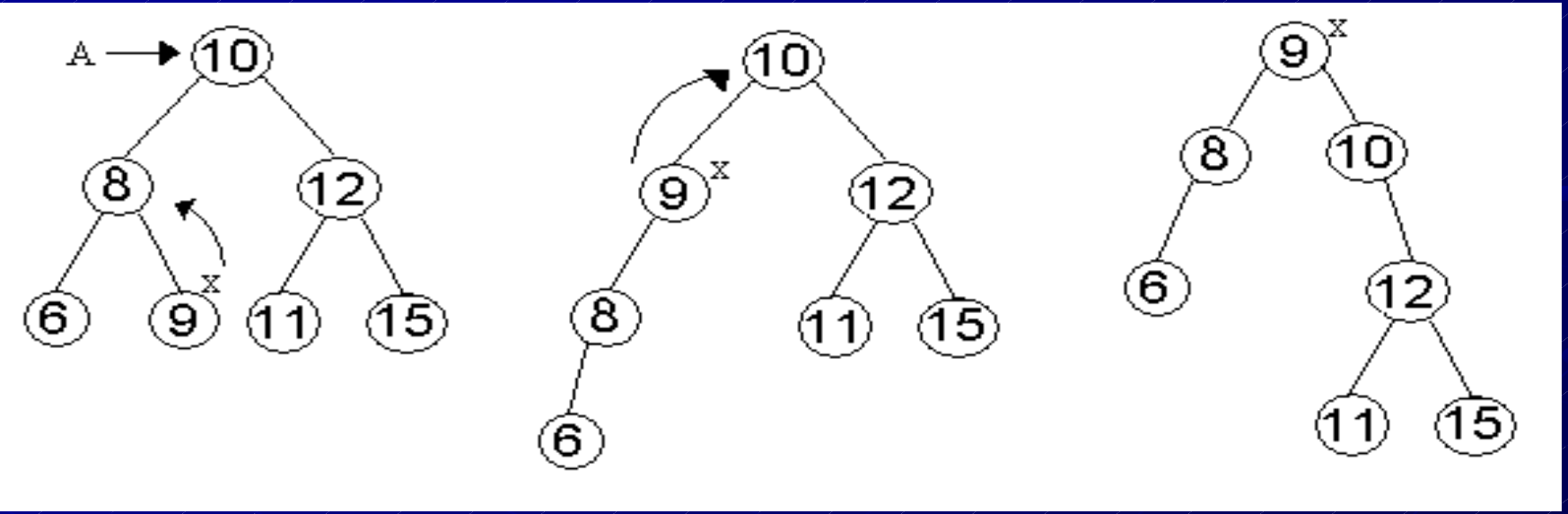


Exemplo de Pesquisa(9, A):

1) Pesquisa tipo BST até encontrar, se existe então fazer SPLAY do x (neste caso 9).
   Caso não exista, fazer SPLAY do último nodo não nulo encontrado na busca (elemento menor ou maior, mais próximo de 9.
2) Neste caso, para fazer SPLAY, utilizar o ZIG-ZAG(x).
3) Árvore final com o x na raiz.



Algorítmo de Pesquisa(x, A)



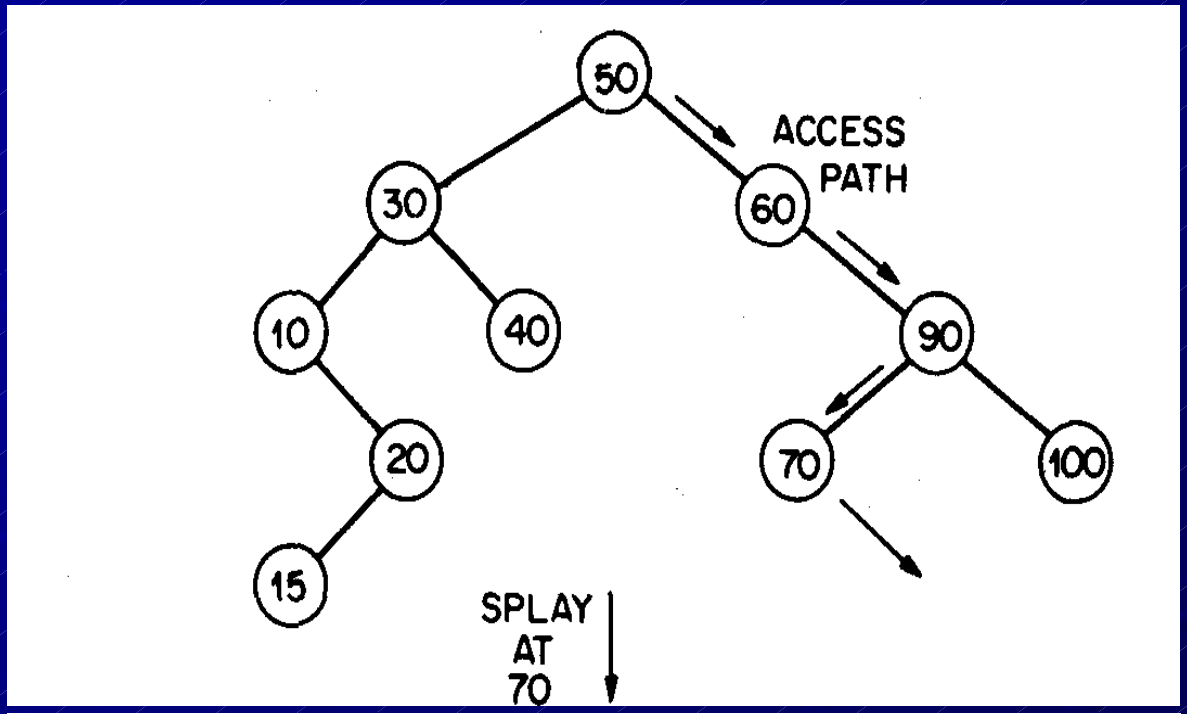
Início
Faz percurso BST até encontrar.
Se existe o elemento x na árvore A
Faz SPLAY do elemento x
Senão
Faz SPLAY do sucessor esquerdo ou direito mais próximo de x
Fim

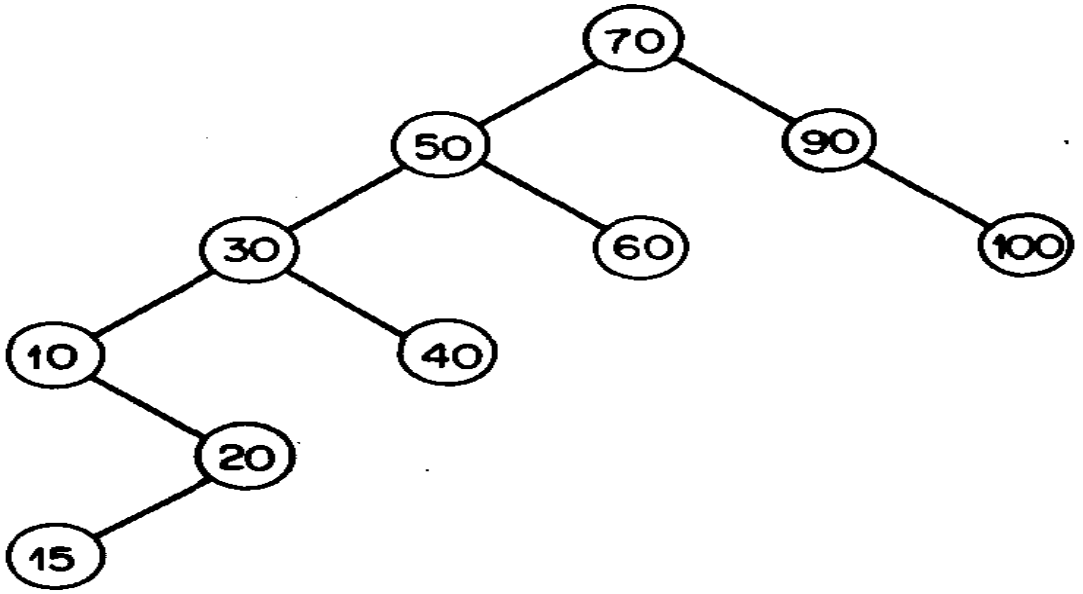
Exemplo de Pesquisa

Pesquisar o valor 80: (não existe na árvore)

Como a pesquisa parou em 70, é este o elemento que vai para a raiz.

Se a primeira árvore não tivesse o nodo com valor 70, a pesquisa pararia em 90 e o valor 90 iria para a raiz.



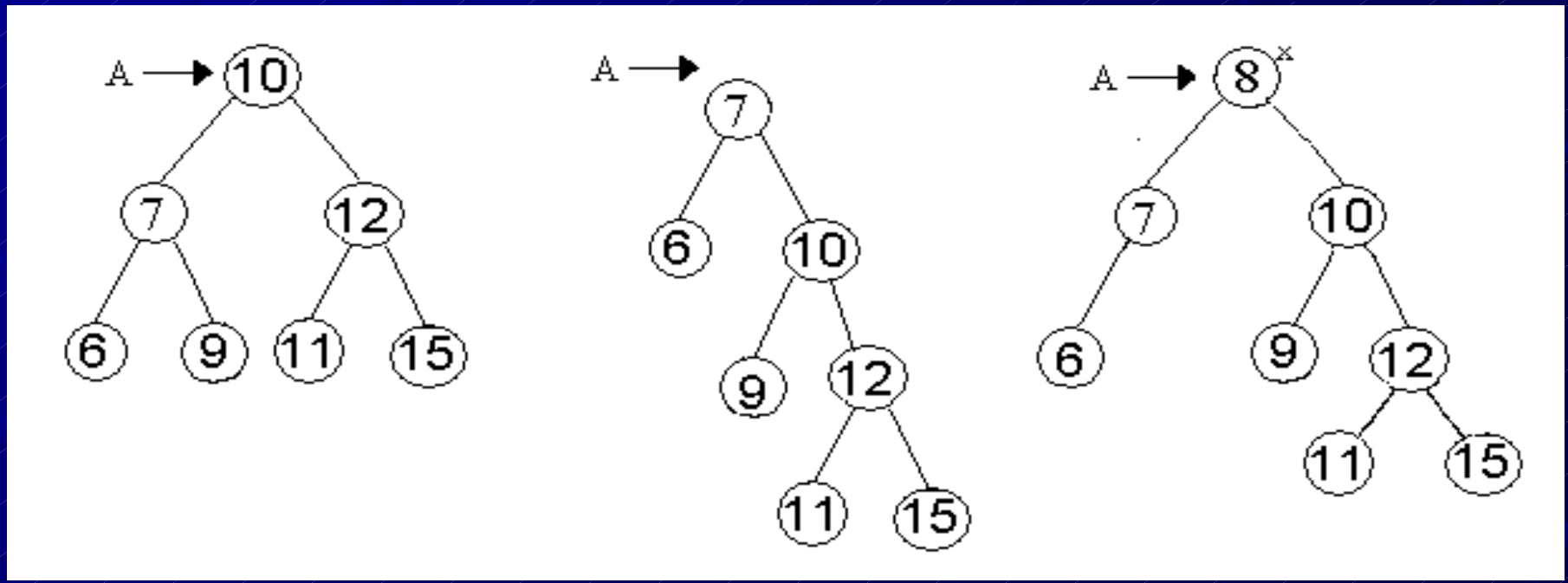


Exemplo de Inserção(8, A)

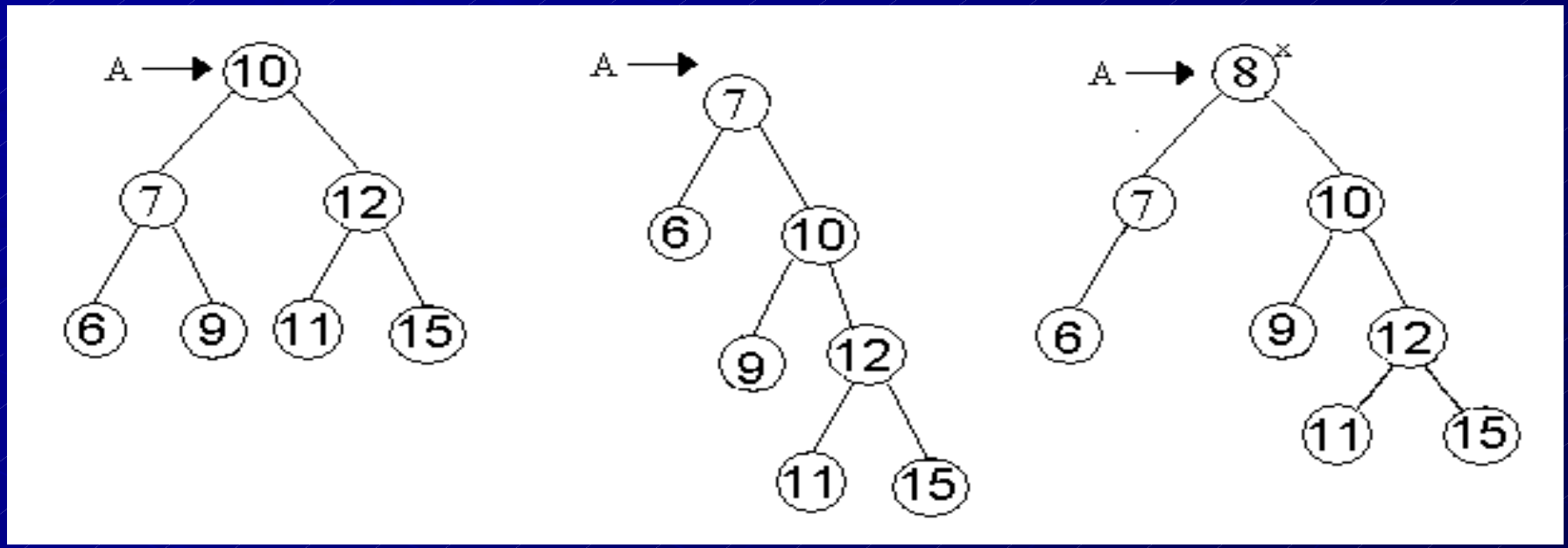
1) SPLAY(8).

2) 7 vai para a raiz (primeiro elemento menor que 8).

3) Inserir 8, 7 ficará o seu filho esquerdo, e 10 o seu filho direito.



Algorítmo de Inserção(A, x)



Início

Faz SPLAY do elemento x .

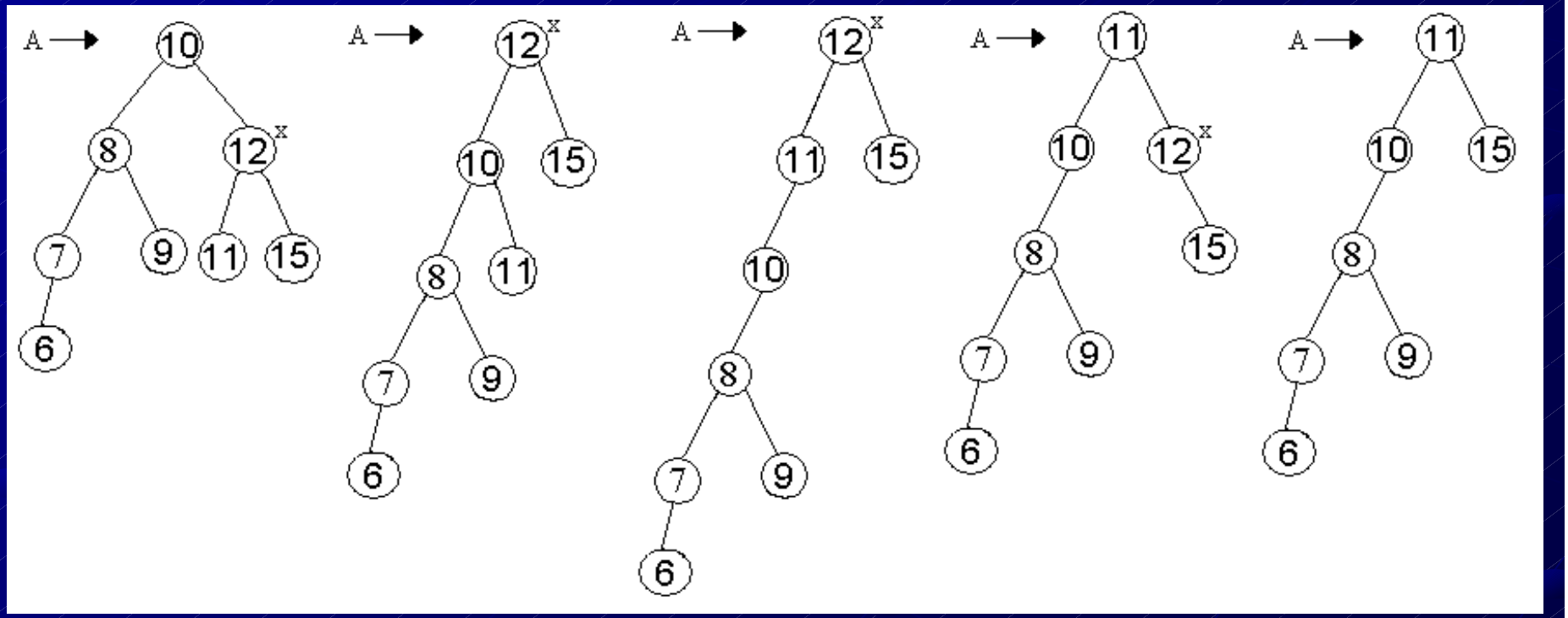
Como não existe, o elemento menor mais próximo de x, fica na raiz.

Agora é só inserir x, que passa a ser a raiz da árvore.

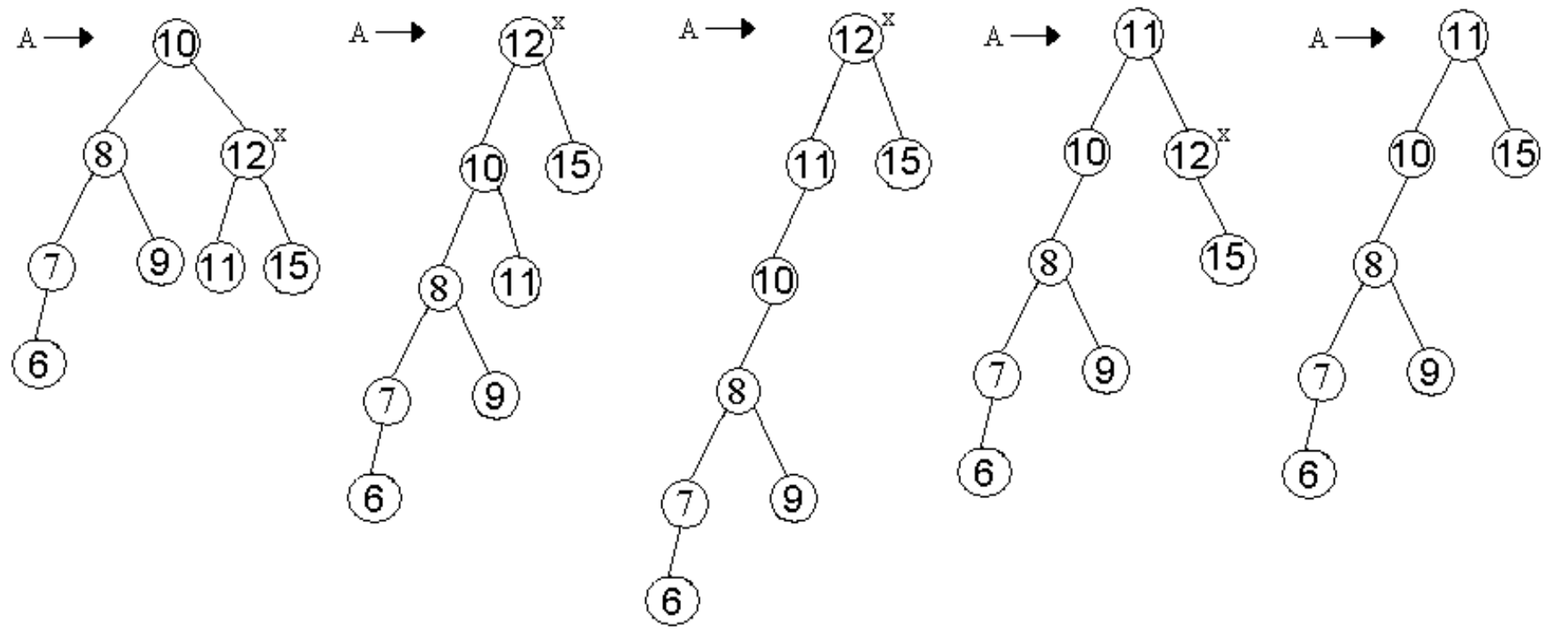
Fim

Exemplo de Remoção(12, A)

1) Pesquisa tipo BST.
2) SPLAY (12,A).
3) SPLAY (12,A') onde A' é a subárvore esquerda do 12.
4) O elemento menor mais próximo de 12 vai para a raiz e sem filho esquerdo, então poderá ser removido.
5) Eliminar o 12 e ligar o pai de X(12) com seu filho direito (15).



Algorítmo de Remoção(A, x)



Início
Se existe o elemento x na árvore A.
Faz SPLAY do elemento x.
Faz SPLAY do elemento x, na sua subárvore esquerda.
Traz o elemento menor mais próximo de x para a raiz.
Remove o x.
Senão
Faz SPLAY do elemento menor mais próximo de x.
Fim

Exemplos de Uso

- **Consultas bancárias**

  Ao fazer uma consulta, o registro de um cliente será levado para o topo da árvore, diminuindo o tempo de acesso para as próximas consultas. Assim, clientes que fazem consultas freqüentemente, terão acesso mais rápido. Os registros de clientes, que não utilizam estes serviços, por sua vez, ficarão nos níveis mais inferiores da árvore.

- **Sistemas de Arquivos**

  Microsoft Windows utiliza na sua indexação de arquivos árvores Splay.

Exemplos de Uso

- **Registro de doentes – Hospital**

  Vai para a raiz no momento da internação e permanece por algum tempo. Vai descendo, caso não seja acessado.

- **Proxy Squid utiliza árvores Splay para manipular o cache interno de páginas Web.**

- **Particularmente útil na implementação de caches.**

Conclusão

- ✓ **Árvores Splay podem ficar desequilibradas, mas garantem uma complexidade O(log n) ao longo do tempo de utilização (*complexidade amortizada*).**
- ✓ **Ajusta a árvore à freqüência de acesso aos dados. Junto à raiz estão os elementos mais usados / mais recentes (mais disponíveis). Os mais inativos ficam mais longe da raiz.**
- ✓ **É importante notar que em um acesso uniforme, a performance da Árvore Splay será considerada pior que em outro tipo de árvore.**
- ✓ **Há estudos empíricos, que defendem que, em muitos casos reais, se verifica que 90% dos acessos são feitos apenas à 10% dos elementos.**

Referências

Bibliografia:
Drozdek, Adam. Estrutura de Dados e Algoritmos em C++
(Árvores Auto-Ajustadas).
Sleator, Daniel D.; Tarjan, Robert D. Self-Adjusting Binary Search Trees.1985.

Sites:
http://www.cs.nyu.edu/algvis/java/SplayTree.html
http://www.cs.cmu.edu/afs/cs.cmu.edu/user/sleator/www/papers/self-adjusting.pdf
http://gdias.artinova.pt/projecto/pt/applets/applet_splay.php

Demonstrações:
http://webpages.ull.es/users/jriera/Docencia/AVL/AVL%20tree%20applet.htm
http://people.ksp.sk/~kuko/bak/index.html