Árvores Splay

INF01203 – Estruturas de Dados
Renata Galante
Viviane P. Moreira

Introdução

- Propostas em 1985 por D. Sleator & R. Tarjan
  - Self-adjusting Binary Search Trees, JACM 32(3) p.652-686
  - http://www.cs.cmu.edu/~sleator/papers/self-adjusting.pdf
- Ao contrário das AVLs, uma árvore splay não usa regras explícitas para forçar seu balanceamento.
- Aplica-se uma operação de mover para a raiz, chamada de *splaying* a cada acesso.

Características

- Uma Árvore Splay é uma ABP
- Os nodos recentemente inseridos/acessados são localizados rapidamente.
- Operações de rotação são aplicadas para mover o nodo acessado para a raiz

Vantagens e Desvantagens

- Os nodos mais freqüentemente acessados ficam mais perto da raiz, o que faz com que eles sejam acessados mais rapidamente.
  - Útil para várias aplicações práticas, incluindo a implementação de caches
- A implementação é mais simples do que a das AVLs e Árvores Rubro-negras.
- Para árvores com acesso uniforme aos nodos, as splay têm um desempenho pior do que outras ABPs.
- O pior caso ocorre quando os nodos da árvore são acessados seqüencialmente em ordem. Isto deixa a árvore completamente desbalanceada.

Quando efetuar o *splaying*

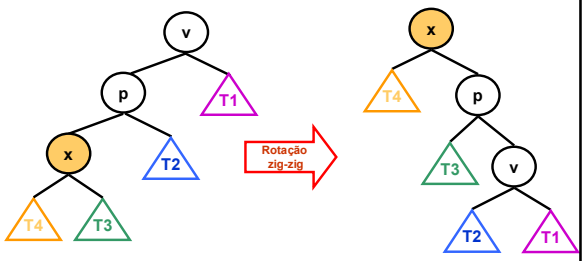
- Quando **pesquisamos** pela chave k, se ela é encontrada em um nodo x, então x é movido para a raiz (*splayed*). Se k não é encontrado, então o pai de um nodo externo no qual a busca termina é movido para a raiz.
- Quando se **insere** a chave k, o novo nodo criado é movido para a raiz logo após a inserção.
- Quando se **remove** uma chave k

A operação splay

- Quando um nodo x é acessado, a operação splay é executada para mover x para a raiz da árvore.
- Para realizar a operação splay, uma seqüência de passos deve ser executada. A cada passo, x é movido para mais perto da raiz.
- Se o nodo x tiver um avô, cada passo depende de dois fatores:
  - Se x é o filho esquerdo ou direito de seu pai p
  - Se p é o **filho esquerdo ou direito** de seu pai v (avô de x)

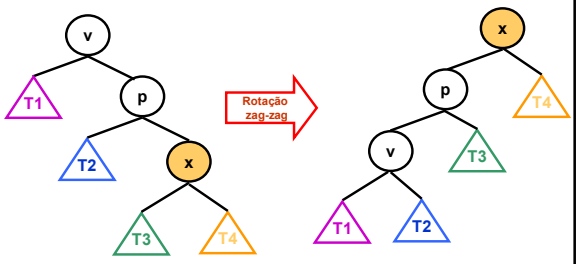
Rotação zig-zig

x < p < v



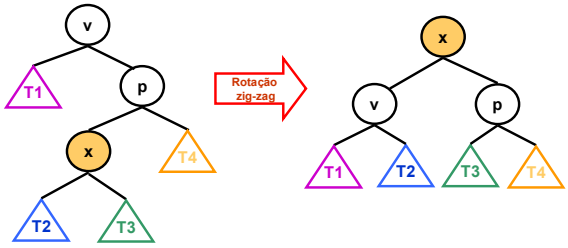
Rotação zag-zag

x > p > v



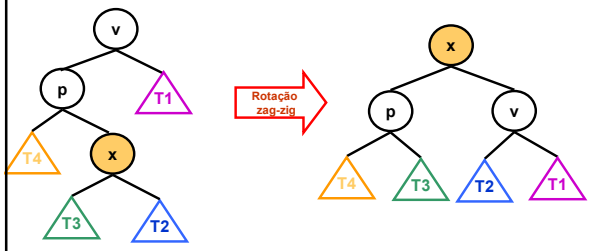
Rotação zig-zag

p > x > v



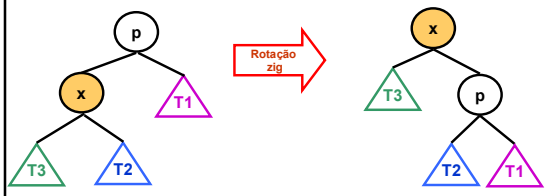
Rotação zag-zig

v > x > p



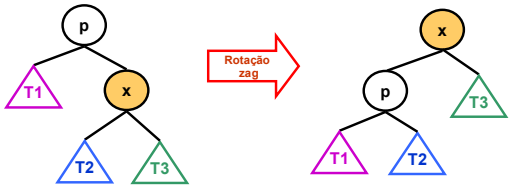
Rotação zig

x é filho da raiz e p > x



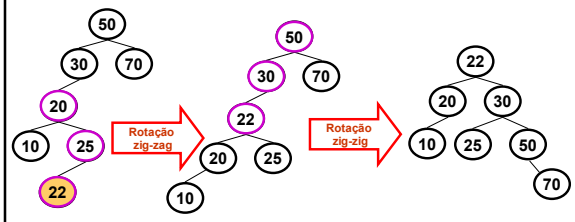
Rotação zag

x é filho da raiz e x > p



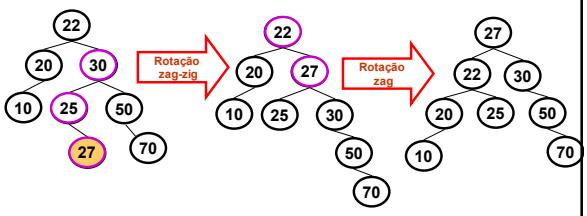
Exemplo 1

Splay 22



Exemplo 2

Splay 27



Operações

Usando splaying, podemos implementar as seguintes operações:

- **`access (i,t)`** – se o item i está na árvore t, retorna um ponteiro para este item, caso contrário retorna nulo.
- **`insert (i,t)`** – insere a chave i na árvore t
- **`delete (i,t)`** – remove o item i da árvore t
- **`join (t1,t2)`** – combina as árvores t_1 e t_2 em uma só árvore contendo todos os itens de ambas as árvores. Esta operação supõe que todos os itens de t_1 sejam menores do que todos os de t_2. Esta operação destrói t_1 e t_2.
- **`split(i,t)`** – constrói e retorna duas árvores t_1 e t_2, onde $t1$ contém todos os itens <= a i e t_2 contém todos os itens >i. Esta operação destrói t.

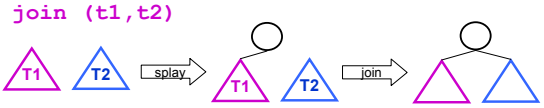
Operação access

`access (i,t)`

- Inicia na raiz da árvore t procurando pelo item i
- Se a busca encontra um nodo x que contenha i, o nodo x é splayed.
- Se a busca não encontra i, o último nodo não nulo da árvore é splayed e um ponteiro nulo é retornado.

Operação join

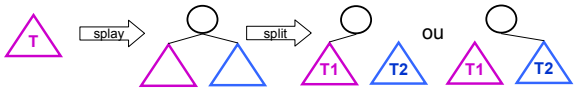
`join (t1,t2)`



- Inicia-se acessando o maior item i em t_1
- Depois do acesso, a raiz de t_1 contém i, e sua sub-árvore direita é nula
- Completa-se o join, fazendo t_2 a sub-árvore direita do nodo que contém i

Operação split

`split(i,t)`

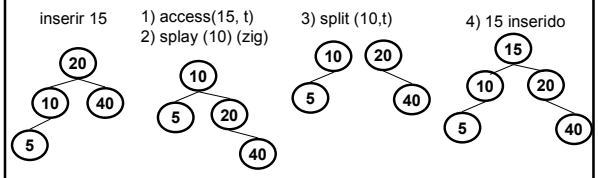


- Inicia-se acessando o i em t, que é movido para a raiz
- Retorna-se as duas árvores formadas pela separação da ligação esquerda ou direita da nova raiz de t

Operação insert

`insert (i,t)`

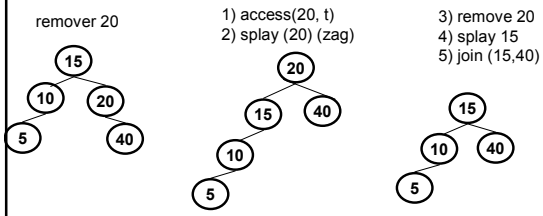
- Para inserir a chave *i* na árvore *t* faz-se um `split(i,t)`
- Após, substitui-se a árvore *t* por uma árvore cuja raiz contenha *i* com sub-árvores t_1 e t_2 retornadas pelo split



Operação delete

`delete (i,t)`

- Para remover a chave *i* na árvore *t* faz-se um `access(i,t)`
- Após, substitui-se *t* pela junção (join) de suas sub-árvores esquerda e direita.



Ferramenta Visual

- http://webpages.ull.es/users/jriera/Docencia/AVL/AVL%20tree%20applet.htm