Binary Search Tree Sort

Las Ratas Recursivas

Universidad Tecnológica Metropolitana

https://github.com/RatasRecursivas

2 de agosto de 2013

Indice

¿Que es?

Introducimos a nuestro querido amigo el BST (**B**inary **S**earch **T**ree):
Preambulo: ¿Que es un nodo?: Consiste en un registro de información, generalmente consta de dos o tres partes:

- Una llave, en base a esta ordenaremos.
- La información como tal.
- Vínculos a otros nodos.

¿Que es?

Introducimos a nuestro querido amigo el BST (**B**inary **S**earch **T**ree):
Preambulo: ¿Que es un nodo?: Consiste en un registro de información, generalmente consta de dos o tres partes:

- Una llave, en base a esta ordenaremos.
- La información como tal.
- Vínculos a otros nodos.

¿Que es?

Introducimos a nuestro querido amigo el BST (**B**inary **S**earch **T**ree):
Preambulo: ¿Que es un nodo?: Consiste en un registro de información, generalmente consta de dos o tres partes:

- Una llave, en base a esta ordenaremos.
- La información como tal.
- Vínculos a otros nodos.

Un BST es una estructura de datos abstracta basada en nodos que posee las siguientes características/reglas:

- El subarbol izquierdo **solo** contiene nodos cuyas llaves son **menores** a la llave de su raiz.
- El subarbol derecho **solo** contiene nodos cuyas llaves son **mayores** a la llave de su raiz.
- Cada subarbol también es un BST.
- No hay llaves duplicadas *.

Un BST es una estructura de datos abstracta basada en nodos que posee las siguientes características/reglas:

- El subarbol izquierdo **solo** contiene nodos cuyas llaves son **menores** a la llave de su raiz.
- El subarbol derecho **solo** contiene nodos cuyas llaves son **mayores** a la llave de su raiz.
- Cada subarbol también es un BST.
- No hay llaves duplicadas *.

Un BST es una estructura de datos abstracta basada en nodos que posee las siguientes características/reglas:

- El subarbol izquierdo **solo** contiene nodos cuyas llaves son **menores** a la llave de su raiz.
- El subarbol derecho **solo** contiene nodos cuyas llaves son **mayores** a la llave de su raiz.
- Cada subarbol también es un BST.
- No hay llaves duplicadas *.

Un BST es una estructura de datos abstracta basada en nodos que posee las siguientes características/reglas:

- El subarbol izquierdo **solo** contiene nodos cuyas llaves son **menores** a la llave de su raiz.
- El subarbol derecho **solo** contiene nodos cuyas llaves son **mayores** a la llave de su raiz.
- Cada subarbol también es un BST.
- No hay llaves duplicadas *.

Con ustedes el arbolito

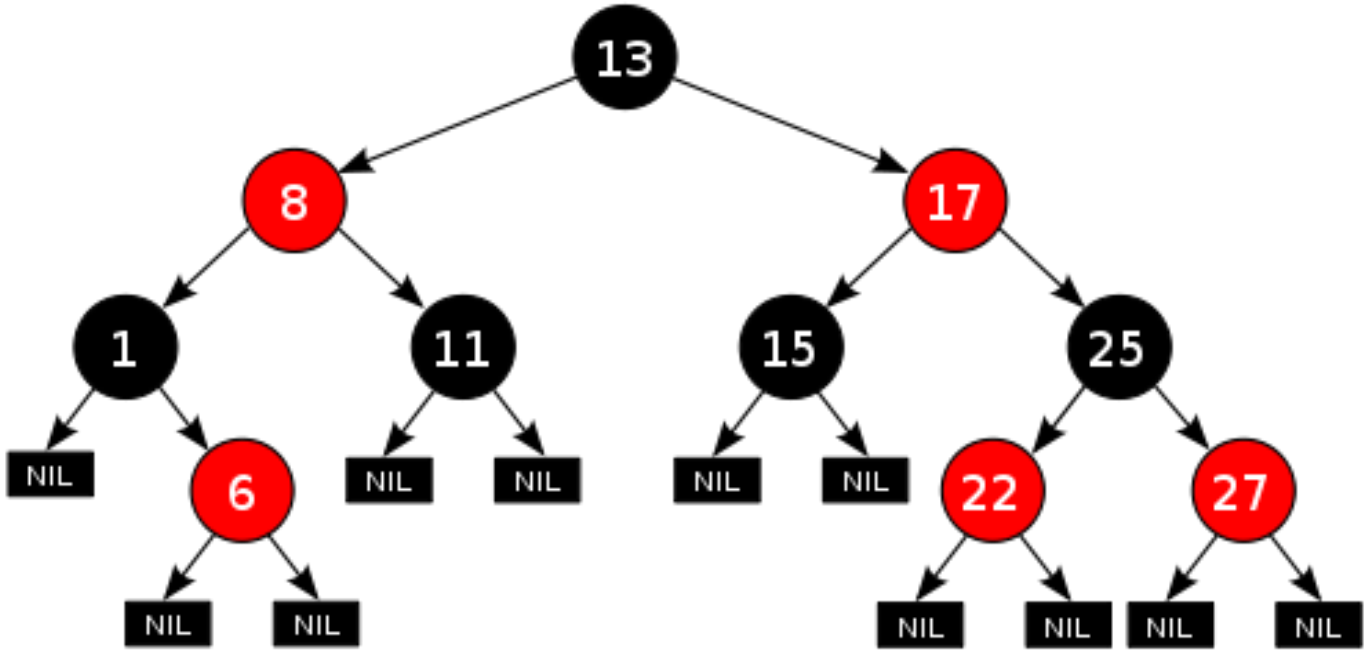

Figura : Red-Black Tree

¿Cuantos BST ven en la imagen?

Nota: Pregunta retórica, pero definitivamente hay mas que uno :-)

Operaciones basicas

Las principales operaciones de un BST son:

- **Inserción**
- **Borrado**: No es necesaria para el sorting.
- **Búsqueda**: No es necesaria para el sorting.
- **Recorridos**:
  - EnOrden
  - PostOrden
  - PreOrden

Operaciones basicas

Las principales operaciones de un BST son:

- **Inserción**
- **Borrado**: No es necesaria para el sorting.
- **Búsqueda**: No es necesaria para el sorting.
- **Recorridos**:
  - EnOrden
  - PostOrden
  - PreOrden

Operaciones basicas

Las principales operaciones de un BST son:

- **Inserción**
- **Borrado**: No es necesaria para el sorting.
- **Búsqueda**: No es necesaria para el sorting.
- **Recorridos**:
  - EnOrden
  - PostOrden
  - PreOrden

Operaciones basicas

Las principales operaciones de un BST son:

- **Inserción**
- **Borrado**: No es necesaria para el sorting.
- **Búsqueda**: No es necesaria para el sorting.
- **Recorridos**:
  - EnOrden
  - PostOrden
  - PreOrden

Operaciones basicas

Las principales operaciones de un BST son:

- **Inserción**
- **Borrado**: No es necesaria para el sorting.
- **Búsqueda**: No es necesaria para el sorting.
- **Recorridos**:
  - EnOrden
  - PostOrden
  - PreOrden

Operaciones basicas

Las principales operaciones de un BST son:

- **Inserción**
- **Borrado**: No es necesaria para el sorting.
- **Búsqueda**: No es necesaria para el sorting.
- **Recorridos**:
  - EnOrden
  - PostOrden
  - PreOrden

Operaciones basicas

Las principales operaciones de un BST son:

- **Inserción**
- **Borrado**: No es necesaria para el sorting.
- **Búsqueda**: No es necesaria para el sorting.
- **Recorridos**:
  - EnOrden
  - PostOrden
  - PreOrden

El peor caso de inserción

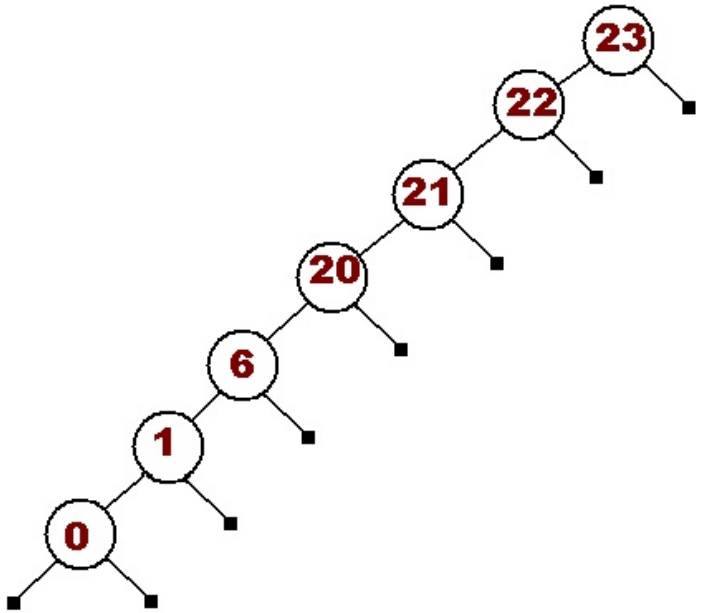

Figura : Worst case evah!

Para insertar -1 tendriamos que pasar por cada nodo!

Ejemplo de recorrido EnOrden

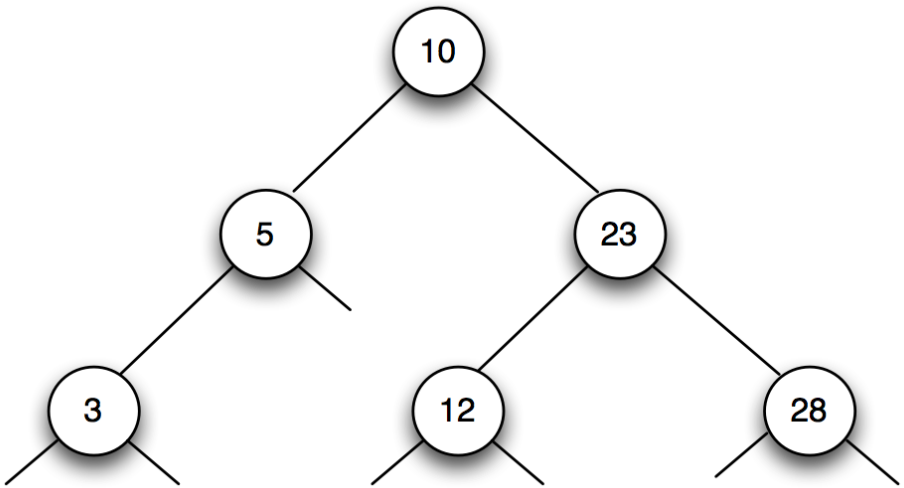

Figura : In-Orden

El recorrido sería: [3-5-10-12-23-28]

Entonces ... ¿Que tiene que ver el bst con el ordenamiento?

Nos aprovechamos de la inserción y el recorrido EnOrden del arbol.

Al insertar en el arbol, los elementos son ordenados segun las reglas de este y al recorrerlo EnOrden son presentados de forma ordenada.

Complejidad y relación con la altura del árbol

En base al árbol podemos postular que su altura esta expresada en la siguiente formula:

$$n <= 2^{(h+1)} - 1 \quad (1)$$

Obteniendo que:

$$h >= \log_2(n) \quad (2)$$

Esta es la característica principal de un árbol, la que permite que las operaciones sean tan rápidas, el único inconveniente es asegurar este mínimo valor para la altura ...

Complejidad y relación con la altura del árbol

Si sumamos las operaciones de inserción y recorrido tendriamos lo siguiente para el mejor caso:

$$O(n) = n \log_2 (n) + n \tag{3}$$

O esto para el peor caso:

$$O(n) = n^2 + n \tag{4}$$

Codigo

```
from bst import BST

def bstsort(l):
    arbol = BST()
    # Generamos nuestra estructura
    # de datos auxiliar

    while l:
        # Vaciamos la lista en el arbol
        item = l.pop()
        arbol.add(item)

    l.extend(arbol.inorder())
    # Ingresamos a la lista los
    # elementos con el recorrido EnOrden
```

Conclusiones

- Se puede mejorar usando arboles autobalanceables, aunque estar asegurando minuciosamente puede añadir un costo considerable.
- No se nota la tendencia logaritmica de las inserciones debido a la complejidad lineal del recorrido.

Conclusiones

- Se puede mejorar usando arboles autobalanceables, aunque estar asegurando minuciosamente puede añadir un costo considerable.
- No se nota la tendencia logaritmica de las inserciones debido a la complejidad lineal del recorrido.