

Informe de Laboratorio 06

Tema: Arbol AVL

INFORMACIÓN BÁSICA

| | | | | | |
|---|----------------------------------|------------------------------|--------|-----------------------|-----|
| ASIGNATURA: | Estructura de Datos y Algoritmos | | | | |
| TÍTULO DEL TRABAJO: | Arbol AVL | | | | |
| NÚMERO DE TRABAJO: | 04 | AÑO LECTIVO: | 2023-A | NRO. SEMESTRE: | III |
| FECHA DE PRESENTACIÓN: | 26/06/23 | HORA DE PRESENTACIÓN: | 23:59 | | |
| INTEGRANTE (s) | | | | NOTA (0-20) | |
| Hidalgo Chinchay, Paulo Andre Betanzos Rosas, Taylor Anthony Villafuerte Ccapira Frank Alexis | | | | | |
| DOCENTE(s): Mg. Edith Giovanna Cano Mamani | | | | | |

Tabla 1: Mi tabla extendida

INTRODUCCIÓN

En este trabajo presentaremos el método de ordenamiento por inserción aplicado sobre una lista enlazada normal y doble. Se tratará además la complejidad de éste método de ordenamiento. Usaremos la librería de GNUPlot para graficar una función que represente el tiempo que tarda el ordenador en ejecutar el ordenamiento por cantidad de datos en el peor escenario posible.

MARCO CONCEPTUAL

aquí ira la MARCO CONCEPTUAL

SOLUCIONES Y PRUEBAS

Métodos search, getMin, getMax

Para el método search se da como argumento un nodo, el cual sirve para redirigirlo a otro método search privado donde se pasan 2 nodos, con la finalidad de compararlos, se el 1er nodo que pasamos como argumento es mayor retorna la búsqueda del nodo derecho, en caso no sea así, retorna la búsqueda del izquierdo

```

133  public NodeAVL<T> search(NodeAVL<T> nodo){
134      return search(nodo,root);
135  }
136  private NodeAVL<T> search(NodeAVL<T> nodo,NodeAVL<T> otro){
137      if (nodo.getData().compareTo(otro.getData())==0)return otro;
138      else if (nodo.getData().compareTo(otro.getData())>0) {//nodo es mayor
139          return search(nodo, (NodeAVL<T>) otro.getRigth());
140      }else return search(nodo, (NodeAVL<T>) otro.getLeft());
141  }
```

Para obtener el máximo solo se retorna la raíz, ya que es un AVL máximo.

```

130  public NodeAVL<T> getMax(){
131      return this.root;
132  }
```

Para obtener el mínimo se usa otro método privado getMin pasando como argumento un nodo, inicialmente la raíz en la cual se valida que la izquierda no sea nula, en caso sea así se retorna el nodo paso por argumento o si no entra recursivamente la función hasta que ya no haya más nodos a la izquierda.

```
123  ✓ public NodeAVL<T> getMin(){  
124      return getMin(root);  
125  }  
126  ✓ private NodeAVL<T> getMin(NodeAVL<T> nodo){  
127      if (nodo.getLeft()==null)return nodo;  
128      return getMin((NodeAVL<T>) nodo.getLeft());  
129  }
```

LECCIONES APRENDIDAS Y CONCLUSIONES

Se aprendió que es el método de ordenamiento por inserción, al igual que como implementarlo para Listas simples y dobles. Por otro lado se a crear los peores casos para este ordenamiento y a como guardar los resultados hasta n casos. Con esto se uso JavaPlot para graficar los resultados, como se muestran en el ejercicio 1 y 2.

REFERENCIAS Y BIBLIOGRAFÍA

<https://www.youtube.com/watch?v=35zTmB9HB6g>
<https://www.youtube.com/watch?v=7wDeHDASoSw>
<https://javaplot.yot.is/example.html>