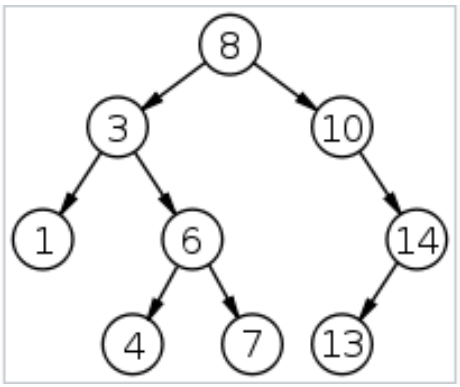
Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignatura: Estructuras de Datos y Algoritmos - Algoritmos I
Segundo Cuatrimestre de 2020

Práctica No. 5 (Árboles)

Parte de esta práctica cuenta con esquemas de programación que pueden accederse a partir del repositorio `https://github.com/EstructurasUNRC/practicas-algoritmos.git`.

1. Dado el árbol binario de la siguiente figura:



   - Marcar el nodo Raíz
   - Cuántos y cuáles son los nodos Hoja?
   - Cuales nodos son ancestros del nodo 4?
   - Cuál es la altura del árbol?
   - Cuántos y cuáles son los nodos del nivel 3?
   - Cual es la profundidad del nodo 6?
   - Imprimir el camino del nodo 8 al nodo 4

2. Acceder al directorio `playground/colecciones/arbol/` del repositorio de la materia y complete la implementación de un árbol binario.

   - En el caso del método aListaInOrder, dar dos implementaciones (recursiva e iterativa) y comparar el tiempo de ejecución.

3. Tenemos un árbol binario t, su recorrido preorden es HDBACFGLJIKNM y en inorden es ABCDFGHIJKLMN, dibujar el árbol, y dar su recorrido postorden.

4. Implemente la clase ABB, de árboles binarios de búsqueda con los métodos insert, delete, search y RepOk.

   a Para cada método calcule su tiempo de ejecución en el peor caso. (En los comentarios de la clase debe incluir un comentario para decir que orden es su algoritmo).

   b Implemente un método para calcular el índice de desbalanceo.

5. implemente la clase Ntree en Java, la clase NTree implementa los árboles n-arios. Defina al menos dos formas de recorrer sus árboles.

6. Usando su clase ABB, implemente el algoritmo TreeSort visto en clases. Compare este algoritmo con el resto de la clase ArraySorter.

7. Implemente la clase Heap con las operaciones insertar, remover, esVacio y repOk. Para cada método calcule su tiempo de ejecución en el peor caso.

   - Utilizando su clase Heap implemente el algoritmo HeapSort (agregarlo en el template de la clase ArraySorter). La idea del algoritmo es construir un heap con los elementos del arreglo a ordenar, para luego utilizar el remover para construir el arreglo resultante. Compare este algoritmo con el resto de la clase.

8. Supongamos que poseemos una implementación de AVL's, empezando desde el árbol vacío, ilustrar como va quedando el AVL cuando se ejecutan las siguientes operaciones:

   - t.insert(10)
   - t.insert(100)
   - t.insert(30)
   - t.insert(80)
   - t.insert(50)
   - t.delete(10)
   - t.insert(60)
   - t.insert(70)
   - t.insert(40)
   - t.delete(80)
   - t.insert(90)
   - t.insert(20)
   - t.delete(30)
   - t.delete(70)

9. Busque en internet una implementación de AVL, utilice la implementación para ejecutar las operaciones del ejercicio 10. Además analice la implementación del cálculo del índice de desbalanceo.

10. Ejecutar esas mismas operaciones en un árbol 2-3.

11. Demuestre por inducción las siguientes propiedades de árboles binarios y defina las funciones correspondientes en Haskell.

    - Para todo árbol t: $alt.t \leq size.t$, en donde alt devuelve la altura y $size$ devuelve su tamaño (definir estas operaciones en Haskell).
    - Para todo árbol t: $espejo.espejo.t = t$, en donde $espejo$ es la función que da vuelta los hijos de un árbol recursivamente (definirla en Haskell).
    - Definir la función $mapTree : (a \to b) \to (Tree\ a) \to (Tree\ b)$, que dado un árbol, aplica una función dada a cada elemento del árbol.