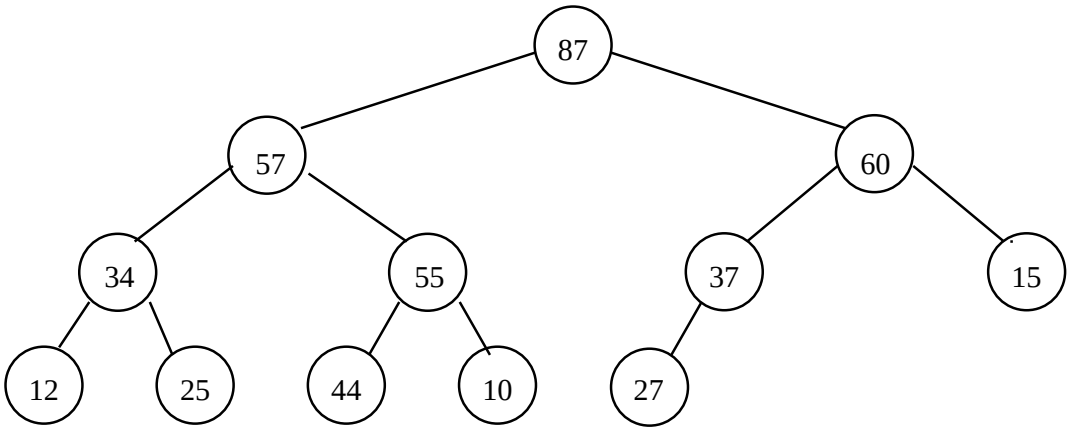
CC30A Algoritmos y Estructuras de Datos
Control 2

Profesor: Patricio Poblete

Junio 4, 1999

1. Para el heap representado por el árbol siguiente:



   a) Dibuje la representación como arreglo respectiva.

   b) Muestre el efecto de insertar "65"tanto en el árbol como en el arreglo.

   c) Muestre el efecto de extraer el máximo desde el heap original.

   d) Si en lugar de tener heaps binarios tuviéramos heaps ternarios, encuentre la fórmula para pasar de padre a hijos y viceversa en el arreglo respectivo.

2. *a*) Comenzando con un árbol 2-3 vacío, inserte los números $1, 2, \ldots, 7$. Muestre paso a paso los árboles resultantes.

   b) Suponiendo que se implementa árboles 2-3 usando una clase `Nodo`, de la cual se derivan 3 clases: `Nodo2` (con campos `info`, `izq` y `der`), `Nodo3` (con campos `info1`, `info2`, `izq`, `medio` y `der`) y `Hoja`, implemente en cada clase un método `recorrer`, que recorra el subárbol respectivo en inorden imprimiendo los campos `info`, de modo que la salida completa consista de todas las llaves en orden ascendente. La idea es que si `a` es un puntero a la raíz del árbol, la llamada inicial sea `a.recorrer()`.

3. *a*) Invente un ejemplo de un árbol con 7 llaves que sea AVL, y otro que no lo sea. En cada caso, anote junto a cada nodo la altura del subárbol respectivo, y marque con asterisco los nodos en donde no se cumple la condición AVL.

   b) Suponga que en un árbol AVL en cada nodo se anota, además, el "peso" del subárbol respectivo (esto es, el número de nodos internos de ese subárbol). Una hoja tiene peso cero. Dibuje las reglas que definen las rotaciones simple y doble, y anote en esos dibujos la manera como se calculan los nuevos pesos de los nodos en función de los antiguos.

4. *a*) Muestre paso a paso cómo se evalúa la siguiente fórmula en notación polaca:

   ```
   5  2  *  4  +  6  3  /  10  2  /  +  /
   ```

   usando un stack.

   b) Suponga que en algún lenguaje de programación extraño una fórmula puede escribirse usando diversos tipos de paréntesis, por ejemplo

   ```
   2/(x-[2/y-z]*{1-(3-a/4)})
   ```

   Describa un algoritmo para que procese de izquierda a derecha los caracteres de una fórmula de ese tipo y verifique si los paréntesis estén bien balanceados. Esto es, que cada cierra paréntesis que aparece calce con un abre paréntesis del mismo tipo. Por ejemplo, la fórmula

   ```
   (3*[2-1/x)]
   ```

   sería incorrecta. El algoritmo debe considerar sólo los paréntesis e ignorar todos los otros caracteres.

   Para que el algoritmo sea general y pueda manejar cualquier tipo de paréntesis (por ejemplo " "), *supongaquesetieneunafunción*(*estática*)`esAbre`(`c`)*queretornaverdaderosiel*

   Es importante notar que el algoritmo *no* puede volver atrás en la fórmula.

   La descripción del algoritmo puede hacerse en "seudo-Java", esto es, complementando Java con partes descritas con palabras.

Tiempo: 2:30 horas Entregar en hojas separadas Sólo una hoja de apuntes