Apellidos, Nombre: ***DNI:***

Examen PED julio 2019
Modalidad 0

Normas:

- Tiempo para efectuar el test: **23 minutos**.
- Una pregunta mal contestada elimina una correcta.
- Las soluciones al examen se dejarán en el campus virtual. Este test vale 4 puntos (sobre un total de 10 de la nota de Teoría).
- **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
- En la **hoja de contestaciones** el verdadero se corresponderá con la **A**, y el falso con la **B**.

	V	**F**		
La especificación algebraica (ecuacional) establece las propiedades de un TAD mediante ecuaciones con variables cuantificadas universalmente, de manera que las propiedades dadas se cumplen para cualquier valor que tomen las variables.	❑	❑	1	V
Dada una especificación de un TAD, sólo existe una implementación posible.	❑	❑	2	F*
En C++, el constructor de copia crea un objeto de una clase determinada a partir de un puntero a un objeto de la misma clase.	❑	❑	3	F
La complejidad espacial del algoritmo de ordenación de vectores de tamaño *n* visto en clase como "intercambio directo (burbuja)" es de O(n).	❑	❑	4	V
La semántica de la operación concatena del tipo cola vista en clase es la siguiente: VAR c, q: cola; x: item; concatena(c, crear_cola ( )) = c concatena(crear_cola ( ), c) = c concatena(c, encolar(q, x)) = encolar(concatena(c, q), x)	❑	❑	5	V*
Grado de un árbol es el número máximo de hijos que pueden tener sus subárboles (si el árbol es n-ario, el grado es n).	❑	❑	6	V*
La sintaxis y la semántica de la operación quita_hojas que actúa sobre un árbol binario y devuelve el árbol binario original sin sus hojas se definen del siguiente modo: quita_hojas(arbin) --> arbin VAR i, d: arbin; x: item; quita_hojas(crea_arbin( )) = crea_arbin( ) quita_hojas(enraizar(i, x, d)) = enraizar(quita_hojas(i), x, quita_hojas(d))	❑	❑	7	F*
Se puede reconstruir un árbol binario cualquiera teniendo sus recorridos en preorden e inorden.	❑	❑	8	V*
Todo árbol binario mínimo es un árbol binario de búsqueda.	❑	❑	9	F*
En un árbol AVL, al realizar una inserción de una sola clave se puede producir como máximo una rotación.	❑	❑	10	V*
Un árbol AVL es un árbol balanceado (equilibrado) con respecto al número de nodos de los subárboles.	❑	❑	11	F*
Dado cualquier ítem en un nodo que no es de tipo hoja de un árbol 2-3, el ítem menor del subárbol derecho de ese ítem estará siempre en un nodo hoja.	❑	❑	12	V
La operación de Rotación en un árbol 2-3 se da cuando el hermano adyacente que hay que consultar (según el criterio especificado) del nodo que se ha quedado sin elementos es del tipo 3-Nodo.	❑	❑	13	V*
Tras el borrado de un elemento que está en una hoja de un árbol 2-3-4, la raíz nunca puede ser un 2-nodo.	❑	❑	14	F
En un árbol 2-3-4 de altura *h*, el máximo número de nodos se da cuando todos los nodos son de tipo 2-nodo.	❑	❑	15	F*
Sea la dispersión cerrada con la siguiente función de redispersión: $h_i(x)=(h_{i-1}(x) + C)$ MOD B. Dos claves sinónimas *x*, *y* tendrán la misma secuencia de intentos, es decir, $h_i(x) = h_i(y)$.	❑	❑	16	V*
Un Heap de *n* elementos representado como un vector de *n* posiciones no tendrá ninguna casilla del vector vacía (sin elementos).	❑	❑	17	V
Dado un digrafo implementado mediante lista de adyacencia en el que sabemos que nunca habrá más de *2n* arcos (siendo *n* el número de vértices), el cálculo de la adyacencia de entrada de un vértice tiene una cota de complejidad temporal O(n).	❑	❑	18	V

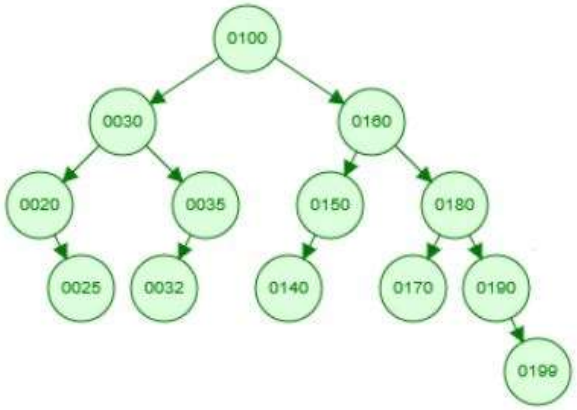
Examen PED julio 2019

Normas:
- Tiempo para efectuar el examen: **2 horas**
- En la cabecera de cada hoja **Y EN ESTE ORDEN** hay que poner: ***APELLIDOS, NOMBRE***.
- Cada pregunta se escribirá en hojas diferentes.
- Se puede escribir el examen con lápiz, siempre que sea legible

1. a) En un árbol AVL inicialmente vacío insertar los siguientes elementos (números Naturales) en el orden especificado:
30, 40, 50, 20, 10, 35, 55 **(0,75 puntos)**

b) En el siguiente árbol AVL borra los siguientes elementos (números Naturales): 20, 30, 170 **(0,75 puntos)**

Nota: indica el factor de equilibrio en cada nodo del árbol y las transformaciones necesarias para reequilibrar el árbol. ***Borrado****: al borrar un nodo con 2 hijos, sustituir por el mayor de la izquierda.*



2. Sea el Grafo DIRIGIDO representado por esta MATRIZ DE ADYACENCIA:

DESTINO

	1	2	3	4	5	6	7	8	9	10	11	12	13
1			1					1					
2				1	1		1		1	1			
3					1	1				1			
4		1				1		1					
5							1						
6							1	1		1			
7													
8	1									1			
9				1									
10			1		1								
11										1		1	1
12		1	1								1		
13													

a) Obtén el RECORRIDO EN PROFUNDIDAD empezando por el Vértice 1 **(0,5 puntos)**
b) Obtén el BOSQUE EXTENDIDO en profundidad con su correspondiente CLASIFICACIÓN DE ARCOS **(0,5 puntos)**
c) Escribe los CICLOS que presenta este Grafo (en forma de LISTA DE ETIQUETAS, separadas por comas) **(0,5 puntos)**

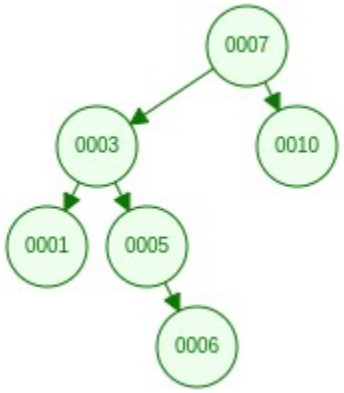
NOTAS:
• Criterio para el recorrido: adyacencia de salida de cada vértice ordenada de MENOR A MAYOR valor.
• Formato para representar la CLASIFICACIÓN DE ARCOS para el apartado **b)** : se debe representar en la matriz de adyacencia, tal y como muestra la figura de abajo:
- Si el arco entre 1 y 3 resulta de "Arbol" (A)
- Si el arco entre 1 y 8 resulta de "Avance" (Av)
- Si el arco entre 2 y 4 resulta de "Retroceso" (R)
- Si el arco entre 2 y 5 resulta de "Cruce" (C)
- etc.

DESTINO

	1	2	3	4	5	6	7	8	9	10	11	12	13
1			A					Av					
2				R	C								

3. a) Utilizando exclusivamente las operaciones constructoras generadoras del tipo árbol, define la sintaxis y la semántica de la operación *EsHoja*, que actúa sobre un árbol y nos indica si éste es una hoja o no. **(0,5 puntos)**

b) Utilizando exclusivamente las operaciones constructoras generadoras del tipo árbol, la operación *EsHoja* definida en el apartado anterior, y las operaciones vistas en clase del tipo conjunto, define la sintaxis y la semántica de la operación *ConjuntoPadresHojas*, que actúa sobre un árbol y devuelve un conjunto con todos los ítems almacenados en nodos que son padres de un nodo hoja. Por ejemplo, al aplicar *ConjuntoPadresHojas* sobre el árbol que se muestra a continuación, se obtiene el conjunto `{7,3,5}`. **(1 punto)**



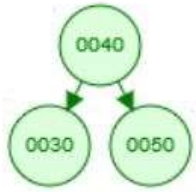
4. a) Realizad el borrado del **MÍNIMO (marcando los intercambios entre claves)** en el montículo doble almacenado en el siguiente array: [raíz vacía, 4, 53, 5, 7, 33, 50, 15, 10, 9, 8, 32, 28, 40, 45, 20, 17, 12, 19, 35, 11, 29, 27, 21, 25, 24, 27, 36, 31] **(0,5 puntos)**

b) Realizad el borrado del **MÁXIMO (marcando los intercambios entre claves)** en el montículo doble almacenado en el siguiente array: [raíz vacía, 4, 53, 5, 7, 33, 50, 15, 10, 9, 8, 32, 28, 40, 45, 20, 17, 12, 19, 35, 11, 27, 29, 21, 25, 24, 27, 36, 16] **(0,5 puntos)**

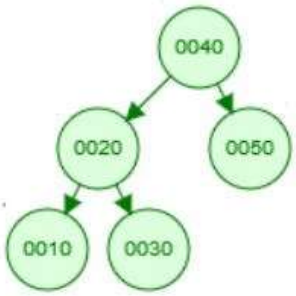
c) **Justificar** las cotas de complejidad temporal en su mejor y peor caso de las operaciones de borrado en un DEAP en función del número de elementos del DEAP. **(0,5 puntos)**

Examen PED julio 2019. Soluciones

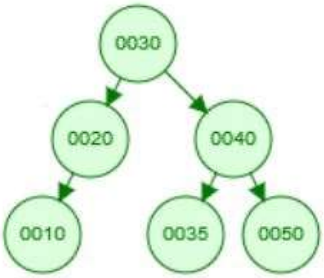
1.a) Inserción 30,40,50 → rotación DD



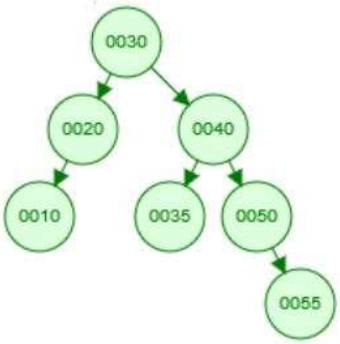
Inserción 20,10 → rotación II



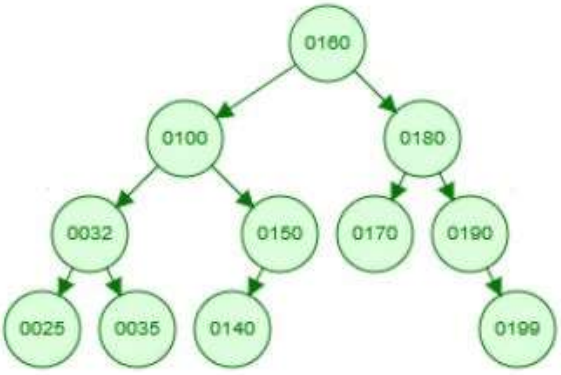
Inserción 35 → rotación ID



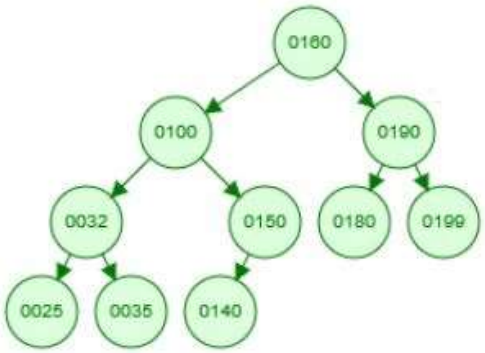
Inserción 55



1. b) Borrado de 20,30 → (20) sustituir por 25 y (30) sustituir por 25 ➔ rotación DI, rotación DD



Borrado 170 → rotación DD

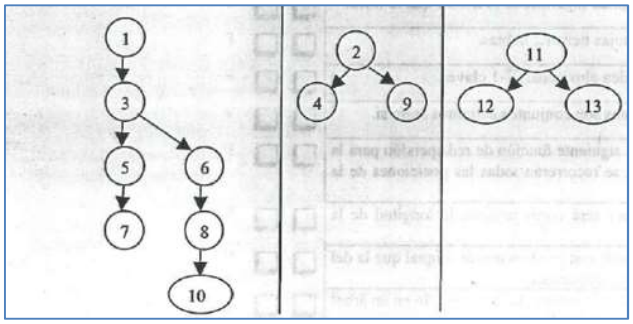


2.

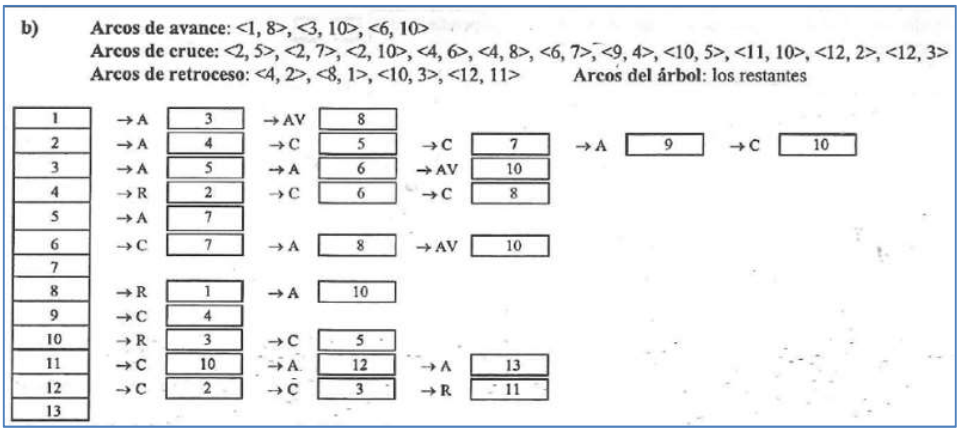
a) Realizar el RECORRIDO EN PROFUNDIDAD empezando por el Vértice 1. ***(0'5 pts.)***

DFS (1): 1, 3, 5, 7, 6, 8, 10

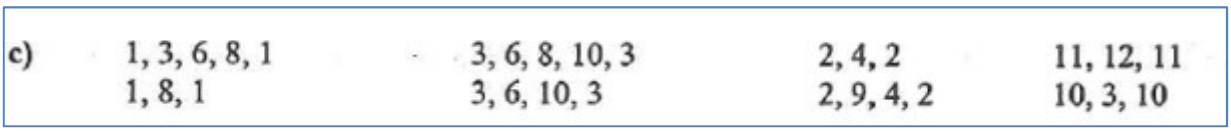
b) Realizar el BOSQUE EXTENDIDO EN PROFUNDIDAD con su correspondiente CLASIFICACIÓN DE ARCOS. ***(0'5 pts.)***



1	2	3	4	5	6	7	8	9	10	11	12	13	
		A					Av						1
			A	C		C		A	C				2
				A	A				Av				3
	R				C		C						4
						A							5
						C	A		Av				6
													7
R									A				8
			C										9
		R		C									10
									C		A	A	11
	C	C								R			12
													13



a) Escribe los CICLOS que presenta este Grafo (en forma de LISTA DE ETIQUETAS, separadas por comas) **(0,5 puntos)**.



3.
a)

sintaxis:
EsHoja: arbin -> bool

semántica:

var x:item; i,d: arbin

EsHoja(crea_arbin()) = FALSO
EsHoja(enraizar(crea_arbin(), x, crea_arbin())) = CIERTO

EsHoja(enraizar(i, x , d)) = FALSO

b)

sintaxis:
ConjuntoPadresHojas: arbin -> conjunto

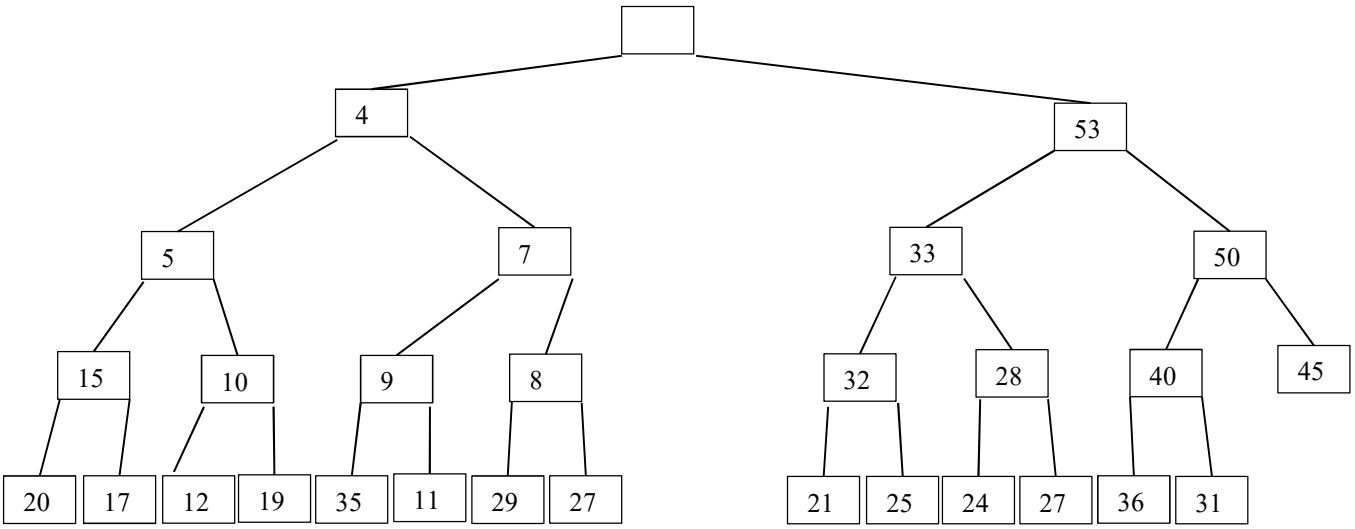
semántica:

var x:item; i,d: arbin

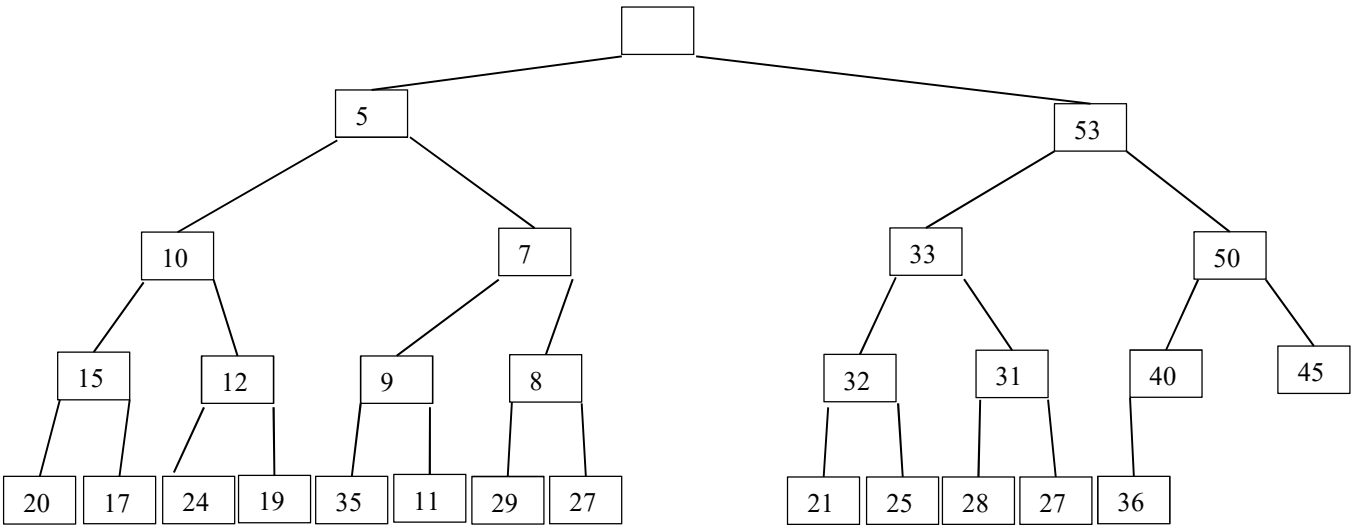
ConjuntoPadresHojas(crea_arbin()) = CrearConjunto()
ConjuntoPadresHojas(enraizar(i, x , d)) =
si EsHoja(i) o EsHoja(d):
    Insertar(Union(ConjuntoPadresHojas(i) , ConjuntoPadresHojas(d)) , x)
si no:
    Union(ConjuntoPadresHojas(i) , ConjuntoPadresHojas(d))

4.

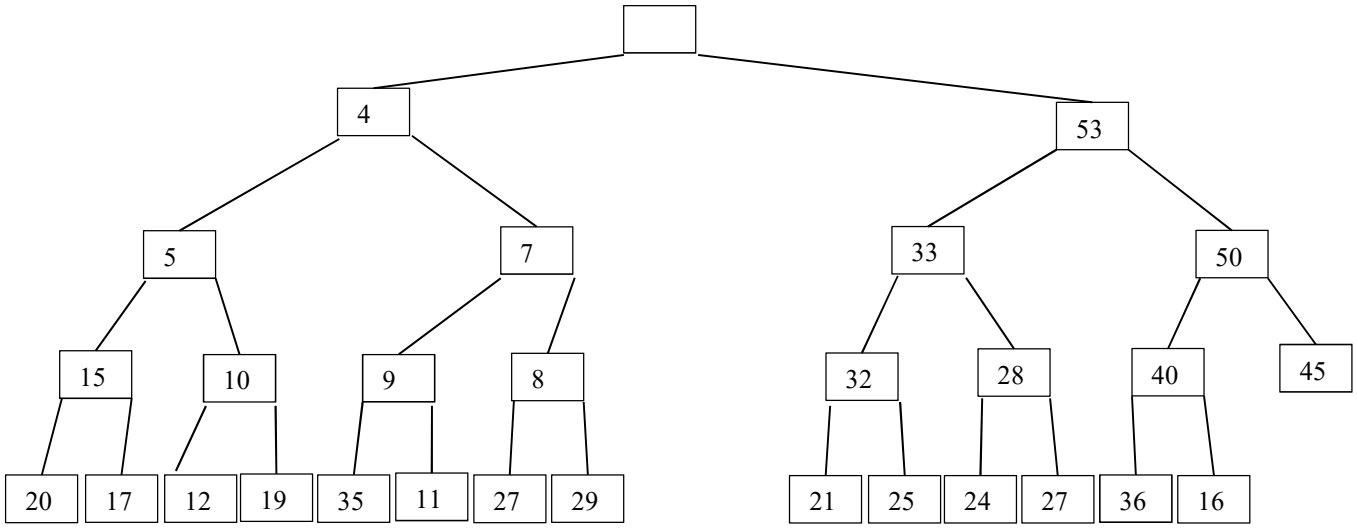
a) DEAP original:



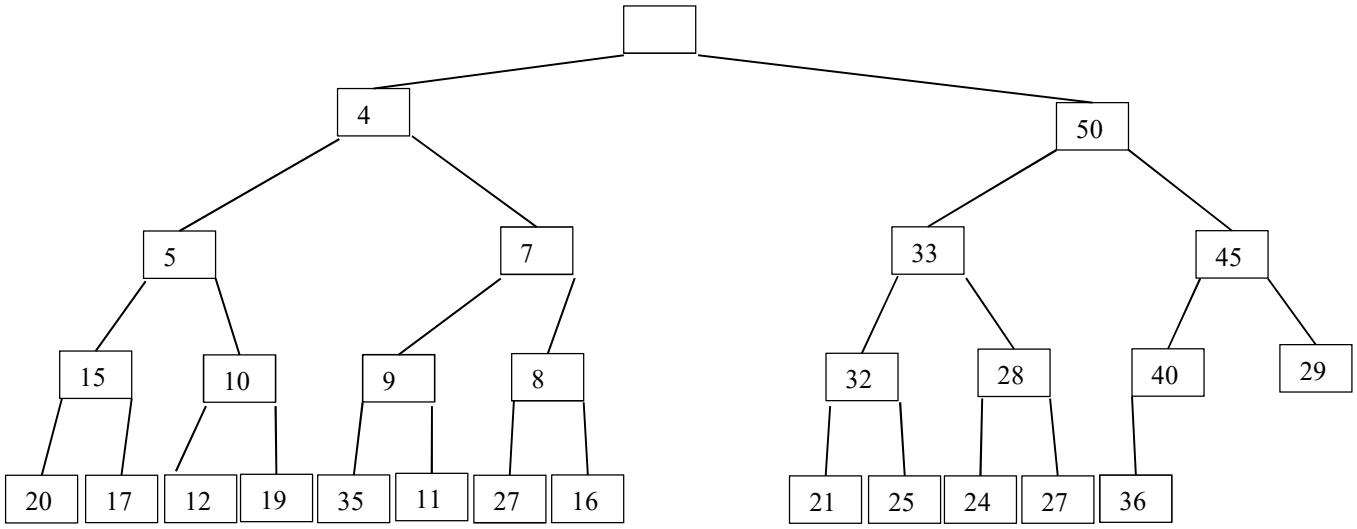
DEAP tras borrado del mínimo:



b) DEAP original:



DEAP tras borrado del máximo:



c) Sería $O(\log_2 n)$ y $\Omega(\log_2 n)$, ya que ha de recorrer la altura del árbol completo correspondiente.