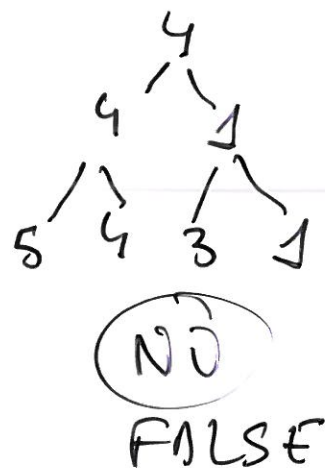
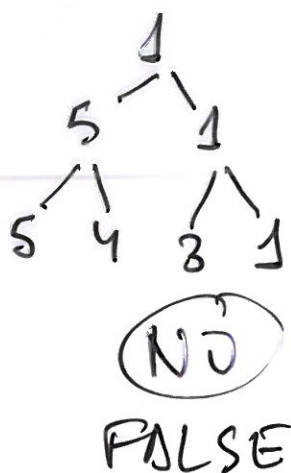
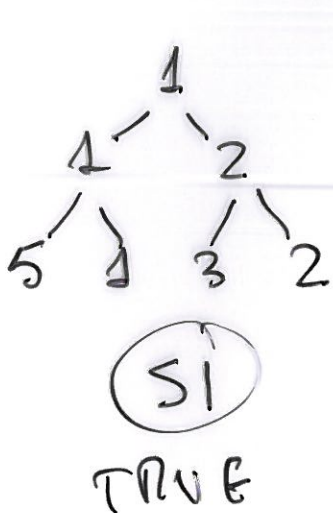


Se define un árbol BWM como un ÁRB en el que la altura de los subárboles izquierdo y derecho en cada nodo puede diferir como máximo en 2.

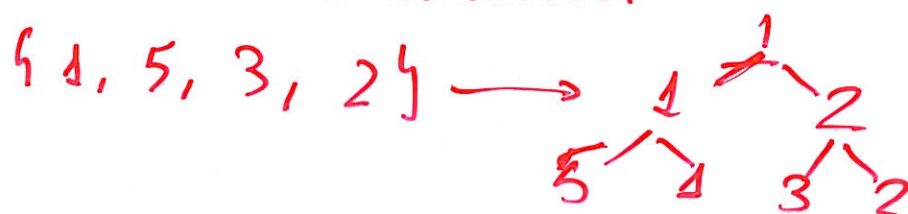
Construir el árbol BWM asociado al conjunto de claves: $\{11, 10, 4, 5, 2, 6, 7, 9, 8\}$

Un árbol de selección es un árbol binario en el que ~~la~~ cada nodo tiene la etiqueta del menor de sus hijos. Diseñar una función que determine si un árbol binario T es un árbol de selección.

bool seleccion (bintree <Inf> &T):

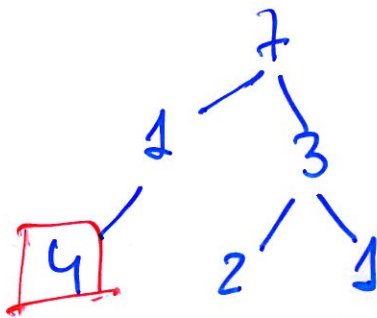


Dada una vector de enteros, construir a partir de ellos un árbol de selección



Se define la trayectoria de una hoja en un árbol binario como la suma del contenido de todos los nodos desde la raíz hasta la hoja multiplicado por el nivel en que se encuentra.

Implementar un procedimiento que, dado un árbol binario ~~devuelva~~ ~~la hoja con mayor trayectoria~~ devuelva la hoja con mayor trayectoria



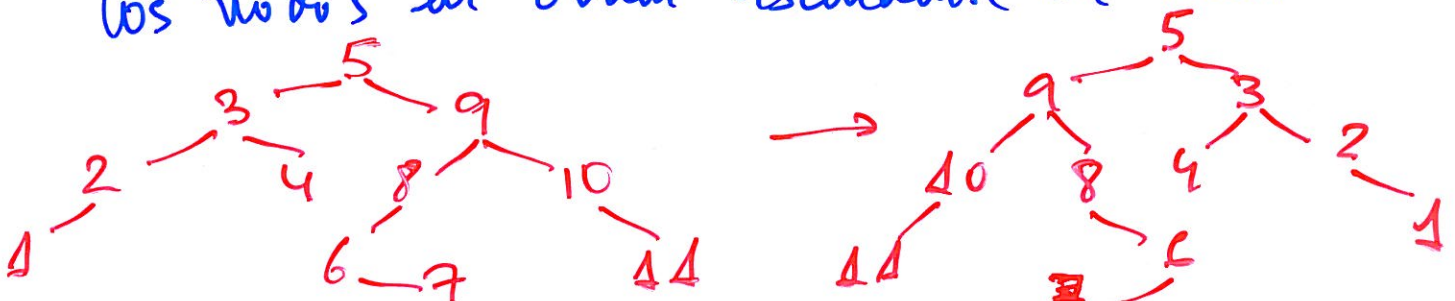
$$\text{Tray}(4) = 4 \times 3 + 1 \times 2 + 7 \times 1 = 21$$

$$\text{Tray}(2) = 2 \times 3 + 3 \times 2 + 7 \times 1 = 19$$

$$\text{Tray}(1) = 1 \times 3 + 3 \times 2 + 7 \times 1 = 16$$

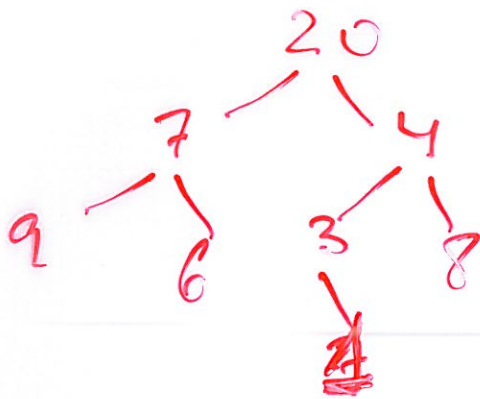
Se denomina "onaedro" ~~al árbol binario~~ ^{ABR} de un ~~árbol binario~~ ^{ABR} T , al árbol binario \hat{T} construido de forma que todo hijo izquierda de T pasa a ser hijo a la derecha de \hat{T} y todo hijo derecha de T pasa a ser hijo izquierda de \hat{T} .

Implementar la función que dado un ABR construya su "onaedro" y diseñar un algoritmo para recorrer el "onaedro" de forma que visite los nodos en orden ascendente de valor.

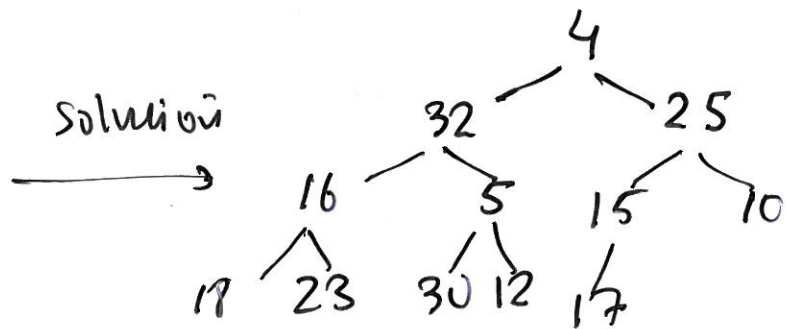
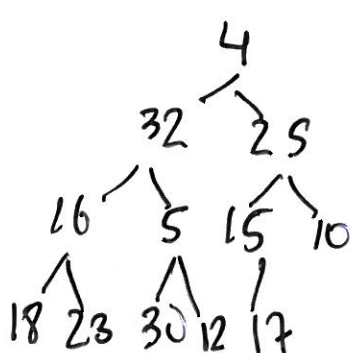
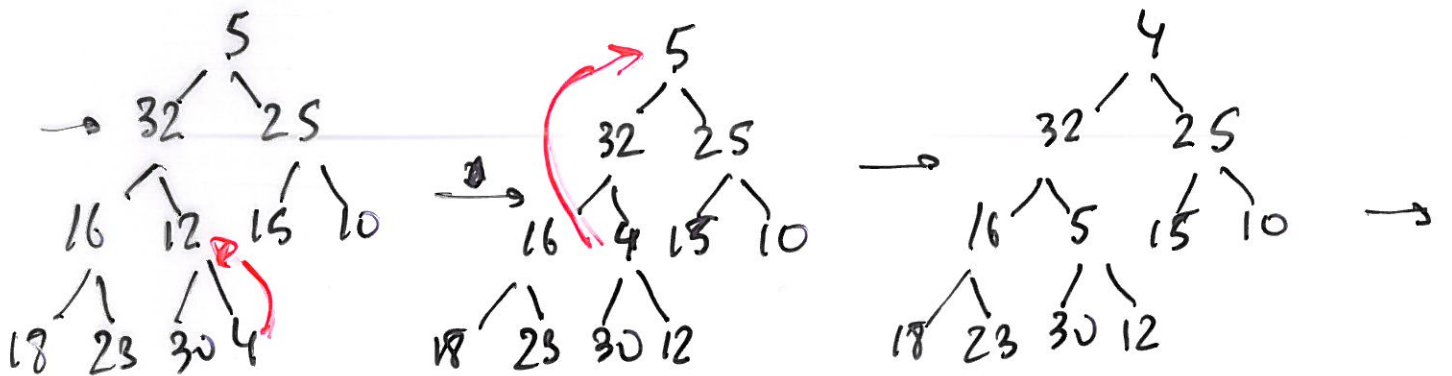
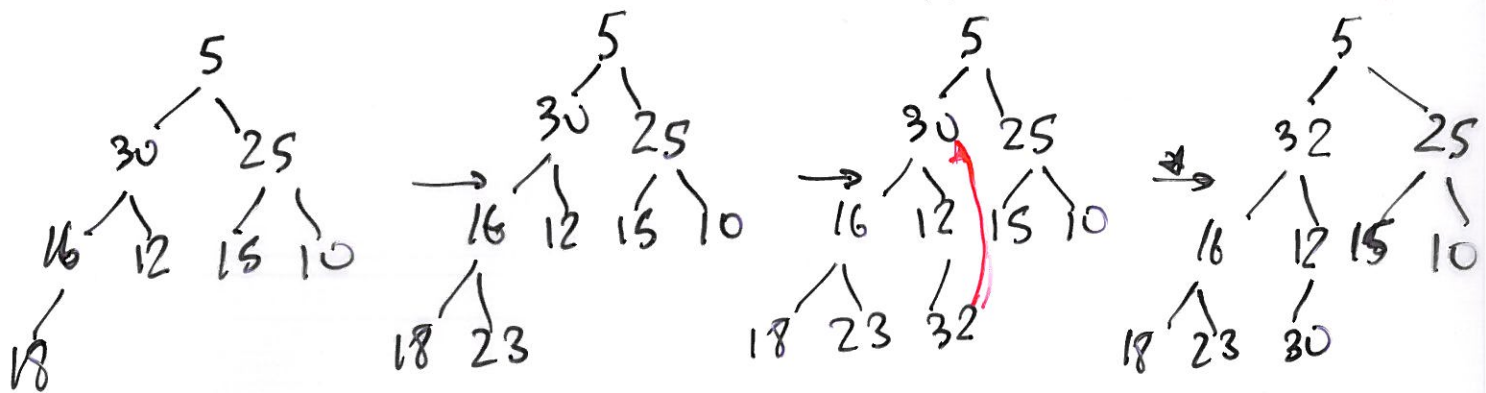
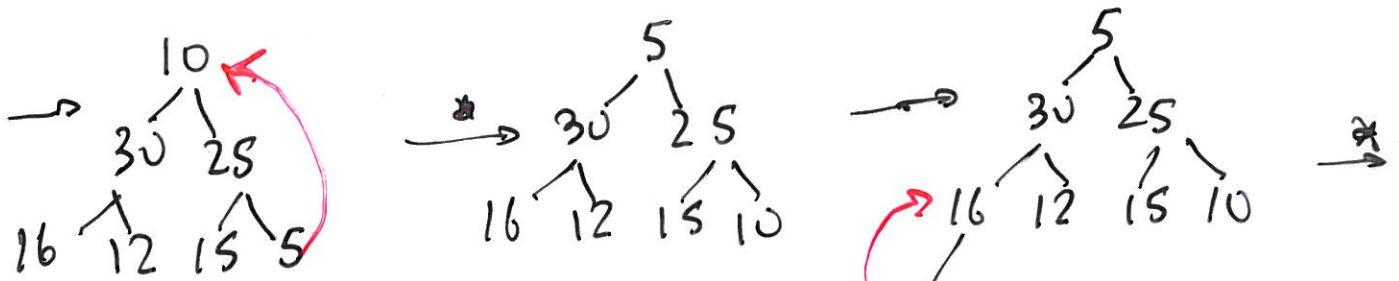
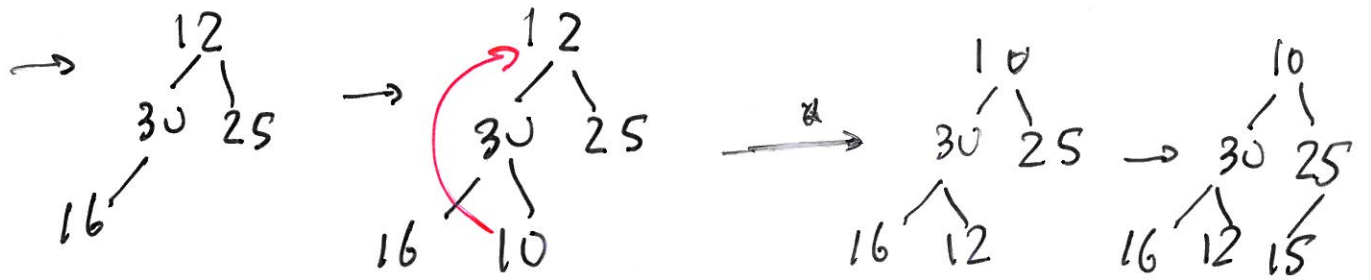
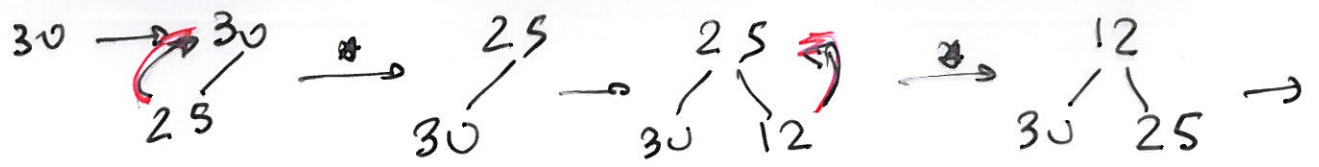


Un "recorrido guiado" sobre un árbol binario comienza listando la raíz para a continuación en cada iteración seleccionar el nodo más pequeño de entre los nodos disponibles en ese momento que no hayan sido listados independientemente de en qué rama se encuentren. Se entiende por nodo disponible aquel cuyo padre ya ha sido procesado (excluyendo la raíz).

Construir una función que permita hacer un recorrido guiado en un árbol. Pueden usarse estructuras auxiliares.



Recorrido guiado: 20, 4, 3, 1, 7, 6, 8, 9



Sea T un árbol binario con n nodos. Se define un k nodo como un nodo v que cumple la condición de que el número de descendientes en el subárbol izquierdo de v difiere del número de descendientes del subárbol derecho en al menos k . Usando el TDA binario construir una función que encuentre los k nodos de T

~~bool~~ ~~cinconodo~~ (const binario <int> :: nodo n) ~~return~~

{
if abs (contar (n.left()) - contar (n.right())

return true;
else return false;
}

int contar (binario <int> :: nodo n)

{
if n = null()
return 0;
else return 1 + contar (n.left())
+ contar (n.right())

}

int recuento (binario <int> <a>)

{ int num = 0;
binario <int> :: preorder_iterator p = a.begin();
while (p != a.end())
if cinconodo (p) num++;
++p } }

Dado un árbol binario de enteros, implementar una función que cuente el número de caminos cuya suma de valores de las etiquetas de los nodos que los componen sea exactamente K .

```
int numeroCamino (bintree <int> & ab, int k,  
                 bintree <int> :: nodo n)
```

```
{  
  if (n.left() == bintree <int> null() &&  
      n.right() == bintree <int> null())  
    if (*n == k) return 1  
    else return 0;
```

```
  else {  
    int contador = 0;  
    if (n.left() != bintree <int> :: null())  
      contador += numeroCamino (ab, k - *n,  
                                n.left());  
    if (n.right() != bintree <int> :: null())  
      contador += numeroCamino (ab, k - *n,  
                                n.right());  
    return contador;  
  }
```

```
}
```

multiset<int> multi_intersection (const multiset<int>
& m1, const multiset<int> m2)

{

multiset<int>::iterator i1 = m1.begin();

multiset<int>::iterator i2 = m2.begin();

while ((i1 != m1.end()) && (i2 != m2.end()))

{ if (*i1 == *i2)

{

result.insert(*i1);

i1++;

i2++;

}

else if (*i1 < *i2)

{

i1++;

}

else i2++;

}

return result;

}