Árboles.

ED Grupo A. Profesor: Isabel Pita.
Tipo A

Nombre del alumno:

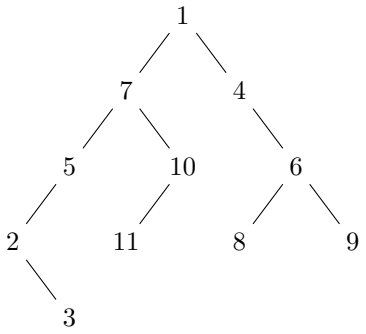
1. (1 punto) ¿Cuántas hojas tiene un árbol binario completo de altura h?. Justifica tu respuesta.

   Respuesta:

   En un árbol completo, todas las hojas se encuentran en el nivel h del árbol. Sabemos que en cada nivel se duplican los nodos del nivel anterior, ya que todos los nodos tienen dos hijos. En el nivel 1 tenemos 2^0 nodos, en el nivel 2 tenemos 2^1 nodos, en el nivel tres tenemos 2^3 nodos Por lo tanto el número de hojas es el número de nodos en el nivel h: 2^{h-1}.

2. (1 punto) El recorrido en postorden de un árbol binario de enteros es: 3 2 5 11 10 7 8 9 6 4 1 y el recorrido en inorden es 2 3 5 7 11 10 1 4 8 6 9. Dibuja el árbol binario al que corresponden estos dos recorridos e indica su recorrido en preorden.

   Respuesta:

   Para obtener el árbol, observamos que la raíz es el último elemento del recorrido en postorden. La posición de la raíz en el recorrido en inorden nos indica los elementos que están en la parte izquierda del árbol y en la parte derecha. Aplicamos recursivamente el mismo proceso para construir los árboles izquierdo y derecho.

   Recorrido en preorden: 1 7 5 2 3 10 11 4 6 8 9

3. (1 punto) La siguiente función tiene coste cuadrático en el número de nodos en el caso peor. Realiza una implementación con coste lineal en el número de nodos, escribe la recurrencia que define su coste e indica su orden de complejidad en el caso de árboles equilibrados y en el caso de árboles degenerados.

```
bool f (bintree<int> const& a) {
  if (a.empty()) return true;
  else {
    bool iz = f(a.left());
    bool dr = f(a.right());
    return numNodos(a.left()) > numNodos(a.right()) && iz && dr);
    }
}
```

   siendo `numNodos` una función que recorre el árbol que recibe como parámetro y devuelve su número de nodos.

Respuesta:

Para evitar el coste cuadrático en el número de nodos de la función anterior debemos evitar la llamada a la función `numNodos`. El número de nodos del árbol no es necesario calcularlo en cada llamada recursiva. Se puede ir calculando al tiempo que se recorre el árbol para comprobar que sus elementos cumplen la propiedad. Para ello la función devolverá dos valores, el primero indica si el árbol de entrada a la función cumple la propiedad y el segundo lleva su número nodos.

```
std::pair<bool, int> f (bintree<int> const& a) {
  if (a.empty()) return {true,0};
  else {
    std::pair<bool, int> iz = f(a.left());
    std::pair<bool, int> dr = f(a.right());
    return {iz.second > dr.second && iz.first && dr.first, iz.second + dr.second + 1};
    }
}
```

La recurrencia que define el coste de esta nueva función es, siendo n el número de nodos del árbol:

$$T(n) = \begin{cases} c_0 & \text{si } a.empty() \\ T(n_i) + T(n_d) + c_1 & \text{si } !a.empty() \text{ y } n_i + n_d = n \end{cases}$$

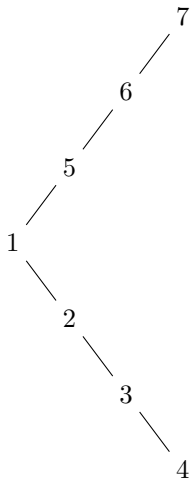
Tanto en el caso del árbol equilibrado como en el caso del árbol degenerado, el coste de esta recurrencia está en $\mathcal{O}(n)$ siendo n el número de nodos del árbol.

4. (1 punto) Dibuja el árbol de búsqueda resultante de insertar sobre un árbol vacío la secuencia

   7 6 5 1 2 3 4 en este orden. Indica el orden en que deberían insertarse los datos para que el árbol resultante fuese completo.

   Respuesta:

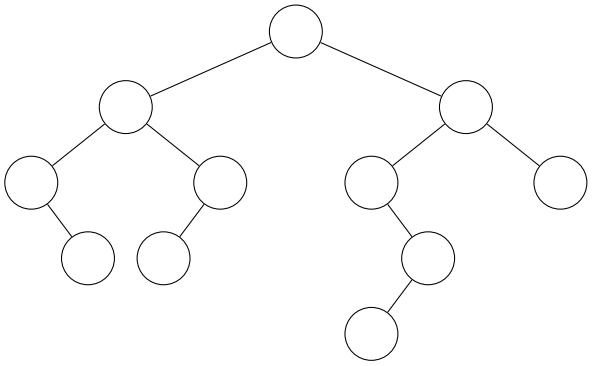
   Al insertar los valores en el orden dado obtenemos:



   Para que el árbol resultante sea completo el primer dato que se inserte debe ser la mediana de la colección de valores a insertar, de forma que la mitad de los valores sean menores que él y la otra mitad mayores. En cada etapa de la construcción del árbol se debe cumplir esta propiedad.

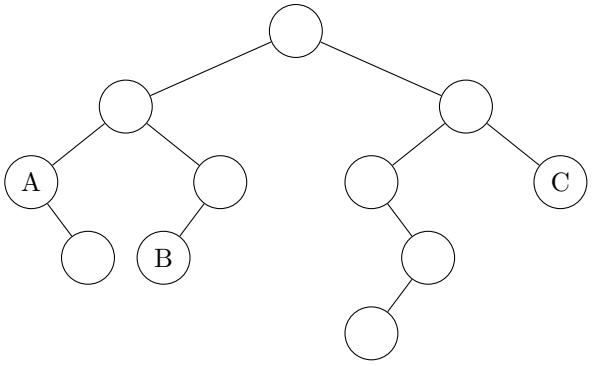
   En este caso concreto una posible forma de añadir los elementos es 4 6 7 5 2 3 1. Existen otras muchas permutaciones de los elementos que dan también lugar a un árbol completo.

5. (1 punto) En el siguiente árbol binario de búsqueda indica en que nodos se encuentran el elemento menor, el cuarto menor elemento y el elemento mayor.

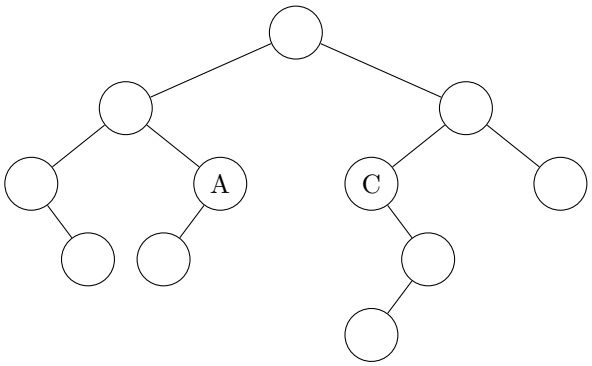


Respuesta:

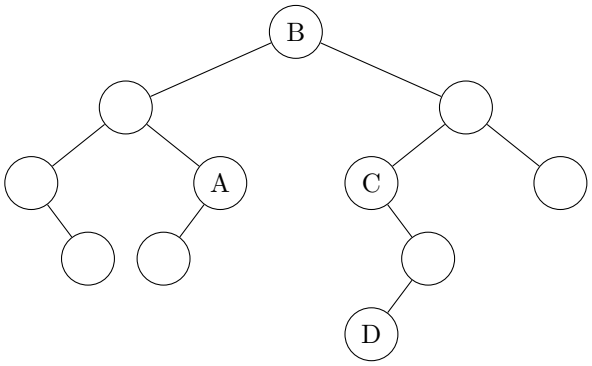
El menor elemento se encuentra en la posición marcada con A. El cuarto nodo se encuentra en la posición marcada con B y el último nodo se encuentra en la posición marcada con C.



6. (1 punto) Dado el siguiente árbol binario de búsqueda indica con el valor B el siguiente en inorden del nodo A y con el valor D el siguiente en inorden del nodo C. Justifica la respuesta.



Respuesta:



El sucesor en inorden del nodo A tiene que ser mayor que el nodo, por lo tanto no puede ser su hijo izquierdo ni su padre, ya que A es el hijo derecho. Cómo A se encuentra en el hijo izquierdo de B, este es el valor inmediatamente superior, ya que todo lo que hay a la derecha de B es mayor que B.

El sucesor en inorden de C estará en su hijo derecho, ya que estos valores son mayores que C. El padre de C es mayor que todos los valores de su parte izquierda y por lo tanto mayor que los valores del hijo derecho de C. El valor D es el más pequeño de los valores mayores que C, ya que es el que se encuentra más a la izquierda.

7. Indica el coste de las siguientes operaciones sobre un árbol binario de búsqueda con n nodos.

tipo	insertar	eliminar	begin	end
Arbol equilibrado				
Árbol degenerado				

Respuesta:

tipo	insertar	eliminar	begin	end
Arbol equilibrado	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$
Árbol degenerado	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$

Dependiendo de la implementación realizada el coste del método `begin()` puede ser constante. Esto se consigue manteniendo en la clase en que se implementa el árbol binario de búsqueda un puntero al nodo inicial del recorrido además del puntero a la raíz. La implementación del método `begin()` de la librería STL tiene conste constante.

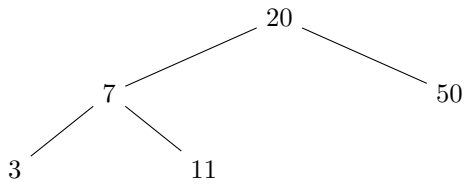
8. (1 puntos) Completa la siguiente frase:

   El recorrido en inorden de un árbol binario de búsqueda obtiene los valores del árbol .....

   Justifica tu respuesta dibujando un árbol binario de búsqueda de 5 nodos e indicando su recorrido en inorden.

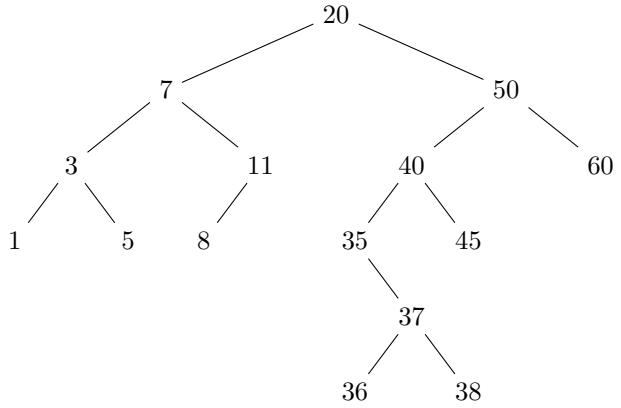
   Respuesta:

   El recorrido en inorden de un árbol binario de búsqueda obtiene los valores del árbol ordenados de menor a mayor según el orden definido al crear el árbol



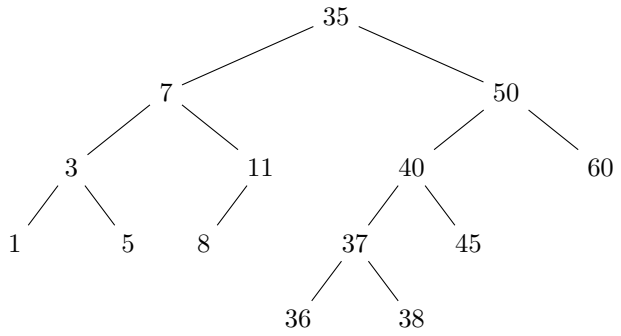
   Su recorrido en inorden es: 3 7 11 20 50

9. (1 punto) Dado el siguiente árbol binario de búsqueda, indica cuál es el resultado de eliminar el nodo 20 según el algoritmo visto en clase.

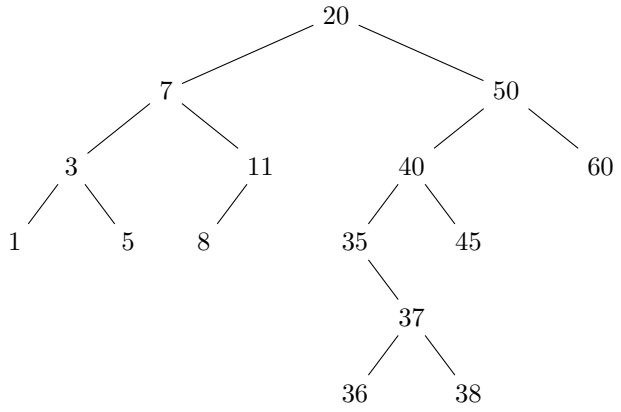


Respuesta:

El nodo 20 tiene 2 hijos, por lo tanto para eliminarlo se localiza el menor valor de su hijo derecho, en este caso el nodo 35 y se intercambia el nodo 20 por este nodo. El nodo 35 solo tiene un hijo, para eliminarlo de su posición actual el hijo izquierdo del nodo 40 pasa a ser el hijo derecho del nodo 35.

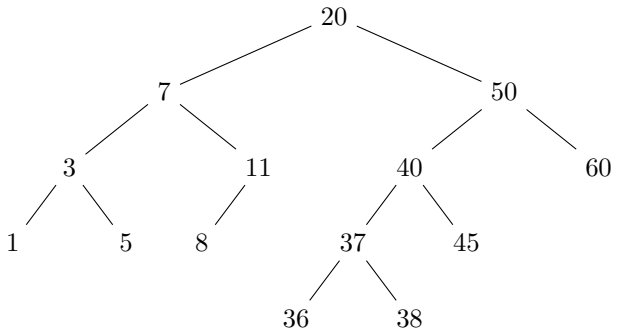


10. (1 punto) Dado el siguiente árbol binario de búsqueda, indica cuál es el resultado de eliminar el nodo 35 según el algoritmo visto en clase.



Respuesta:

El nodo 35 tiene un sólo hijo. El hijo izquierdo del nodo 40 pasa a ser el único hijo del nodo 35.


20
7
50
3
11
40
60
1
5
8
37
45
36
38