Árboles.

ED Grupo A. Profesor: Isabel Pita.
Tipo B

Nombre del alumno:

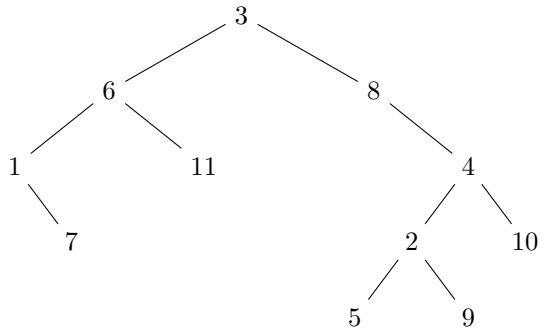
1. (1 punto) ¿Cuántos nodos tiene un árbol binario completo en el nivel k?. Justifica tu respuesta.

   Respuesta:

   En un árbol binario completo todos los niveles tienen su máximo número de nodos. 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 en el nivel k el número de hijos es: 2^{k-1}.

2. (1 punto) El recorrido en postorden de un árbol binario de enteros es: 7 1 11 6 5 9 2 10 4 8 3 y el recorrido en inorden es 1 7 6 11 3 8 5 2 9 4 10. 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: 3 6 1 7 11 8 4 2 5 9 10

3. (1 punto) La siguiente función tiene coste cuadrático en el número de nodos del árbol 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 numPares(a.left()) > numPares(a.right()) && iz && dr);
    }
}
```

   siendo `numPares` una función que recorre el árbol que recibe como parámetro y devuelve el número de valores pares.

   Respuesta:

Para evitar el coste cuadrático en el número de nodos de la función anterior debemos evitar la llamada a la función `numPares`. El número de valores pares 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 el número de valores pares.

```
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 + (a.root()%2==0)?1:0};
    }
}
```

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

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

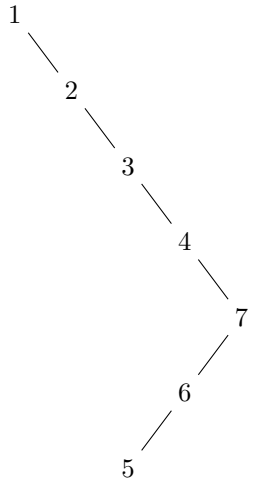
Tanto en el caso del árbol equilibrado como en el caso del árbol degenerado, el coste de esta recurrencia está en $\mathcal{O}(n)$ siendo n el número de nodos del árbol.

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

   1 2 3 4 7 6 5 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.

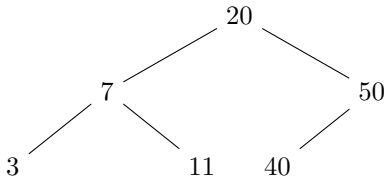
5. (1 puntos) Completa la siguiente frase:

   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 6 nodos e indicando su recorrido en inorden.

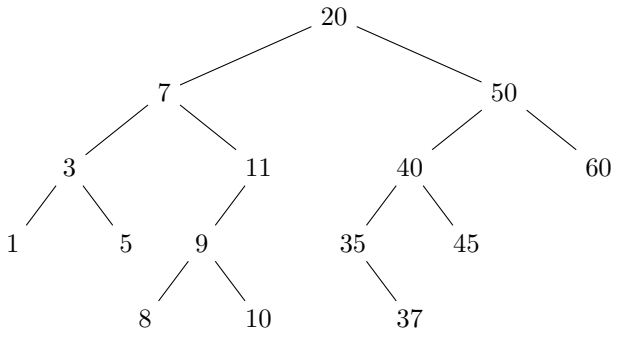
   Respuesta:

   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



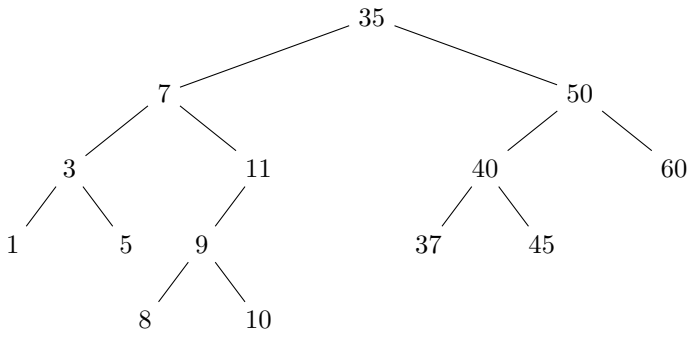
   Su recorrido en inorden es: 3 7 11 20 50

6. (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.

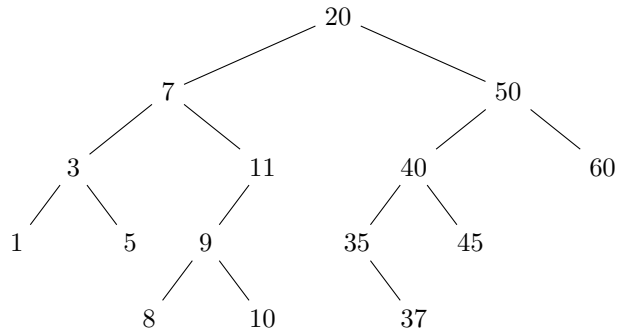


   Respuesta:

   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.

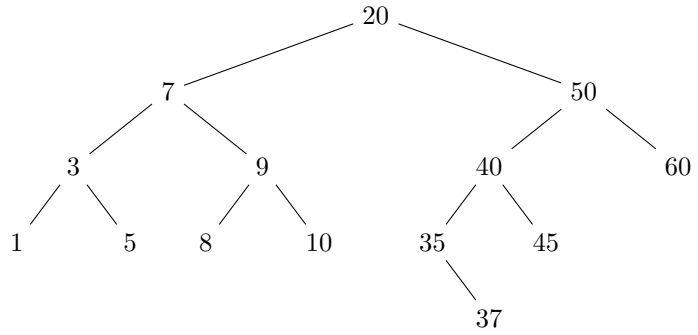


7. (1 punto) Dado el siguiente árbol binario de búsqueda, indica cuál es el resultado de eliminar el nodo 11 según el algoritmo visto en clase.



Respuesta:

El nodo 11 tiene un sólo hijo. El hijo derecho del nodo 7 pasa a ser el único hijo del nodo 11.



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

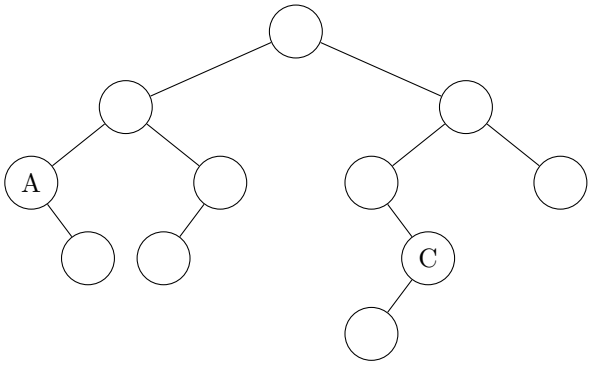
tipo	begin	end	insertar	eliminar
Arbol degenerado				
Árbol equilibrado				

Respuesta:

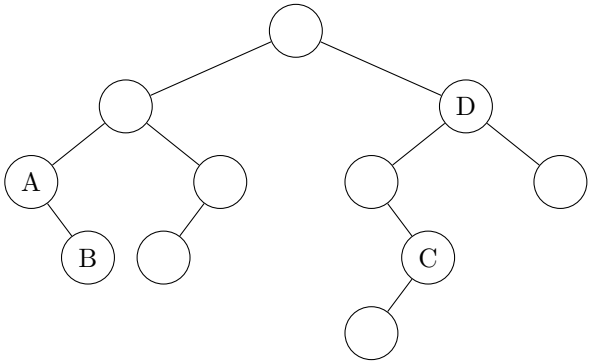
tipo	begin	end	insertar	eliminar
Árbol degenerado	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Arbol equilibrado	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$

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.

9. (1 punto) Dado el siguiente árbol binario de búsqueda indica con una B el siguiente en inorden del nodo A y con una D el siguiente en inorden del nodo C. Justifica la respuesta.



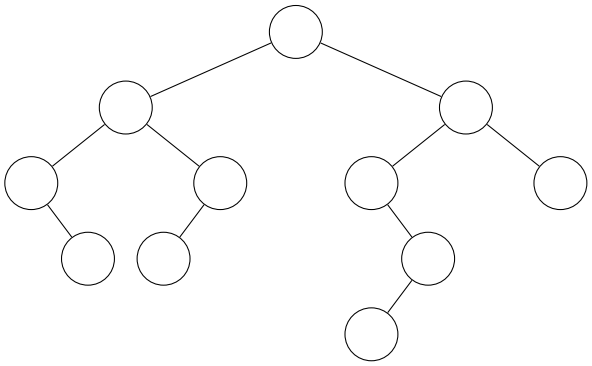
Respuesta:



El sucesor en inorden del nodo A tiene que ser mayor que el valor del nodo, como el nodo A tiene hijo derecho y éste es el menor del hijo derecho, ya que no tiene hijo izquierdo, éste es el nodo buscado

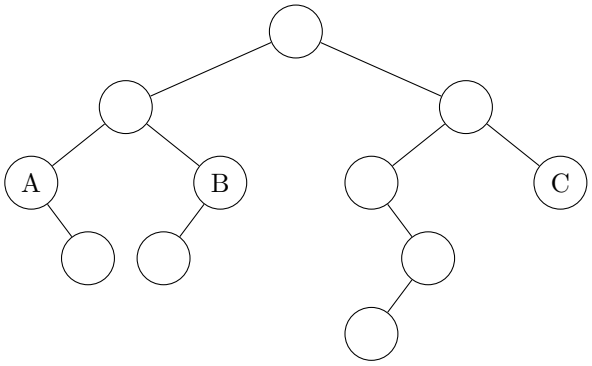
El nodo C no tiene hijo derecho, por lo tanto su sucesor en inorden no puede ser un hijo de C. Buscamos el primer nodo entre sus antepasados tal que C pertenezca a su hijo izquierdo. Este es el nodo que tiene el primer valor mayor que C.

10. (1 punto) En el siguiente árbol binario de búsqueda indica en qué nodos se encuentran el elemento menor, el quinto menor elemento y el elemento mayor. Justifica tu respuesta



Respuesta:

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


A
B
C