Actividad 3_1
Juan Pablo Montoya Estevez

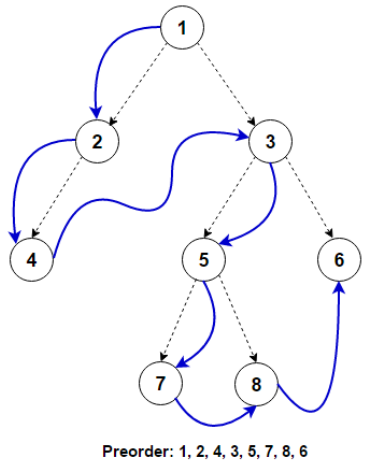
Operaciones con BST

Durante esta actividad se realizaron ciertas operaciones con árboles de búsqueda binaria. Se implementaron los tipos de traversal, inorder, postorder, preorder y levelorder. Además se implementó un algoritmo para obtener la altura, los ancestros y finalmente otro para obtener el nivel en el que está un nodo target.

Primeramente se creó un archivo llamado BST.cpp y BST.hpp en la carpeta tree. En el archivo .hpp se declaró la clase con sus respectivas funciones a utilizar. En el archivo .cpp se implementó está clase y cada una de sus funciones. Las funciones fueron las siguientes:

- BST(): Este era el constructor default, es decir donde no había valor inicial como parámetro.
- BST(int): En este constructor se ingresaba un valor inicial como parámetro que se le ponía a la raíz del bst.
- ~BST(): Este es el destructor.
- insert(BST*): Esa es la función
- preorder(BST*): Esta es la función que imprime el traversal en preorder
- inorder(BST*): Esta función imprime el traversal en inorder
- postorder: Esta función imprime el traversal en postorder.
- levelorder: Esta función imprime el traversal en level order, lo cual usa el algoritmo Breadth first Search.
- height(): retorna la altura del árbol binario.
- Ancestors(): Esto imprime los ancestros de un nodo target
- whichlevelami(): Esto dice en que nivel está un nodo buscado.

Preorder traversal:
En preorder traversal lo que se hace es imprimir los nodos antes de llamar las dos llamadas recursivas. Esto forma un patrón recursivo siguiente.

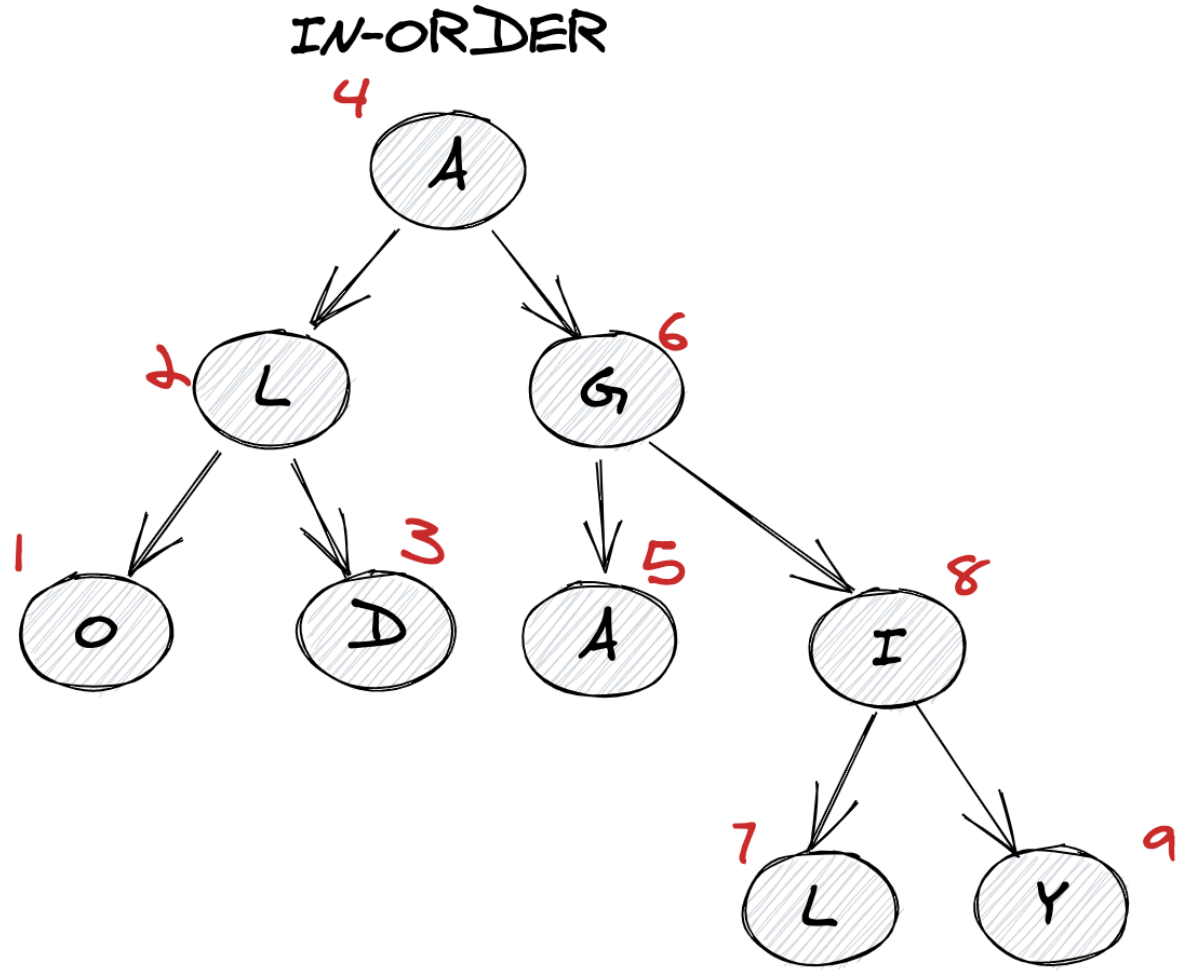


Actividad 3_1
Juan Pablo Montoya Estevez

```
PREORDER :
10
20
20
30
42
394
```

Este es el resultado de correr el algoritmo en un árbol binario con los siguientes números.

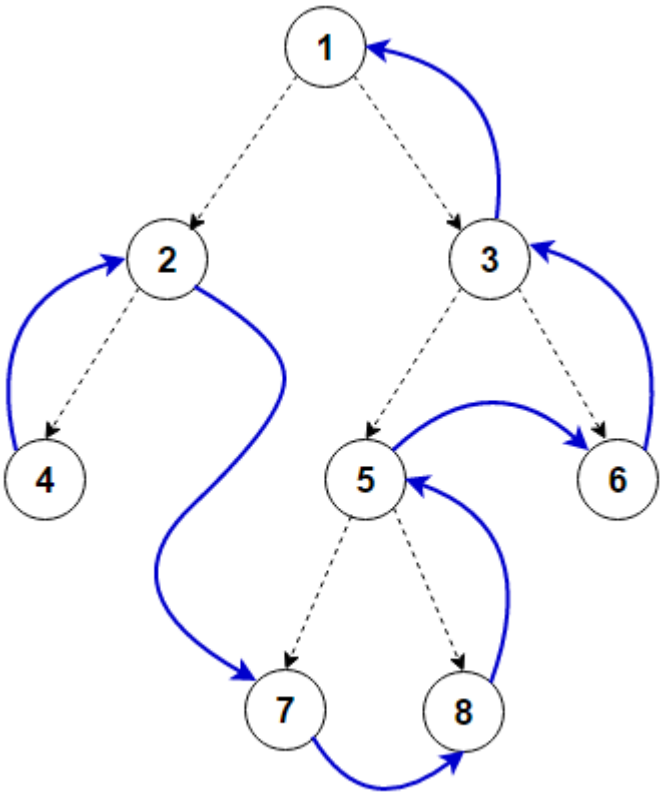
Inorder Traversal:
En este algoritmo se imprime el valor antes de llamar la segunda llamada recursiva. El patrón se ve de la siguiente forma:



Actividad 3_1
Juan Pablo Montoya Estevez

```
INORDER :
10
20
20
30
42
394
```

Esto es lo que se imprime con el algoritmo de inorder traversal.
Postorder:
En este algoritmo se imprime el valor después de llamar ambas llamadas recursivas.



Postorder: 4, 2, 7, 8, 5, 6, 3, 1

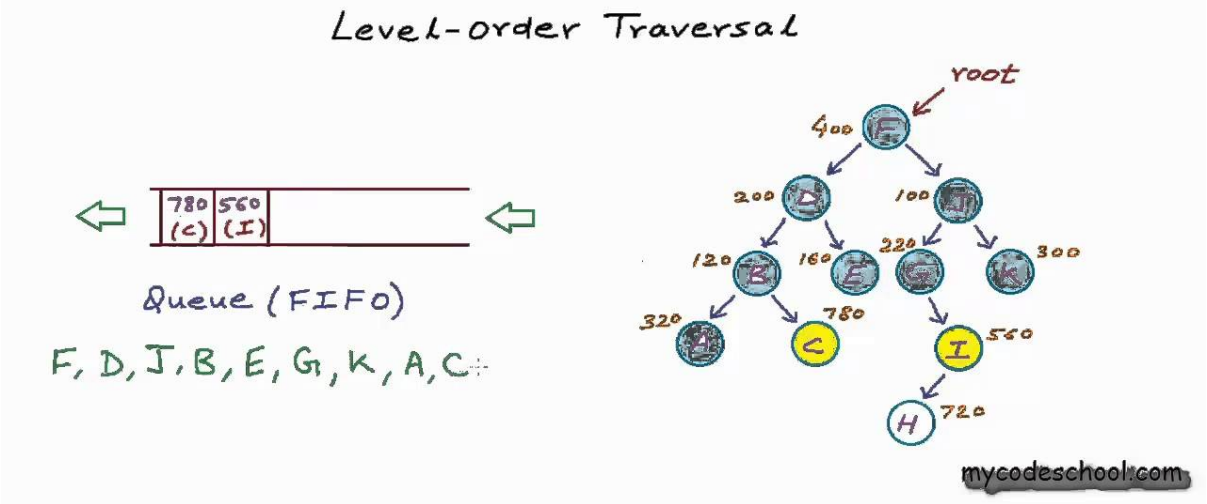
El resultado se ve de la siguiente forma:

Actividad 3_1
Juan Pablo Montoya Estevez

```
POSTORDER :
20
394
42
30
20
10
```

Levelorder: Este algoritmo utiliza breadth first search y se implementa de manera iterativa. Lo que hace es crear un queue, e ir quitando elementos del queue hasta que se acabe.

La visualización sería la siguiente:



Height

Para sacar la altura se parte del concepto de obtener la altura recursivamente y obtener el máximo de cada lado del árbol. De esta manera se saca el camino más largo desde la raíz y con eso se obtiene la altura.

Ancestors:
El algoritmo de imprimir los ancestros es un ejemplo de backtracking, en el cual trackear el estado de las llamadas recursivas una vez que van saliendo del stack. En este caso se llama a la función dentro de un condicional y si la función se cumple se imprime, en el caso de que no se cumplan ambas no se imprime.

Whatlevelami
Este algoritmo toma como ventaja que la estructura de datos se trata de un árbol de búsqueda binaria. Esto significa que se puede utilizar como condición que el valor sea menor o mayor que el valor actual, con lo que se descarta la mitad del árbol. De esta manera se obtiene la altura a la que pertenece, ya que sabes que el valor que busques, esté a la derecha o a la izquierda, será un nivel abajo.

Actividad 3_1
Juan Pablo Montoya Estevez