Árbol binario en un socket utilizando recursividad

Diego Murillo Suárez – 202460-6A - 61

Laura Ramirez Barrera – 202460-6A - 61

Ingrid Karina Quiroga - 202460-6A - 62

Sacha Salazar - 202460-6A - 61

Fundación universitaria del areandina

Modelos de programación II - IS - 202460-6A - 61

Deyvis Morales

Jueves 26 de Septiembre del 2024

Introducción

En el siguiente documento se presentará el entendimiento de cómo se usan 2 estructuras de datos, sockets y un árbol binario en un mismo proyecto, eso sí, cada una teniendo una responsabilidad única, esto debido a que si juntamos varias estructuras de datos en un mismo documento o archivo, no puede ser muy escalable y en el peor de los casos volverse código espagueti.

Árbol binario en un socket utilizando recursividad

Objetivos

El objetivo de este proyecto es usar diferentes estructuras de datos como lo son los sockets y los árboles binarios para entender la concurrencia y la recursividad dentro de un código. En este caso puntual lo haremos con un ejercicio que va a aplicar lo mencionado anteriormente.

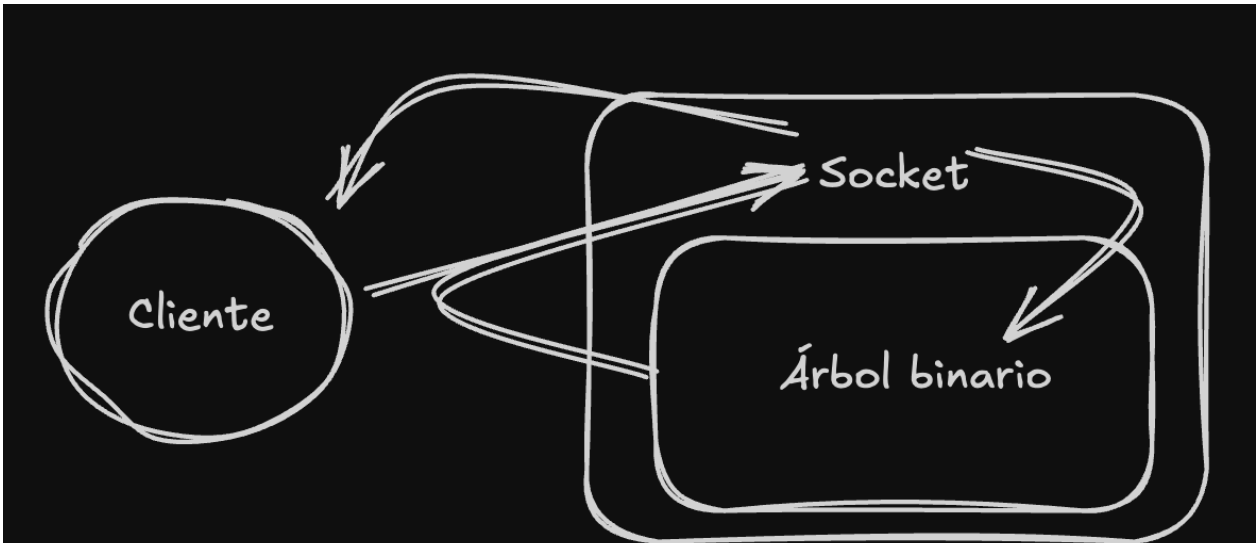
Ejercicio

Construir una aplicación en Python que permite aplicar el tema de árboles binarios y socket, la aplicación consiste en:

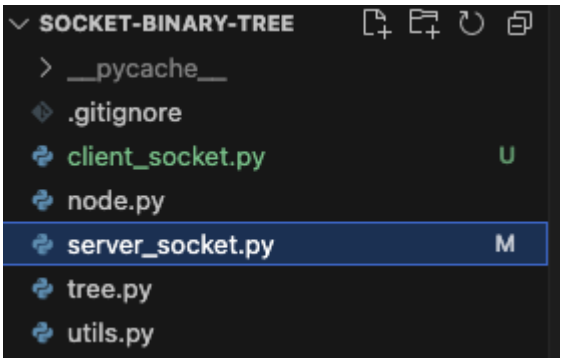
- El desarrollo de la actividad consiste en establecer una comunicación entre el cliente y el servidor que permita por parte del cliente enviar 30 datos numéricos enteros de 2 cifras, uno a uno al servidor y este vaya construyendo un árbol binario con los datos que le envía el cliente a través de la comunicación por medio de socket, tener en cuenta la forma para construir árboles binarios vistos en el eje.

Solución

Para comenzar, les mostraré a continuación un pequeño dibujo sobre la arquitectura que usaremos en el ejercicio. Donde el cliente va a enviar la información hacia el socket, mientras que el socket internamente le enviará la información que debe ingresar al árbol binario. El árbol binario le avisará al server socket cuando se agreguen los datos, y este sí le dará una respuesta al cliente.

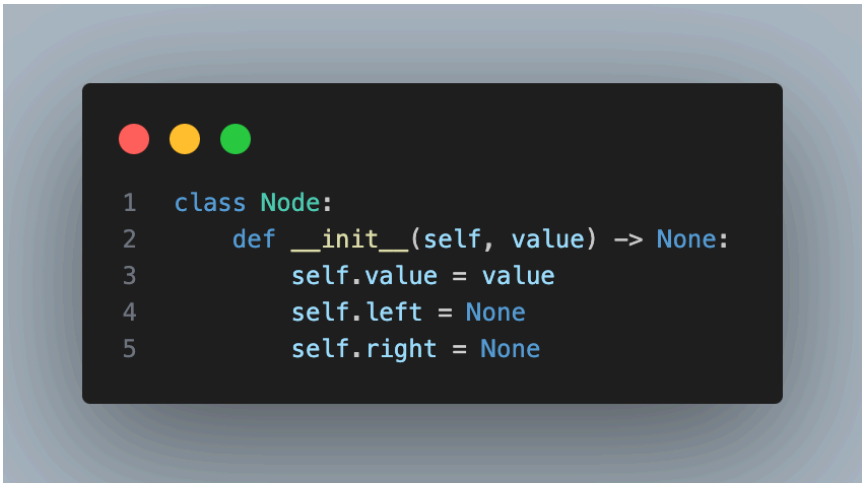


Para comenzar, se mostrará el directorio que utilizaremos con los siguientes archivos:



Empezaremos con el árbol binario:

Para esto necesitaremos una clase nodo la cuál utilizaremos dentro de un árbol binario.



```
1  class Node:
2      def __init__(self, value) -> None:
3          self.value = value
4          self.left = None
5          self.right = None
```

Ya una vez la clase nodo esté creada, debemos crear la clase del árbol binario en el que se haga una instancia del objeto nodo para insertar los datos dentro del árbol binario

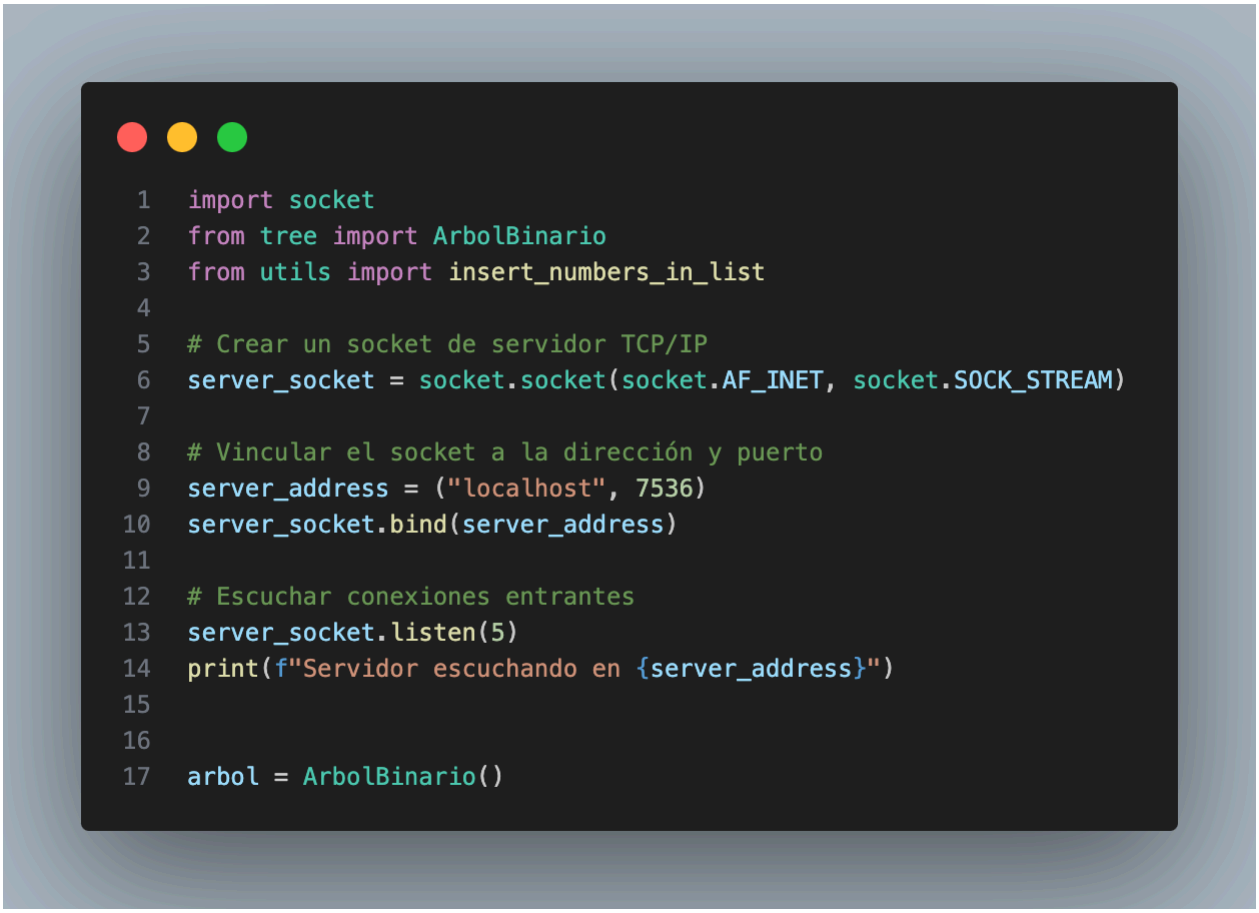
```python
class ArbolBinario:
    def __init__(self):
        self.root = None

    def insertar(self, value):
        if self.root is None:
            self.root = Node(value)
        else:
            self._insertar_recursivo(value, self.root)

    def _insertar_recursivo(self, value, actual_node):
        if value < actual_node.value:
            if actual_node.left is None:
                actual_node.left = Node(value)
            else:
                self._insertar_recursivo(value, actual_node.left)
        elif value > actual_node.value:
            if actual_node.right is None:
                actual_node.right = Node(value)
            else:
                self._insertar_recursivo(value, actual_node.right)
        elif value == actual_node.value:
            print(f"El valor {value} ya está en el árbol.")
```

Como podemos ver acá estamos utilizando la recursividad y concurrencia para insertar datos dentro del árbol. ¿en qué parte se ve la concurrencia? La concurrencia se realiza cuando se llama a sí misma, es por eso mismo que dentro del árbol binario se está utilizando. En este caso, se llama ***insertar_recursivo*** a sí misma cuando el nodo no está vacío.

Luego de esto, debemos prepararnos para que cuando llegue la información al socket, se registren datos al árbol binario.



```python
import socket
from tree import ArbolBinario
from utils import insert_numbers_in_list

# Crear un socket de servidor TCP/IP
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Vincular el socket a la dirección y puerto
server_address = ("localhost", 7536)
server_socket.bind(server_address)

# Escuchar conexiones entrantes
server_socket.listen(5)
print(f"Servidor escuchando en {server_address}")


arbol = ArbolBinario()
```

En este caso, como podemos ver, el socket internamente va a recibir datos del cliente. Luego de ello lo que va a suceder es que dentro de un bucle infinito while insertaremos los datos dentro del árbol para que se ingresen y se ordenen.

```
# Esperar a que un cliente se conecte
        print("Esperando conexión de un cliente...")
        connection, client_address = server_socket.accept()
        print(f"Conexión establecida con: {client_address}")

        # Recibir los datos en fragmentos
        data = connection.recv(1024)
        if data:
            received_message = data.decode()
            print(f"Recibido: {received_message}")
            # Separar los números usando el delimitador
            numbers_str = received_message.split(",")
        arbol.insertar(value)

        # Responder al cliente
        message_to_send_to_client = f"Mensaje recibido. Se han insertado los datos del árbol binario"
        connection.sendall(message_to_send_to_client.encode())
    arbol.recorrer_inorden()
```

Esto retornará un mensaje al cliente indicando que se registró correctamente los datos al árbol binario y también imprimirá los datos inorden en la consola del servidor.

De lado del cliente, tendremos el siguiente código que se conectará al servidor y enviará los números hacia el servidor:

```python
import socket
import random

def obtener_datos():
    datos = []
    while len(datos) < 30:
        try:
            numero = random.randint(-99,99)
            if 10 <= numero <= 99:
                datos.append(numero)
            else:
                print(f"Error: El número {numero} debe tener 2 cifras.")
            if numero < 0:
                print(f"El número {numero} es un número negativo. Por lo tanto no se pondrá en el arbol binario")

        except ValueError:
            print("Error: Por favor, ingrese un número entero válido.")
    enviar_lista_de_numeros(datos)

def enviar_lista_de_numeros(datos):
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(("localhost", 7536))
    client_socket.send(str(datos).encode())
    client_socket.close()


def main():
    obtener_datos()
```

Discusión y conclusiones

Este ejercicio nos mostró de una manera pequeña y simple cómo funciona la arquitectura cliente servidor, sistemas modernos como el buscador de Google, Amazon lo utiliza para su sistema de recomendaciones, Microsoft también lo utiliza para su navegador llamado Bing y finalmente Facebook lo utiliza para ver las personas en común que tú tienes con una persona, en todas estas empresas, utilizan árboles binarios. Sin embargo, hay que saber cómo dejar que tu árbol binario no sea tan pesado ni de un lado ni del otro, de lo contrario a nivel de memoria, se va a consumir más de lo esperado, esto debido a que vamos a guardar datos, pero para sus búsquedas si pueden costar más de lo normal.

Referencias

Python Organization. *Random - How To - Sockets programming.* Última actualización de la documentación el 14 de Septiembre del 2024 de https://docs.python.org/es/3/howto/sockets.html.

Python Organization. *Random - Generate pseudo-random numbers.* Última actualización de la documentación el 14 de Septiembre del 2024 de https://docs.python.org/es/3/library/random.html

Roberto Gomez. *Los sockets de unix*. https://cryptomex.org/SlidesProg/Sockets.pdf

Steve Chipman. *How does google docs work?.* Artículo creado el 8 de octubre del 2022 de How does google docs work?.

Diego Murillo. *Socket.* Última actualización el 28 de septiembre del 2024 de https://github.com/DiegoMurillo7536/Binary-Tree-into-Socket