ENTREGA PARCIAL 1

Estudiantes: Alejandra Díaz Navarro – 2240063 *Grupo:* 2

Cristián David Quintana – 2240072

Ana María Fernández – 2240059

1. Planteamiento el contexto de la situación problema para implementar la estructura de datos y sus métodos.

Sistema De Transporte Público - Rutas

En una ciudad en crecimiento, el sistema de transporte público juega un papel fundamental en la movilidad de los ciudadanos. Para optimizar las rutas y mejorar la eficiencia del servicio, la administración del transporte ha decidido implementar un sistema dinámico de gestión de paradas de bus.

Este sistema permite agregar nuevas paradas cuando se detecta una mayor demanda en ciertas zonas y eliminar aquellas que tienen un bajo flujo de pasajeros. Por lo cual, se genera una estructura de datos eficiente para realizar este tipo de operaciones. Las operaciones de esta estructura de datos incluyen poder agregar una nueva parada a la ruta, visualizar cuántas paradas hay en esta ruta, cuáles son y eliminar las paradas que ya se recorrieron.

2. El Scrib del código pertinente con todas las operaciones que realiza la estructura de datos se encuentra adjunto en el presente archivo (.zip).

3. El análisis Big O del código para la situación del sistema de transporte público y sus rutas.

```python
class Nodo:
    def __init__(self, data):
        self.data = data # O(1)
        self.siguiente = None # O(1)

class ListaParadas:
    def __init__(self):
        self.cabeza = None # O(1)

    def vacio(self):
        if self.cabeza is None: # O(1)
            print("No hay ninguna parada") # O(1)
        else:
            print("Se encuentran paradas") # O(1)

    def contar(self):
        contador = 0 # O(1)
        control = self.cabeza # O(1)
        while control != None: # O(n)
            contador += 1 # O(1)
            control = control.siguiente # O(1)
        print(f"Hay un total de {contador} paradas") # O(1)

    def imprimir(self):
        control = self.cabeza # O(1)
        if control == None: # O(1)
            print("No hay paradas en la lista.") # O(1)
            return
        print("Listado de paradas:") # O(1)
        while control != None: # O(n)
            print(f"La parada {control.data}") # O(1)
            control = control.siguiente # O(1)

```

```python
    def agregar(self, dato):
        nuevo_nodo = Nodo(dato) # O(1)
        nuevo_nodo.siguiente = self.cabeza # O(1)
        self.cabeza = nuevo_nodo # O(1)

    def busqueda(self, dato):
        control = self.cabeza # O(1)
        contador = 1  # O(1)
        while control.data == dato or control == None: # O(n)
            contador += 1 # O(1)
            control = control.siguiente # O(1)
        if(control == None): # O(1)
            print("La parada no se encunetra en la lista") # O(1)
        else:
            print(f"La parada se encuentra en la posicion {contador}") # O(1)

    def eliminarParada(self):
        if self.cabeza != None: # O(1)
            print(f"Eliminando la parada: {self.cabeza.data}") # O(1)
            self.cabeza = self.cabeza.siguiente # O(1)
        else:
            print("No hay paradas para eliminar") # O(1)

# # #
# La complejidad general dominante es O(n) ya que en el peor caso, los bucles While tienen que recorrer toda la lista.
# # #
```