Instituto Técnico Superior Córdoba
LAB 1 - Proyecto de Programación II - Sistema de Gestión de Tareas Pendientes
Profesor: BORDONE, Matías
Estudiantes: CONTRERAS, Yamila y FIGUEROA, Ezequiel
Fecha de entrega: 10 de Agosto

INFORME DE ANALISIS DE ALGORITMO

Cálculo del orden del método agregar_tarea

```python
def agregar_tarea(self, descripcion, prioridad, categoria):

        tarea = Tarea(self.id_actual, descripcion, prioridad, categoria) # 1 accion de creacion de intancia de tarea

        if not self.buscar_tarea_descripcion(descripcion): # n acciones porque el metodo recorre la lista enlazada.
            nuevo_nodo = Nodo(tarea)  # 1 accion de asignacion
            self.id_actual += 1  # 1 accion de asignacion
            self.cantidad_tareas +=1 #1 accion de asignacion

            if self.esta_vacia() or tarea.prioridad > self.cabeza.tarea.prioridad: # al ser el camino corto se descarta
                nuevo_nodo.siguiente = self.cabeza
                self.cabeza = nuevo_nodo

            else:
                actual = self.cabeza     # 1 asignacion
                while actual.siguiente is not None and actual.siguiente.tarea.prioridad >= tarea.prioridad: # n acciones
                    actual = actual.siguiente # n * 1 acciones de asignacion

                nuevo_nodo.siguiente = actual.siguiente # 1 accion de asignacion
                actual.siguiente = nuevo_nodo #1 accion de asignacion

            if not tarea.completada:  # 1 accion de condicion
                if tarea.categoria in self.categorias_pendientes: # 1 accion de condicion
                    self.categorias_pendientes[tarea.categoria] += 1 # 1 accion de asignacion
                else:
                    self.categorias_pendientes[tarea.categoria] = 1

            print("Tarea agregada con éxito.")


        else:
            print("No se pudo cargar la tarea. La tarea ya existe")
```

Suma de todas las acciones:
1 + n + 1 + 1 +1 + 1 + n + n*1 + 1 + 1 + 1 +1 +1
10 + n + n + n*1
T(n)= 3n + 10
O(n)= n

Análisis:
El método es de orden lineal ya que cuenta con bucles while

Cálculo del orden del método contar_tareas_pendientes de orden lineal y constante

```python
def contar_tareas_pendientes_lineal(self)->int:
  actual = self.cabeza      # 1 accion de asignacion
  tareas_pendientes = 0     # 1 accion de asignacion
  while actual is not None: # n acciones que dependen de la cantidad de nodos
    if not actual.tarea.completada: # n*1 condiciones
      tareas_pendientes += 1        # n*1 asignaciones
      actual = actual.siguiente     # n*1 asignaciones
  return tareas_pendientes
```

Suma de todas las acciones:
1 + 1 + n + n*1 + n*1 + n*1
T(n)= 4n+2
O(n)= O(n) ---Este método es de orden lineal.

```python
def contar_tareas_pendientes_cte (self) ->int:
  return self.cantidad_tareas # 1 accion de return
```

Suma de todas las acciones:
#T(n)= 1
#O(n)= O(1) --- Este método es de orden constante

Análisis:
El método es de orden constante, ya que se eliminó el bucle while y se agregaron contadores en los metodos agregar_tarea, completar_tarea y eliminar_tarea.
De esta manera logramos que cada vez que creemos una tarea nueva se agrega 1 al valor de la variable self.cantidad_tareas y cuando se completa o elimina una tarea se descuenta 1 del valor de la misma.