Hoja de Trabajo HT-M1-2

Sección 4 - EDA 2025-20

SOLUCIÓN

1. Considere el siguiente programa, el cual contextualiza la pregunta 1:

```python
def new_double_node(element):
    """
    Crea un nodo para una lista doblemente enlazada.
    """
    return {"info": element, "next": None, "prev": None}

def new_double_list():
    """
    Crea una lista doblemente enlazada vacía con nodos centinela.
    """
    header = {"info": None, "next": None, "prev": None}  # Nodo centinela inicial
    trailer = {"info": None, "next": None, "prev": header}  # Nodo centinela final
    header["next"] = trailer

    return {"header": header, "trailer": trailer, "size": 0}

def is_empty(my_list):
    """
    Verifica si la lista doblemente enlazada está vacía.
    """
    return my_list["size"] == 0

def size(my_list):
    """
    Retorna el tamaño de la lista doblemente enlazada.
    """
    return my_list["size"]

def add_last(my_list, element):
    """
    Agrega un elemento al final de la lista doblemente enlazada.
    """
    trailer = my_list["trailer"]
    last = trailer["prev"]
    node = new_double_node(element)

    node["prev"] = last
    node["next"] = trailer
    last["next"] = node
    trailer["prev"] = node

    my_list["size"] += 1

    return my_list
```

2. Considere los siguientes casos para responder a la pregunta 1:

- Caso 1:
Si la lista está vacía, la lista permanece vacía.

- Caso 2:
Si la lista contiene exactamente un elemento, no hay duplicados posibles. La lista no se modifica.

- Caso 3:
Si la lista no tiene elementos consecutivos repetidos, la lista no se modifica.

- Caso 4:
Si algunos elementos aparecen repetidos consecutivamente, se conservan solo una vez, eliminando todas las repeticiones consecutivas.

- Caso 5:
Si la lista contiene únicamente elementos idénticos, debe conservarse una sola ocurrencia.

3. Considere la siguiente rúbrica, la cual se usaría para calificar la pregunta 1:

Restricciones:

- La complejidad espacial debe ser **O(1)**. La función modifica la lista *en sitio*, es decir, modifica directamente la lista que llega como parámetro a la función. No debe usar ninguna estructura de datos adicional (Ej: set, list, dict, tuple).

- La complejidad temporal debe ser **O(n)**.

- Debe usar la implementación de lista enlazada doble que se describe en la página 1.

- La eliminación no afecta el orden relativo original de los elementos. Es decir, el primer nodo de cada bloque de duplicados se conserva y todos los posteriores consecutivos se eliminan.

Pregunta 1*		
Descripción	**Cumple**	**Incumple**
2.5 pts - La implementación cumple con todas las restricciones y considera correctamente el Caso 1. 0 pts si incumple.		
2.5 pts - La implementación cumple con todas las restricciones y considera correctamente el Caso 2. 0 pts si incumple.		
10 pts - La implementación cumple con todas las restricciones y considera correctamente el Caso 3. 0 pts si incumple.		
15 pts - La implementación cumple con todas las restricciones y considera correctamente el Caso 4. 0 pts si incumple.		
20 pts - La implementación cumple con todas las restricciones y considera correctamente el Caso 5. 0 pts si incumple.		
Total sobre 50pts:		

***** La nota mínima posible es 0.0.

Pregunta 1 (50pts). Implemente la función que se describe a continuación.

Nota: A continuación, se provee solo un ejemplo de solución. Hay muchas formas correctas de solucionar la pregunta 1. Al verificar su propia solución, revise como esta cumple con lo requerido en la rúbrica de calificaciones incluida en la página 2.

```
def remove_duplicates(my_list):
    """
    Elimina los nodos duplicados consecutivos en una lista doblemente enlazada ordenada.
    Como la lista está ordenada, todos los duplicados (si alguno) aparecen consecutivos.
    La función no retorna nada.

    :param my_list: Lista doblemente enlazada ordenada.
    :type my_list: dict
    """
    if is_empty(my_list) or my_list["size"] == 1:
        return  # No hay nada que hacer si la lista está vacía o tiene un solo elemento:

    current = my_list["header"]["next"]

    while current["next"] != my_list["trailer"]:
        if current["info"] == current["next"]["info"]:
            # Hay un duplicado consecutivo, eliminarlo:
            duplicate = current["next"]
            current["next"] = duplicate["next"]
            duplicate["next"]["prev"] = current
            my_list["size"] -= 1
        else:
            current = current["next"]  # Avanzar solo si no hay duplicado consecutivo
```