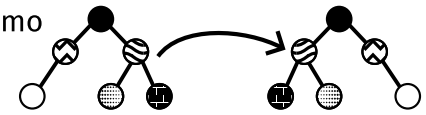
Algoritmos y Estructuras de Datos, Curso Mendez ~ 4to Final, 1er C. 2025 ~ 2025-07-24

Apellido y nombre: _____

Padrón: _____ Modalidad: Completo / Reducido

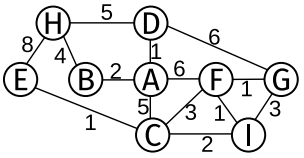
Nota final:				

1) Escriba una función (en **C99** o **Python**) como parte de la implementación de un árbol binario que invierta el arbol. Explique cómo es la estructura del árbol y cómo funciona el algoritmo utilizando gráficos. Muestre un ejemplo del funcionamiento. Justifique la complejidad.



2) Explique qué es un **heap binario** y qué operaciones admite. Utilice **Heapsort** para ordenar de mayor a menor (*in-place*) el siguiente vector: **V = [6,3,8,0,1,2,5,4]**. Muestre cada paso del algoritmo. Justifique cuál sería la diferencia metodológica en caso de un **Heapsort** que no sea *in-place*.

3) Explique para qué sirve y cómo funcionan el algoritmo de **Dijkstra**. Muestre cómo se aplica paso a paso al siguiente grafo desde **E**:



4) Para el grafo del punto anterior, encuentre la cantidad mínima de aristas a recorrer desde **E** al resto de los vértices (sin impotar su peso). Escriba (en **C99** o **Python**) un algoritmo que dado cualquier grafo y un vértice del mismo, pueda determinar esta métrica. Justifique la solución y explique cómo funciona.

5) Explique utilizando diagramas 3 formas diferentes de almacenar grafos. Explique ventajas y desventajas de cada uno y armar una tabla comparativa con la complejidad de agregar/quitar nodos/aristas. Compare también la complejidad espacial de cada forma.

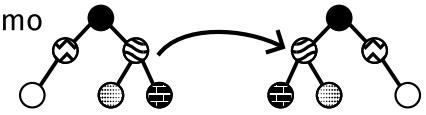
Algoritmos y Estructuras de Datos, Curso Mendez ~ 4to Final, 1er C. 2025 ~ 2025-07-24

Apellido y nombre: _____

Padrón: _____ Modalidad: Completo / Reducido

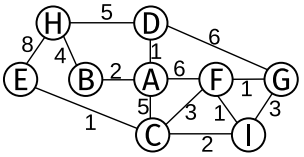
Nota final:				

1) Escriba una función (en **C99** o **Python**) como parte de la implementación de un árbol binario que invierta el arbol. Explique cómo es la estructura del árbol y cómo funciona el algoritmo utilizando gráficos. Muestre un ejemplo del funcionamiento. Justifique la complejidad.



2) Explique qué es un **heap binario** y qué operaciones admite. Utilice **Heapsort** para ordenar de mayor a menor (*in-place*) el siguiente vector: **V = [6,3,8,0,1,2,5,4]**. Muestre cada paso del algoritmo. Justifique cuál sería la diferencia metodológica en caso de un **Heapsort** que no sea *in-place*.

3) Explique para qué sirve y cómo funcionan el algoritmo de **Dijkstra**. Muestre cómo se aplica paso a paso al siguiente grafo desde **E**:



4) Para el grafo del punto anterior, encuentre la cantidad mínima de aristas a recorrer desde **E** al resto de los vértices (sin impotar su peso). Escriba (en **C99** o **Python**) un algoritmo que dado cualquier grafo y un vértice del mismo, pueda determinar esta métrica. Justifique la solución y explique cómo funciona.

5) Explique utilizando diagramas 3 formas diferentes de almacenar grafos. Explique ventajas y desventajas de cada uno y armar una tabla comparativa con la complejidad de agregar/quitar nodos/aristas. Compare también la complejidad espacial de cada forma.