Algoritmos 2, Curso Mendez ~ 2do Final, 2do Cuatrimestre 2024 ~ 2024-12-19

Apellido y nombre: _____

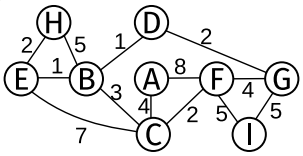
Padrón: _____ Modalidad: Completo / Reducido

Nota final:				

1) El algoritmo **Babilónico** para calcular la raíz cuadrada de **n** consiste en calcular una estimación $\mathbf{n_0 = n/2}$ y luego refinarla iterativamente calculando $\mathbf{n_i = (n_{i-1} + n/n_{i-1})/2}$. Esto se repite hasta que $\mathbf{n_i}$ se acerque a la respuesta correcta lo suficiente. Implementar el algoritmo de manera **recursiva** en **C99** o **Python** y justificar su complejidad.

2) Ordene el siguiente vector de menor a mayor utilizando **QuickSort**. Muestre cada paso del algoritmo. Justifique la complejidad. Explique qué cuidado hay que tener para al aplicar el **Teorema Maestro** y por qué estas consideraciones no son necesarias en el caso de **Mergesort**. **V = [6,4,2,9,8,1,7,3]**

3) Explique qué es el algoritmo de **Kruskal** y para qué sirve. Aplíquelo al siguiente grafo mostrando el resultado de cada paso y el resultado final.



4) Explique qué es un **árbol**. De un ejemplo. Escriba (en **C99** o **Python**) un algoritmo que permita detectar si un grafo es un árbol o no. Muestre de qué forma debe estar representado el grafo para su algoritmo. Explique y muestre cómo funciona.

5) Explique qué es un diccionario. Explique qué características tiene la tabla de hash presentada. Inserte (**+**) y elimine (**-**) los siguientes pares <clave;valor> mostrando el resultado en cada paso: +<F;3>,+<M;1>,+<C;2>,+<H;3>,-<B>,-<A>,+<D;1>;+<B;2>

C;0	A;1		B;3	

Algoritmos 2, Curso Mendez ~ 2do Final, 2do Cuatrimestre 2024 ~ 2024-12-19

Apellido y nombre: _____

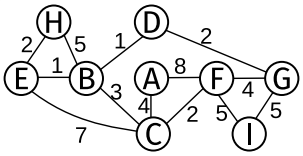
Padrón: _____ Modalidad: Completo / Reducido

Nota final:				

1) El algoritmo **Babilónico** para calcular la raíz cuadrada de **n** consiste en calcular una estimación $\mathbf{n_0 = n/2}$ y luego refinarla iterativamente calculando $\mathbf{n_i = (n_{i-1} + n/n_{i-1})/2}$. Esto se repite hasta que $\mathbf{n_i}$ se acerque a la respuesta correcta lo suficiente. Implementar el algoritmo de manera **recursiva** en **C99** o **Python** y justificar su complejidad.

2) Ordene el siguiente vector de menor a mayor utilizando **QuickSort**. Muestre cada paso del algoritmo. Justifique la complejidad. Explique qué cuidado hay que tener para al aplicar el **Teorema Maestro** y por qué estas consideraciones no son necesarias en el caso de **Mergesort**. **V = [6,4,2,9,8,1,7,3]**

3) Explique qué es el algoritmo de **Kruskal** y para qué sirve. Aplíquelo al siguiente grafo mostrando el resultado de cada paso y el resultado final.



4) Explique qué es un **árbol**. De un ejemplo. Escriba (en **C99** o **Python**) un algoritmo que permita detectar si un grafo es un árbol o no. Muestre de qué forma debe estar representado el grafo para su algoritmo. Explique y muestre cómo funciona.

5) Explique qué es un diccionario. Explique qué características tiene la tabla de hash presentada. Inserte (**+**) y elimine (**-**) los siguientes pares <clave;valor> mostrando el resultado en cada paso: +<F;3>,+<M;1>,+<C;2>,+<H;3>,-<B>,-<A>,+<D;1>;+<B;2>

C;0	A;1		B;3	