2do parcial, 2024, 1C:

1)

¿Cuál es el objetivo de un **índice** de una base de datos?

Rta:

Mejorar el rendimiento en la búsqueda de datos.

2)

¿Cuál de las siguientes estrategias algorítmicas **no** pueden aplicarse a los algoritmos de búsqueda por hashing? Puede haber una o más respuestas correctas.

Rtas:

Paralelismo SIMD.

3)

Relaciona los algoritmos de ordenamiento con situaciones en las que tales algoritmos resultan óptimos.

Rtas:

Insertion sort: Es óptimo cuando los datos vienen aproximadamente preordenados.

Selection sort: Es óptimo cuando debemos minimizar los accesos de escritura.

Pigeonhole sort: Es óptimo cuando el espacio de claves (el rango de valores que pueden tomar las claves) es pequeño.

Merge sort: Es óptimo cuando necesitamos asegurarnos de que el ordenamiento se complete en O (n* log(n))

Quick sort: Es óptimo en situaciones generales, cuando no sabemos nada sobre el conjunto de datos a ordenar.

4)
En las tablas hash, ¿para qué se aplica la operación de **rehashing**? Puede haber una o más respuestas correctas.

Rtas:
Para mejorar la eficiencia de la tabla Hash. Para evitar que el factor de carga se vuelve muy pequeño o demasiado grande.

5)
¿Qué **algoritmo de búsqueda** utiliza una fórmula de aproximación para estimar la posición del elemento deseado?

Rta:
Busquedo por interpolación lineal.
6)
¿Qué **estrategias algorítmicas** son compatibles con el algoritmo merge sort? Puede haber una o más respuestas correctas.

Rtas:
Paralelismo multithreading, Divide-and-Conquer

7)
Supongamos que aplicamos radix sort de base 256 a una lista de 10000 elementos cuyas claves abarcan el rango 1000000 a 9999999. ¿Cuántas veces itera el **bucle principal**?
Mostrar opciones de respuesta

Rta:
3 veces.

8)
En relación a la búsqueda binaria, ¿cuál de las siguientes afirmaciones es **verdadera**?

Rta:
Si el elemento esta en el arreglo, siempre encuentra al elemento.

9)
En relación a las tablas hash, ¿para qué sirve una **política de resolución de colisiones**? Puede haber una o más respuestas correctas.

Rta:
Para resolver el problema de 2 claves cuyo valor hash coincide. Para resolver el problema de asignar un numero de claves M a una tabla hash de tamaño N, con M > N.

10)
¿Cuál es el tipo de datos STL más apropiado para implementar una **tabla hash** en forma explícita?

Rta:
Vector.

11)

Ordenar datos suele **acelerar** muchas operaciones informáticas. ¿En cuáles de las siguientes situaciones es conveniente contar con datos pre-ordenados? Puede haber una o varias respuestas correctas, o incluso ninguna.

Rtas:
Encontrar los elementos de una lista cuyas claves son mayores a cierto valor.
Calcular el valor mínimo y máximo de una lista de elementos.

12)
En relación al ordenamiento, ¿cuáles de las siguientes afirmaciones son **verdaderas**? Puede haber una o varias respuestas correctas, o incluso ninguna.

Rta:
Las funciones de comparación deben ser **determinísticas** (producir un resultado único y predecible a partir de un conjunto determinado de entradas)

13)
¿Sobre qué tipos de datos STL podemos aplicar un **ordenamiento** en forma eficiente? Puede haber una o varias respuestas correctas, o incluso ninguna.

Rtas:
Vector y Deque.

14)
¿Cuál es la estructura de datos más eficiente **en cálculo** para la búsqueda de un elemento en el conjunto de datos: {61, 86, 41, 82, 5, 55, 78, 2, 19, 23, 32, 30, 36, 39, 44, 63, 53, 54, 94, 35, 16, 22, 62, 3, 38, 57, 90, 91, 69, 1, 95, 46, 68, 70, 20, 83, 96, 84, 49, 97, 87, 66, 85, 100, 76, 48, 15, 6, 58, 9}?

Rta:
Una look-up table.

15)
En relación a la **programación dinámica**, ¿cuál de las siguientes afirmaciones es cierta?

Rta:
Es especialmente eficiente para problemas que exhiben la propiedad de superposición de subproblemas.

16)
¿Cuál es la **principal** diferencia entre una look-up table y una tabla hash?

Rta:
La función hash.

17)
Para ordenar el conjunto de datos de 50 elementos {61, 86, 41, 82, 5, 55, 78, 2, 19, 23, 32, 30, 36, 39, 44, 63, 53, 54, 94, 35, 16, 22, 62, 3, 38, 57, 90, 91, 69, 1, 95, 46, 68, 70, 20, 83, 96, 84, 49, 97, 87, 66, 85, 100, 76, 48, 15, 6, 58, 9}, ¿cuál es el algoritmo de ordenamiento más **eficiente**?

Rta:
Pigeonhole sort

18)
¿Qué **algoritmo de búsqueda** divide el espacio de búsqueda por la mitad en cada paso?

Rta:
Busquedo binaria.

19)
¿Cuáles son las **principales ventajas** de los tries en comparación con otras estructuras de datos que operan sobre cadenas de texto?

Rtas:
Los tries permiten encontrar eficientemente todas las palabras que comienzan con cierto prefijo y por tanto son útiles para implementar texto predictivo. Los tries son eficientes para realizar búsqueda de texto completo (determinar los documentos que contengan cierta palabra).

20)
¿Qué **algoritmo de búsqueda** es óptimo para encontrar un elemento en un arreglo ordenado con claves uniformemente distribuidas, con espacio de claves muy grande?

Rta:
Busqueda por interpolación lineal.