Pregunta 2

7,5 / 10

Ordenar datos puede acelerar muchas operaciones. ¿En cuál de las siguiente situaciones es conveniente ordenarlos? Puede haber una o varias respuestas correctas, o incluso ninguna.

Ocultar opciones de respuesta

(A) Calcular el promedio de una lista de elementos.

(B) Determinar la cantidad de elementos que poseen cierta clave

(C) Calcular el valor mínimo y máximo de una lista de elementos.

(D) Determinar si dos palabras (en listas con arreglo) forman un anagrama. Por ejemplo, "algoritmo" y "logaritmo" (https://es.wikipedia.org/wiki/Anagrama).

(E) Encontrar los elementos de una lista cuyas claves son mayores a cierto valor.

Comentarios

- Para determinar si dos palabras son anagrama, conviene ordenarlas. Por ejemplo "algoritmo" y "logaritmo" pasan a ser "agilmoort". Entonces sólo basta comparar ambas secuencias ordenadas.
- Si queremos encontrar todos los valores mayores a cierto valor, y los datos están ordenados, sólo tenemos que ir por derecha a izquierda hasta encontrar un valor menor.
- Para el promedio tenemos que iterar sobre todos los valores, así que ordenarlos no da ningún beneficio.
- Para determinar todos los elementos que con cierta clave, podemos iterar sobre los datos. Si ya están ordenados, cuando hayamos encontrado el dato, ya no tenemos que seguir iterando. Si no están ordenados, debemos iterar sobre todos los datos.
- Si los datos están ordenados, el valor máximo y mínimo está en las puntas.

Pregunta 3

6,66 / 10

¿Sobre qué tipo de datos STL podemos **aplicar** un ordenamiento? Puede haber una o varias respuestas correctas, o incluso ninguna.

Mostrar opciones de respuesta

(A) vector

(B) deque

(C) map

(D) stack

(E) forward_list

Comentarios

Sólo podemos ordenar listas.

Pregunta 4

3,33 / 10

¿Cuál de los siguientes tipos de datos puede utilizarse para **representar** el organigrama de una empresa? Puede haber una o varias respuestas correctas, o incluso ninguna.

Mostrar opciones de respuesta

(A) Un árbol cuyos nodos contienen una lista doblemente enlazada de punteros a nodos hijo.

(B) Un árbol cuyos nodos contienen punteros al primer hijo por izquierda y siguiente hermano por derecha.

(C) Un árbol binario.

(D) Un árbol AVL.

(E) Un árbol cuyos nodos contienen una lista con arreglo de punteros a nodos hijo.

Comentarios

- Los árboles binarios sólo admiten a lo sumo dos hijos. Pero muchas empresas pueden tener a gerentes con más de dos dependientes de él.
- Los árboles AVL son árboles binarios, por tanto tampoco sirven para esta aplicación.
- Los demás árboles no tienen limitación en cuanto a la cantidad de hijos, por tanto sirven.

Pregunta 5

8 / 10

¿Qué **cuellos de botella** pueden presentarse en un ordenamiento? Puede haber una o varias respuestas correctas, o incluso ninguna.

Mostrar opciones de respuesta

- (A) **La comparación**
- (B) El acceso aleatorio (https://es.wikipedia.org/wiki/Acceso_aleatorio)
- (C) **El retardo (https://es.wikipedia.org/wiki/Retardo_(telecomunicaci%C3%B3n))**
- (D) **El acceso de lectura**
- (E) **El acceso de escritura**

Comentarios

Todas las opciones son cuellos de botella!

Pregunta 6

10 / 10

¿Qué tipo de datos STL suele estar **implementado** con un árbol AVL? Puede haber una o varias respuestas correctas, o incluso ninguna.

Mostrar opciones de respuesta

- (A) **multimap**
- (B) deque
- (C) **set**
- (D) heap
- (E) unordered_set

Pregunta 7

10 / 10

En relación al ordenamiento, ¿cuál de las siguientes afirmaciones es **verdadera**? Puede haber una o varias respuestas correctas, o incluso ninguna.

Mostrar opciones de respuesta

- (A) **Las funciones de comparación deben devolver un número real.**
- (B) **Las funciones de comparación deben ser deterministas.**
- (C) **Las funciones de comparación deben diseñarse en función de la aplicación.**
- (D) Las funciones de comparación pueden aplicarse sobre datos cualitativos y cuantitativos (https://es.wikipedia.org/wiki/Propiedad_cualitativa).
- (E) Las funciones de comparación sólo pueden operar con claves numéricas reales.

Pregunta 8

5 / 10

¿Cuál de los siguientes algoritmos es eficiente para encontrar el elemento más pequeño de un **árbol binario**? Puede haber una o varias respuestas correctas, o incluso ninguna.

Mostrar opciones de respuesta

- (A) **DFS pre-orden**
- (B) DFS post-orden
- (C) **BFS**
- (D) DFS in-orden
- (E) El algoritmo de búsqueda estándar (findNode)

Comentarios

Cuidado. Se trata de un **árbol binario**, no un árbol **binario de búsqueda** (por esa razón estaba resaltado en negrita). En un árbol binario los elementos no necesariamente están ordenados, por tanto no queda otra que visitar todos los elementos del árbol.

Pregunta 9

5 / 10

¿Por **qué** los recorridos DFS se implementan con llamadas recursivas a función? Puede haber una o varias respuestas correctas, o incluso ninguna.

Mostrar opciones de respuesta

- (A) Para reducir el consumo de memoria.
- (B) Para mejorar la eficiencia del algoritmo.
- **(C) Para recordar, en orden, dónde nos encontramos en el árbol.**
- (D) Para poder distinguir los casos pre-orden y post-orden.
- (E) No es necesario implementar llamadas recursivas a función, también es posible implementar el algoritmo con una pila.

Comentarios

Para implementar DFS necesitamos una pila, para saber cuán profundo estamos. Podemos aprovechar la pila que nos da el microprocesador (a través de las llamadas recursivas) o podemos usar **stack** de STL.

Pregunta 10

5 / 10

Cierta implementación del algoritmo quicksort utiliza como pivote el **primer elemento** de las listas. ¿Cuándo es este algoritmo $O(n^2)$? Puede haber una o varias respuestas correctas, o incluso ninguna.

Mostrar opciones de respuesta

- (A) Cuando los datos poseen muchas claves repetidas.
- (B) Cuando los datos vienen aproximadamente pre-ordenados.
- (C) Cuando los datos vienen ordenados de menor a mayor.
- **(D) Cuando el pivote es siempre el elemento de clave mayor o menor.**
- **(E) Cuando los datos vienen ordenados de mayor a menor.**

Comentarios

Cuando los datos vienen ordenados de menor a mayor, quicksort elegirá siempre como pivote el elemento menor. Por tanto, las sublistas quedarán muy desbalanceadas.

Lo mismo ocurre cuando los datos vienen ordenados inversamente, o cuando vienen aproximadamente pre-ordenados.

Pregunta 11

10 / 10

¿Cuál de las siguientes afirmaciones es **verdadera**?

Mostrar opciones de respuesta

- (A) Quicksort es más eficiente que insertion sort.
- **(B) La operación merge de merge sort produce listas ordenadas.**
- (C) Bubblesort puede ser más eficiente que quicksort.
- (D) Radix sort es más eficiente que quicksort cuando las claves están distribuidas en forma densa.
- (E) Pigeonhole sort es eficiente cuando las claves están distribuidas en forma dispersa.