

Comenzado el	viernes, 29 de septiembre de 2023, 12:00
Estado	Finalizado
Finalizado en	viernes, 29 de septiembre de 2023, 12:14
Tiempo empleado	14 minutos 44 segundos
Calificación	8,67 de 10,00 (86,67%)

Pregunta 1

Correcta

Se puntúa 1,00 sobre 1,00

La estrategia de ordenación del *heapsort* se puede implementar con cualquier cola de prioridad sin coste adicional, no es necesario utilizar montículos binarios.

- ☐ Verdadero
- ☒ Falso ✓

Falso. El procedimiento se puede aplicar con cualquier cola de prioridad, ya que se basa en las operaciones del tipo abstracto de datos, pero usando otras implementaciones podría no ser posible reutilizar el vector inicial y evitar un coste adicional en espacio.

La respuesta correcta es 'Falso'

Pregunta 2

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es el coste de una implementación eficiente de un algoritmo de ordenación basado en comparaciones de elementos (como son por ejemplo *heapsort*, *mergesort*, etc)?

Seleccione una:

- ☒ a. $O(N \log N)$. ✓
- ☐ b. $O(N)$.
- ☐ c. $O(1)$.
- ☐ d. $O(N^2)$.

No se puede hacer un algoritmo de ordenación basado en comparaciones con coste inferior a $O(N \log N)$ y los ejemplos que hemos puesto son algoritmos con coste en $O(N \log N)$.

La respuesta correcta es: $O(N \log N)$.

Pregunta 3

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es el coste de la operación de añadir un nuevo elemento (*push*) vista en *IndexPQ*?

Seleccione una:

- ☐ a. $O(N \log N)$
- ☒ b. $O(\log N)$ ✓ Cierto. Al añadir un nuevo elemento, éste debe flotarse hasta encontrar su lugar y flotar es logarítmica
- ☐ c. $O(N)$
- ☐ d. Ninguna de las anteriores.

- a. Falso. Incorrecta. Es logarítmica, ya que flotar es logarítmico
- b. Cierto. Al añadir un nuevo elemento, éste debe flotarse hasta encontrar su lugar y flotar es logarítmica
- c. Falso. Incorrecta. Es logarítmica, ya que flotar es logarítmico
- d. Falso. La respuesta correcta es: $O(\log N)$

La respuesta correcta es: $O(\log N)$

Pregunta 4

Incorrecta

Se puntúa -0,33 sobre 1,00

La función

```
void funcion(vector<int> & v) {  
    for (int i = v.size()-1; 0; --i) {  
        flotar_max(v, v.size(), i);  
    }  
}
```

Seleccione una:

- ☐ a. mueve los elementos del vector, pero no lo convierte necesariamente en un montículo.
- ☐ b. convierte un montículo de máximos en uno de mínimos.
- ☐ c. siempre deja el vector sin modificar.
- ☒ d. convierte un vector cualquiera en un montículo. ✗

Un vector puede convertirse en un montículo de dos maneras distintas:

- recorriendo los elementos de izquierda a derecha y *flotando* cada elemento entre los anteriores (ya procesados), o
- recorriendo la primera mitad de los elementos de derecha a izquierda, *hundiendo* cada elemento entre los siguientes, que ya forman montículos.

Aquí se han mezclado las dos ideas, lo que no garantiza que el resultado final sea un montículo, aunque haya elementos que cambien de lugar.

La respuesta correcta es: mueve los elementos del vector, pero no lo convierte necesariamente en un montículo.

Pregunta 5

Correcta

Se puntúa 1,00 sobre 1,00

Para ordenar un vector *de menor a mayor* utilizando el método heapsort (con un coste en espacio adicional constante), primero el vector tiene que convertirse en un montículo de *máximos*.

Seleccione una:

- ☒ a. Verdadero ✓
- ☐ b. Falso

Verdadero. Al convertir el vector, sobre sí mismo, en un montículo de máximos, podemos ir sacando los elementos de mayor a menor, e irlos colocando de derecha a izquierda en el vector, en las posiciones que van quedando vacías al no ser necesarias para almacenar el montículo.

La respuesta correcta es: Verdadero

Pregunta 6

Correcta

Se puntúa 1,00 sobre 1,00

Hemos visto en *IndexPQ* una implementación de *colas con prioridad variable* que hace uso de un vector de pares (elemento, prioridad) y un vector de posiciones de esos elementos.

Imagina que modificamos esta implementación eliminando el vector de posiciones. ¿Cuál de estas afirmaciones es correcta?

Seleccione una:

- ☒ a. Modificar la prioridad (update) pasa a tener coste lineal. ✓ Cierto. El montículo está ordenado según prioridades, no elementos, eso quiere decir que para modificar la prioridad de un elemento, antes, hay que encontrarlo. Como el elemento puede estar en cualquier posición hay que recorrer todos los elementos.
- ☐ b. Hundir pasa a tener coste lineal.
- ☐ c. No hay ningún cambio en el coste de ninguna operación.
- ☐ d. Ninguna de las anteriores.

- a. Cierto. El montículo está ordenado según prioridades, no elementos, eso quiere decir que para modificar la prioridad de un elemento, antes, hay que encontrarlo. Como el elemento puede estar en cualquier posición hay que recorrer todos los elementos.
- b. Falso. Incorrecta. Sigue siendo logarítmica, hundimos un elemento ya encontrado.
- c. Falso. Incorrecta. Por ejemplo, modificar la prioridad (update) pasa a tener coste lineal.
- d. Falso. La respuesta correcta es: Modificar la prioridad (update) pasa a tener coste lineal.

La respuesta correcta es: Modificar la prioridad (update) pasa a tener coste lineal.

Pregunta 7

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es el coste de la operación de eliminar el elemento más prioritario (*pop*) vista en *IndexPQ*?

Seleccione una:

- ☒ a. $O(\log N)$ ✓ Cierto. Al eliminar el elemento más prioritario se debe hundir el elemento en la última posición hasta encontrar su sitio y hundir es logarítmica
- ☐ b. $O(N)$
- ☐ c. $O(N \log N)$
- ☐ d. Ninguna de las anteriores.

- a. Cierto. Al eliminar el elemento más prioritario se debe hundir el elemento en la última posición hasta encontrar su sitio y hundir es logarítmica
- b. Falso. Incorrecta. Es logarítmica, ya que hundir es logarítmico.
- c. Falso. Incorrecta. Es logarítmica, ya que hundir es logarítmico.
- d. Falso. La respuesta correcta es: $O(\log N)$

La respuesta correcta es: $O(\log N)$

Pregunta 8

Correcta

Se puntúa 1,00 sobre 1,00

Dado un vector ordenado de menor a mayor, ¿cuál es el coste de convertirlo en un montículo de máximos?

Seleccione una:

- ☐ a. $O(N \log N)$
- ☐ b. No está garantizado que se pueda convertir en un montículo de máximos.
- ☒ c. $O(N)$. ✓
- ☐ d. $O(1)$.

Sí que se puede convertir en un montículo de máximos. Ahora bien, si el vector está ordenado de menor a mayor, cada elemento (en la primera mitad del vector) es menor que todos los siguientes por lo que tendrá que ser hundido hasta una hoja. Esto quiere decir que el coste será de $O(N)$.

La respuesta correcta es: $O(N)$.


Pregunta 9

Correcta

Se puntúa 1,00 sobre 1,00

Considerando el TAD de *colas con prioridades variables* implementado en la clase *IndexPQ* visto, ¿cuál de estas operaciones tiene un coste lineal?

Seleccione una:

- ☐ a. La operación de consulta del tamaño de la cola (*size()*).
- ☐ b. La operación de añadir un nuevo elemento (*push*).
- ☐ c. La operación de eliminación del elemento más prioritario (*pop*).
- ☒ d. Ninguna de las anteriores.  Cierto. La respuesta correcta es: La constructora, es decir, crear una cola vacía

- a. Falso. Incorrecta. Es constante, basta consultar el tamaño del array.
- b. Falso. Incorrecta. Es logarítmica, ya que flotar es logarítmico.
- c. Falso. Incorrecta. Es logarítmica, ya que tanto flotar como hundir son logarítmicos.
- d. Cierto. La respuesta correcta es: La constructora, es decir, crear una cola vacía

La respuesta correcta es: Ninguna de las anteriores.


Pregunta 10

Correcta

Se puntúa 1,00 sobre 1,00

Cuando en un montículo de máximos cambiamos la prioridad de un elemento a un valor *menor*, el elemento tiene que ser flotado (intercambiado con su padre mientras este sea menor).

Seleccione una:

- ☐ a. Verdadero
- ☒ b. Falso 

Falso. Al ser un montículo de máximos, los elementos mayores son los que más cerca están de la raíz. Si la prioridad de un elemento se hace menor, el elemento tiene que ser hundido.

La respuesta correcta es: Falso