## Pregunta 2

6,66 / 10

¿Cuál de los siguientes tipos de datos podría usarse para representar un **grafo**? Puede haber una o más respuestas correctas.

Mostrar opciones de respuesta

(A) typedef std::unordered_map<int, std::forward_list<int>> Graph;

(B) **typedef std::vector<std::vector<int>> Graph;**

(C) typedef std::stack<std::list<vector>> Graph;

(D) **typedef std::vector<int> GraphNodes;**
**typedef std::vector<int> GraphNodeNeighbors;**

(E) **typedef std::vector<bool> GraphMatrix;**

Comentarios

* typedef std::unordered_map<int, std::forward_list<int>> Graph; define un grafo válido: en el primer nivel hay una lista de nodos con una tabla hash, en el segundo nivel una lista de vecinos.
* typedef std::vector<std::vector<int>> Graph; define un grafo válido: en el primer nivel hay una lista de nodos, en el segundo nivel una lista de vecinos.
* typedef std::vector<bool> GraphMatrix; define un grafo válido: utiliza la representación de matriz de adyacencia.

* typedef std::stack<std::list<vector>> Graph; no define un grafo válido: stack no permite el acceso por índice.
* typedef std::vector<int> GraphNodes; typedef std::vector<int> GraphNodeNeighbors; no definen un grafo válido: si bien define una lista de nodos y una lista de vecinos, no asocia a cada nodo una lista de vecinos.

## Pregunta 3

10 / 10

Si un grafo es **denso** y debemos **recorrerlo frecuentemente**...

Mostrar opciones de respuesta

(A) conviene siempre implementarlo con una matriz de adyacencia.

(B) **en general, conviene implementarlo con una matriz de adyacencia.**

(C) en general, no conviene implementarlo con una matriz de adyacencia.

(D) nunca conviene implementarlo con una matriz de adyacencia.

(E) la respuesta depende de si las claves están densamente distribuidas.

## Pregunta 4

10 / 10

¿Cuál de las siguientes tareas es más **adecuada** para aprovechar el paralelismo SIMD? Puede haber una o más respuestas correctas.

Mostrar opciones de respuesta

(A) **Realizar operaciones aritméticas en una matriz de números enteros.**

(B) **Calcular el producto escalar de dos vectores.**

(C) Buscar un elemento específico en un árbol binario de búsqueda.

(D) Ordenar una lista de elementos de manera ascendente.

(E) Realizar cálculos matemáticos complejos en un solo número.

## Pregunta 5

10 / 10

¿Cuál es la **complejidad computacional** del algoritmo floodfill aplicado a un grafo de N nodos y E arcos?

Mostrar opciones de respuesta

(A) $O(N^2)$

(B) $O(N \log(N))$

(C) $O(N+E)$

(D) $O(N)$

(E) $O(E)$

## Pregunta 6

0 / 10

¿Cuál es el **propósito principal** del algoritmo floodfill?

Mostrar opciones de respuesta

(A) Encontrar el camino más corto entre dos nodos.

(B) **Determinar si un grafo es conexo o no.**

(C) Etiquetar las componentes conexas en un grafo.

(D) Encontrar el nodo con el grado (número de arcos incidentes) más alto en un grafo.

(E) Ordenar los nodos de un grafo topológicamente.

**Comentarios**

El propósito principal de floodfill es "pintar" el área formada por elementos contiguos de una matriz... o sea, etiquetar las componentes conexas.

El propósito principal de floodfill no es:
* encontrar el camino más corto entre dos nodos, ya que floodfill alcanza todos los nodos accesibles desde el nodo inicial; no se detiene antes.
* determinar si un grafo es conexo o no, ya que el algoritmo no determina si se alcanzaron todos los nodos. Sería necesario añadir una segunda operación que determine esto.
* no hace cálculos sobre el número de arcos incidentes, por tanto no encuentra el nodo con el grado más alto.
* no hace ninguna clase de ordenamiento topológico (como vimos en clase).

## Pregunta 7

0 / 10

En un grafo implementado con listas de adyacencia, ¿cuál es la manera más **eficiente** de encontrar un nodo por su clave?

Mostrar opciones de respuesta

(A) Recorrer el grafo con DFS desde un nodo cualquiera.

(B) Recorrer el grafo con BFS desde un nodo cualquiera.

(C) **Iterar sobre los nodos no visitados, y realizar desde cada uno de ellos un recorrido DFS.**

(D) Iterar sobre los nodos no visitados, y realizar desde cada uno de ellos un recorrido BFS.

(E) Iterar sobre la lista de nodos.

**Comentarios**

Iterar sobre la lista de nodos es O(N). Si la lista de nodos está ordenada, puede ser incluso O(log(n)).

Aplicar DFS o BFS es O(N + E), en donde E es la cantidad de arcos.

## Pregunta 8

6,66 / 10

¿Cuáles son ventajas potenciales del uso de multithreading en algoritmos y estructuras de datos? Puede haber una o más respuestas correctas.

Mostrar opciones de respuesta

(A) **Mejor utilización de los recursos de la CPU.**

(B) **Mejor rendimiento en sistemas con múltiples núcleos.**

(C) Mayor facilidad de implementación en comparación con el enfoque secuencial.

(D) Mejor capacidad de escalabilidad (capacidad de adaptarse al aumento de la carga sin comprometer el rendimiento).

(E) Menor consumo de memoria.

**Comentarios**

El multithreading ayuda a aprovechar más los recursos y los núcleos de la CPU. Por tanto ayuda a mejorar la capacidad de escalar el problema.

Implementar código con multithreading es más difícil que implementar código con un sólo núcleo, porque el multithreading requiere sincronización.

El multithreadhing requiere más memoria, por el pasaje de datos entre threads.

## Pregunta 9

10 / 10

En relación a la búsqueda por fuerza bruta, ¿cuál de las siguientes afirmaciones es **verdadera**? Puede haber una o más respuestas correctas.

Mostrar opciones de respuesta

- (A) Siempre garantiza la solución óptima para cualquier problema.
- (B) Es una estrategia que se limita a problemas de búsqueda de texto.
- (C) No requiere conocimiento previo del espacio de búsqueda para ser aplicada.
- (D) Es una estrategia que se basa en técnicas de programación dinámica.
- **(E) Puede requerir un tiempo de ejecución prohibitivo en problemas de gran tamaño.**

## Pregunta 10

10 / 10

Para un grafo no-pesado, ¿cuál de las siguientes afirmaciones es **falsa**?

Mostrar opciones de respuesta

- **(A) DFS encuentra siempre el camino más corto entre dos nodos del grafo.**
- (B) BFS puede ser utilizado para encontrar las componentes conexas del grafo si el grafo es no-dirigido.
- (C) DFS puede ser utilizado para detectar ciclos en el grafo.
- (D) BFS puede ser utilizado para encontrar el camino más corto entre dos nodos.
- (E) DFS puede ser utilizado para encontrar las componentes conexas del grafo si el grafo es no dirigido.

## Pregunta 11

0 / 10

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

Mostrar opciones de respuesta

- **(A) Es una técnica algorítmica utilizada para resolver problemas de manera iterativa.**
- (B) Requiere que los subproblemas se resuelvan en un orden topológico.
- (C) Es especialmente eficiente para problemas que exhiben la propiedad de superposición de subproblemas.
- (D) Se utiliza exclusivamente en problemas de optimización.
- (E) Es una estrategia que sólo se aplica a problemas de grafos.

**Comentarios**

El pilar de la programación dinámica es la memoización, la capacidad de recordar soluciones a sub-problemas. Cuando estos sub-problemas aparecen repetidas veces, o sea, cuando se superponen los sub-problemas, la programación dinámica es muy efectiva.

* La programación dinámica no es iterativa. Éste es más bien el caso de la búsqueda por fuerza bruta.
* No requiere ordenamiento topológico.
* Se puede utilizar en toda clase de problemas en los que hay sub-problemas superpuestos, no sólo en problemas de optimización. Un contraejemplo: el problema de calcular los números de Fibonacci (que vimos en la primera clase).
* Aplica a cualquier clase de problemas, no sólo a problemas de grafos.