Tema C - Hoja 2

Pregunta 8 ♠ ¿Cuál/es de las siguientes afirmaciones son correctas sobre el recorrido BFS?

- ☐ Se necesita que el grafo de entrada sea representado con una matriz de adyacencias para alcanzar la complejidad óptima.
- ☐ Todos los vértices a distancia k de la raíz son visitados antes que los de distancia $k+1$.
- ☐ Para cada grafo G hay un único árbol BFS posible.
- ☐ Para toda arista (u, v) del grafo original, o u es ancestro de v en el árbol BFS, o v es ancestro de u.

Supongamos que tenemos el siguiente algoritmo que toma como entrada un digrafo $D = (V, E)$ con n vértices, donde $V = \{0, \ldots, n-1\}$.

Algoritmo 3 Procesar digrafo

```
1:  function F(D)
2:      Inicializar visitados como un arreglo de n elementos, todos False.
3:      suma ← 0
4:      for v ∈ V do
5:          if ¬visitados[v] then
6:              DFS(D, v, visitados)
7:              suma ← suma + 1
8:          end if
9:      end for
10:     return suma
11: end function
```

Pregunta 9 ¿Qué complejidad temporal se ajusta mejor a la del algoritmo 5 en el peor caso?

- ☐ $O(|V| \cdot (|V| + |E|))$
- ☐ $O(|V| \cdot |E|)$
- ☐ $O(|V| + |E|)$
- ☐ $O(|E|^2)$
- ☐ $O(|V|)$

Pregunta 10 ¿Qué representará el valor de retorno del algoritmo 5?

- ☐ La cantidad de vértices de D con grado de entrada 0.
- ☐ Ninguna de estas respuestas es correcta.
- ☐ La cantidad de componentes fuertemente conexas de D.
- ☐ La cantidad de componentes conexas de D.
- ☐ La cantidad total de vértices de D.

2. Ejercicio a desarrollar (40 ptos.)

Marcar con una cruz el cuadrado (☐) del ejercicio que vas a entregar.
Solo se entrega 1 de los 2 ejercicios.

2.1. Problema A

Alfredo se está tomando unos días de descanso, así que va a recorrer diferentes pueblos de la Argentina y ha decidido que cuando regrese les regalará un alfajor a cada uno de los k demás docentes de la materia. Para esto ha decidido que los comprará en cada uno de los n pueblos que recorrerá. Al llegar al i-ésimo pueblo tiene tres opciones:

- Comprar media docena (6) de alfajores locales por c_i pesos.
- Comerse un alfajor en lugar de ir a la tienda a comprar.
- No hacer ninguna de las dos anteriores.

Y es que a Alfredo le gustan mucho los alfajores, y siente que si pasa g ciudades sin comerse al menos uno, se estresará mucho, lo cual repercutiría en sus clases. Además sabemos que $g \leq n$.

a) Sea c la lista de precios de los alfajores $c_1, c_2, \ldots, c_n$, definir de forma recursiva $f_{c,n,k,g} : \mathbb{N}^3 \to \mathbb{N}$ donde $f_{c,n,k,g}(i, a, h)$ calcula la cantidad mínima de pesos que Alfredo gastará al partir de la ciudad i con a alfajores y h ciudades que le quedan por recorrer sin comer alfajores antes que le agarre mal humor. ¿Qué llamado(s) se necesitan para resolver el problema? ¿Qué operación se debe realizar sobre esos llamados? **Importante:** acompañen a la definición recursiva con una explicación en castellano.

b) Diseñe e implemente un algoritmo empleando dicha función, mencionando las estructuras que utilizaría. La complejidad del algoritmo debe ser $O(n \cdot \text{mín}\{n, k + \frac{n}{g}\} \cdot g)$ o mejor. (Ayuda: argumentar que no necesita comprar más de $k + \frac{n}{g} + 6$ alfajores).

c) Demuestre que el algoritmo cumple con la propiedad de superposición de problemas, comparando con la solución que emplea backtracking.

d) ¿Es posible usar el algoritmo anterior para determinar en qué pueblos tendría que comprar y en cuáles comerse un alfajor? ¿Es posible hacerlo sin empeorar la complejidad temporal? De ser posible, explique cómo lo haría.

2.2. Problema B

Supongamos que un sistema informático consta de varios módulos interconectados, representados por un digrafo D donde cada vértice simboliza un módulo y cada arista dirigida indica una dependencia directa (es decir, el módulo en el vértice de origen necesita que el módulo en el vértice de destino esté actualizado para poder funcionar correctamente). Se desea implementar una serie de actualizaciones en el sistema, pero es crucial que no existan interdependencias circulares que puedan causar bloqueos o fallos en la actualización.

El objetivo es determinar si es posible establecer un plan de actualización que respete las dependencias entre módulos y, en caso afirmativo, proporcionar un procedimiento para actualizar los módulos de manera segura.

- Un sistema es considerado **mantenible** si y solo si no contiene ciclos de dependencias. Esto asegura que existe al menos una forma de actualizar todos los módulos respetando sus dependencias.
- **Orden de actualización seguro:** Un orden en el cual se pueden actualizar los módulos sin violar ninguna dependencia.

a) **Demostración de la secuencia de actualización:** Demostrar que si D no tiene ciclos, entonces es posible establecer una secuencia en la que se puedan procesar los módulos de manera que cada módulo se procese sólo después de haber procesado todos los módulos de los cuales depende directamente.

b) **Diseño de algoritmo:** Describir un algoritmo que verifique si el sistema es mantenible y, en caso positivo, determine un orden de actualización seguro. El algoritmo debería tener una complejidad temporal de $O(n + m)$, donde n es el número de módulos y m el número de dependencias.

c) **Descripción del algoritmo en pseudocódigo:** Proporcionar un pseudocódigo detallado y claro del algoritmo diseñado en el inciso b). La descripción debe permitir comprender plenamente la lógica y justificación del algoritmo.