Técnicas de Diseño de Algoritmos
Primer Cuatrimestre 2024
Primer Parcial

Nombre y Apellido	LU
.....	.....

Duración : 4 horas
*Este examen es a **libro cerrado**.*
Para aprobar el parcial se debe alcanzar una nota de 50 puntos.

1. Preguntas opción múltiple (60 ptos.)

Las preguntas con el símbolo ♣ pueden tener cero, una o varias respuestas correctas. Las otras preguntas tienen una única respuesta correcta.
Un ejercicio con todas sus opciones V marcadas, y todas sus F no marcadas suma 6 puntos.

Pregunta 1 ♣ "A guardar, a guardar..." dice la profe de jardín, y Eric sabe que tiene que guardar todos los juguetes que usó en los cajones. Sabe que puede usar hasta c cajones y que usó j juguetes, con $c < j$. El ancho y largo de los juguetes es justo el ancho y el largo de los cajones. Cada cajón tiene la misma altura A, y cada juguete i, con $1 \leq i \leq j$, tiene altura a_i. Sabe que hay alguna forma de poner todos los juguetes en los cajones, pero no se acuerda de qué cajón sacó cada juguete. Va a hacer lo único que sabe: buscar la solución con backtracking. ¿Qué complejidades temporales serían las más ajustadas para los algoritmos de backtracking que podría usar?

- ☐ $O(A \cdot \sum_{i=1}^{j} a_i)$
- ☐ $O(c^j)$
- ☐ $O(j! \cdot j)$
- ☐ $O(j!)$

Pregunta 2 En la programación dinámica siempre se cumple que:

- ☐ Utilizar memorización garantiza que solo se calculen los subproblemas necesarios, lo que siempre reduce el tiempo de ejecución del algoritmo.
- ☐ Cuándo hay múltiples parámetros en la función implementada se necesita una matriz (de 2 o más dimensiones) para memorizar los resultados
- ☐ Utilizar memorización te obliga a utilizar una solución recursiva para cada problema.
- ☐ Ninguna de estas respuestas es correcta.
- ☐ Hay más de un posible valor de retorno para cada subproblema.

Pregunta 3 ♣ Se tiene un arreglo A ordenado decrecientemente de n números naturales consecutivos con k huecos, es decir, en el arreglo todos los elementos son consecutivos excepto en k posiciones. Por ejemplo, el arreglo $A = [19, 16, 15, 13, 12, 11]$ tiene 2 huecos en los valores $(19, 16)$ y $(15, 13)$. El tamaño de un hueco (x, y) se define cómo $|x - y|$. Se puede asumir que el arreglo tiene tamaño potencia de 2. Sea t el tamaño del hueco más grande. ¿Cuál sería la cota más ajustada del mejor algoritmo $D\&C$ en base a A, n, k, t que encuentra el valor de t?

- ☐ $O(log(n) * k)$
- ☐ No se puede calcular su complejidad con el teorema maestro pues resolver su caso base no es $O(1)$.
- ☐ No se puede calcular su complejidad con el teorema maestro pues no siempre se subdivide en la misma cantidad de suproblemas.
- ☐ $O(log(n) * k + t)$
- ☐ $O(n * k)$

Pregunta 4 El tiempo de ejecución de un determinado algoritmo responde a la siguente recurrencia, con su caso base para $T(i) \in \Theta(1)$ para todo $i < 10$.

$$T(n) = 3 * T\left(\frac{n}{9}\right) + 5 * n^{1/4}$$

Para determinar la complejidad de este algoritmo, se nos presenta la siguiente demostración utilizando el teorema maestro de que la recurrencia es $\Theta(\sqrt{n})$.
Nos encontramos en el primer caso del teorema maestro. Puesto que para $\epsilon = -6$, se cumple que $f(n) = 5 * n^{1/4} \in O\left(n^{\log_c a - \epsilon}\right) = O\left(n^{\log_9 3+6}\right) = O\left(n^1\right)$, por lo que $T(n) = \Theta\left(n^{\log_c a}\right) = \Theta(\sqrt{n})$.

- ☐ La complejidad es ajustada, la demostración es incorrecta.
- ☐ La complejidad no es ajustada, la recurrencia es $O\left(n^{1/4}\right)$
- ☐ La complejidad no es ajustada, la recurrencia es $O\left(n^{1/3}\right)$
- ☐ No se puede utilizar teorema maestro para esta función de recurrencia
- ☐ La demostración es correcta.

Pregunta 5 ♣ Un grafo *etiquetado* G es un grafo cuyos vértices tienen cada uno una etiqueta distinta en el conjunto $\{1, \ldots, n\}$, donde n es la cantidad de vértices de G. En G, cada arista se etiqueta utilizando las etiquetas de sus vértices incidentes. Dos grafos etiquetados de n vértices son *iguales* cuando tienen el mismo conjunto de aristas etiquetadas. Una *orientacion acíclica* de un grafo (etiquetado o no) G es un grafo orientado H que no tiene ciclos dirigidos y que resulta de asignarle una dirección a cada arista de G. Ciertamente, si G es etiquetado, entonces H mantiene las mismas etiquetas que G. ¿Cuál/es de las siguientes afirmaciones son verdaderas?:

- ☐ Todo grafo etiquetado con al menos una arista tiene una cantidad impar de orientaciones acíclicas distintas.
- ☐ Todo grafo etiquetado tiene al menos un *sumidero*[*].
- ☐ Todo grafo etiquetado con al menos una arista tiene una cantidad par de orientaciones acíclicas distintas.
- ☐ Toda orientación acíclica de un grafo tiene al menos un *sumidero*[*].

[*] Un sumidero es un vértice con grado de salida cero.

Pregunta 6 ♣ Seguimos en el contexto del grafo etiquetado de la pregunta anterior. ¿Cuál/es de las siguientes afirmaciones son verdaderas? Aclaración: K_n es el grafo completo de n vértices.

- ☐ Toda orientación acíclica de K_n tiene un único sumidero.
- ☐ Toda orientación acíclica de K_n tiene exactamente dos sumideros (para $n \geq 4$).
- ☐ El grafo K_n etiquetado tiene exactamente n^2 orientaciones acíclicas distintas.
- ☐ Si a una orientación acíclica H de un grafo le agregamos un nuevo vértice v junto a una arista $u \to v$ para todo vértice u de H, entonces el grafo resultante es la orientación acíclica de un grafo.

Pregunta 7 ¿Cuál de las siguientes propiedades sobre grafos es falsa?

- ☐ El digrafo resultado de darle una orientación a un grafo completo tiene un camino de longitud $n - 1$.
- ☐ Si G es un grafo conexo entonces no es un grafo junta. [*]
- ☐ La existencia de un ciclo en un grafo asegura que el grafo es conexo.
- ☐ Si G tiene exactamente 2 vértices con grado impar, tiene que haber un camino entre ellos.

[*] Un grafo junta J, $J = G + H$ de G y H es el grafo que se obtiene de $G \cup H$ agregando todas las aristas (v, w) posibles entre un vértice $v \in V(G)$ y otro vértice $w \in V(H)$

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.
- ☐ Para toda arista (u, v) del grafo original, o u es ancestro de v en el árbol BFS, o v es ancestro de u.
- ☐ 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.

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 1 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 1 en el peor caso?

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

Pregunta 10 ¿Qué representará el valor de retorno del algoritmo 1?

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

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.