Técnicas de Diseño de Algoritmos
Primer Cuatrimestre 2024
1er Recuperatorio (T. Mañana)

Nombre y Apellido	L.U.	N°Orden
.....	.....	.....

*Duración: **3 horas**. Este examen es a **libro cerrado**. Todas las preguntas tienen $k \geq 1$ opciones correctas. Las respuestas donde se marquen exactamente esas k dan 2 puntos, aquellas donde se marquen al menos $\lceil k/2 \rceil$ correctas y más correctas que incorrectas dan 1 punto, las que tengan menos de $\lceil k/2 \rceil$ correctas o igual o más incorrectas que correctas dan 0, y si todas las marcadas son incorrectas dan -1 punto. Para aprobar el parcial se deben sumar $p \geq 12$ puntos y la nota será $5p/12$.*

Pregunta 1 Julieta encontró una caja con n piezas distinguibles de dominó, y se propuso encontrar el camino de mayor longitud que puede hacerse con las mismas[*]. Su estrategia para generar todos los candidatos a caminos consiste en probar cada posible ordenamiento de las piezas con cada posible orientación de cada pieza en en la permutación, para luego buscar el camino más largo que quedó en esa configuración. Por suerte, conoce una técnica que le permite iterar por cada configuración, sin repetir, en tiempo constante. Tras unos minutos empleando esta estrategia se le ocurrió la siguiente idea: va a empezar de cero e ir agregando en cada paso una nueva pieza a la derecha del camino actual que está armando, probando una a una todas las piezas que sean compatibles con el camino armado hasta el momento (es decir, que se puedan orientar para que un lado coincida con la pieza más a la derecha del camino, considerando que la primera se puede orientar de las dos maneras). Cada vez que se quede sin piezas compatibles para seguir poniendo, volverá al último paso donde pueda poner alguna pieza que aún no haya probado en esa posición, y seguirá el camino utilizando esa pieza. Finalmente, se quedará con la mayor longitud que haya alcanzado formar durante este proceso. Decidir:

1 ☐ La segunda estrategia propuesta por Julieta no es correcta, existen casos para los que proveerá una longitud de cadena de dominós menor a la máxima posible.

2 ☐ La primera estrategia implica revisar $2^n \cdot n!$ soluciones candidatas, y toma $O(n)$ tiempo obtener el camino más largo en cada una.

3 ☐ En la segunda estrategia, para todo $1 \leq k \leq n-1$, una vez que Julieta llegó a armar un camino de longitud k, para la $(k+1)$-ésima pieza puede tener $2^{n-k}(n-k)$ opciones distintas posibles a considerar.

4 ☐ Una cota para la complejidad temporal de la segunda estrategia definida por Julieta es $O(2^n \cdot n)$.

[*]Las piezas de dominó son rectangulares y tienen un número de cada lado. Una pieza se puede unir a otra si coinciden en algún número, y se unen a través ese mismo número. Por lo tanto, el número de ese lado ya no se puede usar para unir otra pieza. Por ejemplo, si tenemos las piezas $(2,3)$, $(1,3)$ y $(1,4)$, podemos unrlas en el camino $(2,3),(3,1),(1,4)$. La longitud de un camino es igual a la cantidad de piezas en el camino.

Pregunta 2 ¿Cuál(es) de las siguientes afirmaciones describe(n) a las estrategias greedy?

1 ☐ El algoritmo greedy correcto provee una solución que es estrictamente mejor que la de un algoritmo que no sea greedy.

2 ☐ Para todo problema y estrategia greedy, si la estrategia no devuelve el óptimo para alguna entrada, entonces existe alguna entrada para la que la estrategia proveerá una solución arbitrariamente lejana al óptimo.

3 ☐ Las estrategias greedy garantizan que cada decisión tomada siempre deriva en una solución óptima.

4 ☐ Una forma de demostrar que una estrategia greedy da con el valor óptimo en un problema de optimización consiste en probar que la solución de la estrategia es por lo menos tan buena como cualquier solución posible.

5 ☐ Si se conoce una recursión f para construir las soluciones candidatas de un problema, las estrategias greedy que toman decisiones siguiendo la recursión f exploran exactamente una rama del árbol de backtracking definido por f.

6 ☐ Para poder afirmar fuera de toda duda que una estrategia greedy es un algoritmo que resuelve un problema dado, necesitamos una demostración.

Pregunta 3 Francesco es un niño que adora hacer travesuras. Un día, durante el recreo, Francesco tomó el celular de su *seño* y lo escondió debajo de una baldosa del patio de su colegio. El patio puede representarse como una grilla P de $n \times n$ baldosas, donde n es una potencia de 2. Afortunadamente, contamos con un dispositivo que puede ser de utilidad: dada una intersección (i, j) de P y valor d, con $1 \leq d \leq n$ y $0 \leq i, j \leq n - d$, , el dispositivo (que representamos como la función D) nos puede decir en $O(d)$ tiempo si hay un dispositivo bajo alguna baldosa de la región cuadrada de $d \times d$ en P cuya intersección superior izquierda es (i, j). Luego, dados (i, j) y d, podemos definir el algoritmo $B(i, j, d)$ que nos devuelve la intersección superior izquierda de la baldosa bajo la cual está el teléfono, suponiendo que hay exactamente uno en esa región:

1. Si $d = 1$, entonces devuelve (i, j) (El teléfono sólo puede estar en un lugar).
2. Si no, invoca a $D(i, j, d/2)$, $D(i + d/2, j, d/2)$, $D(i, j + d/2, d/2)$, y $D(i + d/2, j + d/2, d/2)$. Exactamente uno de esos llamados va a dar verdadero bajo nuestra suposición.
3. Se llama a B recursivamente sobre el cuadrante en el que D haya dado verdadero.

¿Cuál de estas recurrencias representa al tiempo que tomaría dar con el teléfono utilizando el algoritmo B en un patio con lado n y cuál es su complejidad temporal más ajustada? (Sabemos que en todo el patio hay exactamente un teléfono escondido).

1 ☐ $T(n) = T(n/4) + O(n)$
2 ☐ $T(n) = T(n/2) + O(n)$
3 ☐ $T(n) = 4T(n/4) + O(n)$
4 ☐ $T(n) = 4T(n/4) + O(1)$
5 ☐ $O(n)$
6 ☐ $O(\log n)$
7 ☐ $O(n^2)$
8 ☐ $O(n \log n)$

Pregunta 4 Para las siguientes recurrencias marque el orden de complejidad más ajustado.

- $T(n) = 3T(n/3) + n/2$

1 ☐ $O(\log n)$
2 ☐ $O(n)$
3 ☐ $O(n^3)$
4 ☐ $O(n \log n)$
5 ☐ $O(n^2)$
6 ☐ $O(n^2 \log n)$

- $T(n) = 5T(n/5) + \sqrt{n}$

7 ☐ $O(\sqrt{n})$
8 ☐ $O(n)$
9 ☐ $O(n \log n)$
10 ☐ $O(\sqrt{n} \log n)$
11 ☐ $O(n^2)$
12 ☐ $O(n^5)$

- $T(n) = 2T(n/4) + n^{3/4}$

13 ☐ $O(n)$
14 ☐ $O(\log n)$
15 ☐ $O(n \log n)$
16 ☐ $O(n^{3/4})$
17 ☐ $O(\sqrt{n})$
18 ☐ $O(n^2)$

- $T(n) = 25T(n/5) + 8n^2$

19 ☐ $O(\log n)$
20 ☐ $O(n)$
21 ☐ $O(n \log n)$
22 ☐ $O(n \log^2 n)$
23 ☐ $O(n^2 \log n)$
24 ☐ $O(n^5)$

Pregunta 5 Román quiere resolver el problema SUBSET-SUM: dada una lista de números naturales $A = \{a_1, \ldots, a_n\}$ de longitud par mayor a cero y un número natural k, quiere decidir si existe un subconjunto $S \subseteq A$ tal que $\sum_{a_i \in S} a_i = k$. Para esto, propone inicialmente un algoritmo que implementa la siguiente recursión:

$$ss(i,j) = \begin{cases} \texttt{True} & i = n+1 \land j = 0 \\ \texttt{False} & (i = n+1 \land j > 0) \lor (j < 0) \\ ss(i+1,j) \lor ss(i+1, j-a_i) & \text{caso contrario} \end{cases}$$

y dice que $ss(1,k)$ le dará la solución al problema. Decidir:

1 ☐ La cota $O(n2^n)$ es ajustada para la complejidad temporal de este algoritmo.

2 ☐ El algoritmo propuesto por Román es correcto.

3 ☐ Para cualquier A, al llamar la función ss con parámetros 1 y k siempre habrá por lo menos 2^n subllamados.

Para mejorar su algoritmo, Román propone la siguiente estrategia: dividirá el conjunto A en dos mitades $A_1 = \{a_1, \ldots, a_{n/2}\}$ y $A_2 = \{a_{n/2+1}, \ldots, a_n\}$, y para $i \in \{1, 2\}$ calculará, con backtracking, el conjunto T_i, que denota todas las sumas que se pueden obtener usando subconjuntos de A_i (formalmente, $T_i = \{m \in \mathbb{N}_0 : \exists S \subseteq A_i, \sum_{a \in S} a = m\}$). Luego, ordenará T_1 y T_2 e iterará sobre los elementos de T_1 haciendo búsqueda binaria sobre T_2 para para detectar si existen dos números $t_1 \in T_1$ y $t_2 \in T_2$ tales que $t_1 + t_2 = k$. Decidir:

4 ☐ La complejidad espacial de este segundo algoritmo es $O(n^2)$.

5 ☐ El segundo algoritmo no es correcto cuando el valor k se obtiene usando elementos de sólo una de A_1 o A_2.

6 ☐ Una cota para la complejidad temporal de este algoritmo es $O(n2^{\frac{n}{2}})$.

7 ☐ El segundo algoritmo propuesto por Román es correcto.

Pregunta 6 Alfredo se fue de viaje, y a la vuelta espera comprar alfajores. Conoce las n ciudades y el orden en que las va a visitar en su camino a Buenos Aires (numerándolas de 1 a n), y en cada una de ellas puede comprar entre 0 y m alfajores. Denotamos como $p_{i,k}$ a lo que cuesta comprar k alfajores en la ciudad i, para $1 \leq i \leq n$ y $1 \leq k \leq m$, y definimos $p_{i,0} = 0$. Alfredo inicia el viaje con un capital P, y su objetivo es maximizar la cantidad de alfajores que tiene al llegar a Buenos Aires. Para resolver este problema propone implementar la siguiente función recursiva:

$$f(i,c) = \begin{cases} -\infty & c < 0 \\ 0 & i = n+1 \\ \text{máx}_{0 \leq k \leq m}\{f(i+1, c - p_{i,k}) + k\} & \text{caso contrario} \end{cases}$$

que (supuestamente) indica "La mayor cantidad de alfajores que Alfredo puede obtener empezando en la ciudad i, con capital c". Alfredo nos dice que la solución al problema se obtiene implementando f y llamando a $f(1, P)$. Decidir:

1 ☐ La implementación de la solución de Alfredo basada en programación dinámica, que toma $p_{i,k}$ con $0 \leq i \leq n$, $1 \leq k \leq m$ y el valor P, tiene complejidad temporal polinomial respecto al tamaño de su entrada.

2 ☐ La función f está bien definida pero la solución del problema es con el llamado $f(n, P)$ en vez de $f(1, P)$.

3 ☐ La estrategia *bottom-up* que vimos en la materia permite una implementación que consume $O(P)$ espacio.

4 ☐ Existe alguna entrada para la que al llamar $f(1, P)$ se terminan haciendo $(m+1)^n$ subllamados a f.

5 ☐ Si D es una cota de la cantidad de posibles llamados con argumentos distintos a f en la recursión comenzando de $f(1, P)$, entonces la complejidad temporal de aplicar programación dinámica para f es $O(D)$.

6 ☐ La estrategia *bottom-up* que vimos en la materia permite una implementación que consume $O(n)$ espacio.

7 ☐ Existen valores de n y P para los cuales f presenta superposición de subproblemas y valores para los que no.

Pregunta 7 Aye está remodelando su casa, y quiere cubrir el piso de su cuarto de n centímetros de largo con hileras de placas de madera. Compró un tablón de n centímetros, el cual quiere usar para la primera hilera. Ella sabe que no puede poner un segmento de madera de más de ℓ centímetros en el piso, porque se rompe. Por esto, quiere recortar su tablón de forma que cada segmento tenga a lo sumo tamaño ℓ. Además, en algunos lugares de la habitación va a poner muebles que cubran algunas partes del piso, así que se le ocurrió que si corta el tablón sólo en lugares que vayan a quedar cubiertos, va a quedar todo mucho más lindo. Como ya tiene proyectado dónde va a poner los muebles, hizo marcas en la madera que corresponden a lugares en donde podría hacer los cortes (y sabe que esas marcas son suficientes como para que se puedan dejar tablas de la longitud requerida). Aye quiere resolver el tema rápido, por lo que propone la siguiente estrategia para minimizar la cantidad de cortes para un tablón dado:

1. Si el tablón tiene longitud menor o igual a ℓ, no hace nada.
2. En caso contrario, desde el borde izquierdo del tablón, elige la marca más lejana que esté a distancia menor o igual que ℓ posible, y corta allí.
3. Realiza el mismo procedimiento con el tablón de madera restante a la derecha del corte.

Aye afirma que esta estrategia la lleva a una manera de cortar la madera la mínima cantidad de veces. Incluso, pensó la siguiente demostración de que cualquier esquema requiere tantos o más cortes que el suyo: Consideremos un esquema de k cortes válido cualquiera C y el esquema de q cortes de Aye C^A, y sean $c_1 < \ldots < c_k$ y $c_1^A < \ldots < c_q^A$ los puntos donde se hacen los cortes de C y C^A medidos desde el borde izquierdo del tablón, respectivamente. Si $q = 0$, es porque el tablón tiene longitud menor o igual a ℓ y Aye no hace ningún corte, así que $0 = q \leq k$. Si $q > 0$, por definición de C_A el tablón tiene longitud mayor a ℓ y en consecuencia todo esquema válido tiene al menos un corte. Sea i tal que c_i es el máximo corte en C con $c_i \leq \ell$. Vemos que $c_i \leq c_1^A$, pues por definición c_1^A está en la marca máxima que no se pase de ℓ. Si $i = k$, entonces $q = 1$, pues con el único corte c_1^A alcanza, por lo que $1 = q \leq k$. Si $i < k$, entonces $c_{i+1} - c_1^A \leq \ell$, pues $c_{i+1} - c_i \leq \ell$ por ser C es un esquema de corte válido y $c_i \leq c_1^A$. Luego, podemos obtener un esquema de corte C' que sea $c_1^A, c_{i+1}, \ldots, c_k$ tal que $|C'| \leq k$ y que comparte su primer corte con C^A. Aplicando esta idea sobre la parte derecha del tablón luego de cortar en c_1^A de manera inductiva, podemos deducir que C^A tiene una cantidad de cortes menor o igual que cualquier esquema posible.
Betina se enteró del problema de Aye pero cree que la estrategia para minimizar la cantidad de cortes es distinta:

1. Si el tablón tiene longitud menor o igual a ℓ, no hace nada.
2. En caso contrario, elige las dos marcas con máxima distancia entre ellas sin pasarse de ℓ (es decir, dos marcas distintas a distancia $d \leq \ell$ tal que $\ell - d$ sea mínimo), y corta en ellas.
3. Realiza el mismo procedimiento con los dos tablones restantes a la izquierda y a la derecha del que se acaba de cortar.

Betina también provee una demostración para probar que su estrategia es la correcta: Sea C un esquema de corte cualquiera y C^B el esquema de corte de Betina. Podemos ver que, en el paso k, la cantidad de madera restante a cortar en C es mayor o igual que en C^B: sean $r_1 > \ldots > r_k$ y $r_1^B > \ldots > r_q^B$ las longitudes acumuladas de los tablones que quedan por cortar después de cada corte en el esquema C y C^B respectivamente. En un principio en ambos casos tengo n centímetros para cortar en ambos esquemas. Sean ℓ_C y ℓ_B las longitudes de los primeros tablones que corto a partir del original. Por definición, $\ell_C \leq \ell_B \leq \ell$, así que ciertamente $r_1 \geq r_1^B$, es decir, en el esquema C queda más madera por cortar, la cual claramente requerirá una cantidad mayor o igual de cortes que en el esquema C^B. Aplicando esta idea de manera inductiva sobre los tablones restantes, podemos deducir que C^B tiene una cantidad de cortes menor o igual que cualquier esquema posible.
Tomando como entrada del problema los valores de n y ℓ más la lista M de marcas en el tablón ordenadas de izquierda a derecha, ¿Cuál(es) de estas afirmaciones es(son) verdadera(s)?

1 ☐ La demostración de Betina no muestra que su esquema sea correcto (lo sea o no).
2 ☐ El esquema de Betina se puede obtener en $O(|M|^2)$ (sea correcto o no).
3 ☐ El esquema de Betina produce la solución correcta.
4 ☐ El esquema de Aye se puede obtener en $O(|M|)$ (sea correcto o no).
5 ☐ El esquema de Aye produce la solución correcta.
6 ☐ La demostración de Aye no muestra que su esquema sea correcto (lo sea o no).

Pregunta 8 Sea P_4 el grafo de 4 vértices que forma un camino, i.e. $P_4 = (\{v_1, v_2, v_3, v_4\}, \{v_1v_2, v_2v_3, v_3v_4\})$. Tuki quiere diseñar un algoritmo que detecte, dado un grafo $G = (V, E)$, si G contiene a P_4 como subgrafo inducido*. Para esto, define el siguiente algoritmo:

Algoritmo 1 Procesar grafo

```
1: function DetectarP4(G = (V, E))
2:     for vw ∈ E do
3:         if N(v) \ N(w) ≠ ∅ and N(w) \ N(v) ≠ ∅ then
4:             return True
5:         end if
6:     end for
7:     return False
8: end function
```

Decidir:

1 ☐ Si G viene dado como lista de adyacencias entonces el algoritmo se puede implementar en $O(nm)$.

2 ☐ Si el grafo G tiene un P_4 como subgrafo inducido entonces el algoritmo propuesto por Tuki devuelve `True`.

3 ☐ Si el grafo G no tiene un P_4 como subgrafo inducido entonces el algoritmo propuesto por Tuki devuelve `False`.

4 ☐ Si se modifica la línea 3 para que la condición del **if** sea que $N(v) \cap N(w) \neq \emptyset$, se obtiene un algoritmo correcto para decidir si G posee triángulos.

5 ☐ Si G viene dado como matriz de adyacencias entonces el algoritmo se puede implementar en $O(n^3)$.

*H es subgrafo inducido de $G = (V, E)$ si un conjunto $V_H \subseteq V$ cumple que V_H y todas las aristas que unen vértices de V_H en G conforman un grafo isomorfo a H.

Pregunta 9 Un grafo $G = (V, E)$ es **bipartito** si y solo si V se puede dividir en dos conjuntos disjuntos U y W, de tal manera que cada arista en E conecta un vértice en U con un vértice en W. En tal caso, decimos que G se puede **bipartir** en U y W. Sea G bipartito y U y W una bipartición de G, ¿Cuál(es) de las siguientes afirmaciones es (son) verdadera(s)?

1 ☐ G no contiene ciclos de longitud impar.

2 ☐ G no contiene ciclos de longitud par.

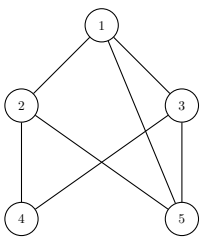
3 ☐ Si aplico DFS desde cualquier vértice v de G, toda *back-edge* siempre unirá dos vértices en niveles del árbol de distinta paridad.

4 ☐ O bien $|U|$ es par o $|W|$ lo es.

5 ☐ El camino más largo de G tiene longitud a lo sumo máx$\{|U|, |W|\}$ aristas.

6 ☐ La suma de los grados de los vértices de U es igual a la suma de los de W.

Pregunta 10 Dado el grafo G de la figura y sabiendo que los vecindarios de los vértices se recorren de manera ordenada **ascendentemente** por etiqueta, ¿Desde qué vértice debe correrse el algoritmo DFS para que la arista $(2, 4)$ sea categorizada como *back-edge*?



1 ☐ 1

2 ☐ 5

3 ☐ 4

4 ☐ 3

☐ 2

6 ☐ Ninguno

Pregunta 11 Para los siguientes problemas, marque la(s) estrategia(s) de recorrido que permitan resolverlos con la menor complejidad temporal e indique cuál es esa complejidad.

- Para un digrafo $D = (V, E)$, determinar el camino con menor cantidad de aristas desde cada uno de los vértices de un conjunto $R \subseteq V$ con $|R| = r$ hasta un vértice v dado, sabiendo que en D todo vértice cumple que $\delta^-(v) + \delta^+(v) \leq k^*$.

1 ☐ DFS
2 ☐ BFS
3 ☐ $O(rn)$
4 ☐ $O(kn)$

- Dado un árbol d-ario T de altura h con raíz conocida r y una función de peso $p\colon V(T) \to \mathbb{N}$, obtener la suma del peso de todos los vértices en el nivel $k \leq h$.

5 ☐ DFS
6 ☐ BFS
7 ☐ $O(dk)$
8 ☐ $O(d^k)$

- Dado un grafo $G = (V, E)$ con $|V| = n$ y $|E| = m$, determinar, para un conjunto $R \subseteq E$ con $|R| = r$ aristas, las aristas de R que pertenecen a al menos un ciclo.

9 ☐ DFS
10 ☐ BFS
11 ☐ $O(m + n)$
12 ☐ $O(rm)$

*Notamos con $\delta^-(v)$ y $\delta^+(v)$ a los grados de entrada y de salida de v, es decir, la cantidad de aristas salientes y entrantes que tiene, respectivamente

Pregunta 12 Considerando que partimos representaciones de un grafo G con vértices $1, \ldots, n$ como lista de adyacencias y matriz de adyacencias (sin estructuras adicionales inicialmente), ¿Cuáles de estas operaciones se pueden realizar en la complejidad indicada?

- Listar el vecindario de un vértice v ordenado por número de vértice en $O(n)$.

1 ☐ Lista de Adyacencias 2 ☐ Matriz de Adyacencias 3 ☐ Ninguna

- Calcular $N(v) - N(w)$ para dos vértices v, w en $O(n)$.

4 ☐ Lista de Adyacencias 5 ☐ Matriz de Adyacencias 6 ☐ Ninguna

- Determinar si un conjunto dado de k vértices es un completo en $O(k^2)$.

7 ☐ Lista de Adyacencias 8 ☐ Matriz de Adyacencias 9 ☐ Ninguna

- Determinar si G tiene un completo inducido de tamaño k en $O(nk^2)$.

10 ☐ Lista de Adyacencias 11 ☐ Matriz de Adyacencias 12 ☐ Ninguna