Técnicas de Diseño de Algoritmos
Primer Cuatrimestre 2024
1er Recuperatorio "B" (T. M.)

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 Revisando los objetos guardados en la baulera de su departamento, Teodoro se encontró con un misterioso cofre, que para ser abierto posee un teclado para ingresar una contraseña. El teclado tiene n símbolos, y por la cantidad de espacios para poner la clave, se deduce que la contraseña buscada también consiste de n símbolos. Antes de ponerse a probar si puede abrir el cofre, Teodoro está planificando su estrategia: piensa intentar combinaciones posibles eligiendo símbolo por símbolo, probando todas las combinaciones que tienen como prefijo a los símbolos ya elegidos. Su hipótesis es que los n símbolos van sin repetir, así que en pocas palabras su estrategia consiste, desde $i = 1$, hacer:

- Elegir un símbolo para la i-ésima posición
- Probar todas las formas de de formar claves de n símbolos donde el prefijo hasta el símbolo i ya está determinado y usando sólo los símbolos que no haya usado en el prefijo.
- Cuando $i = n$, probar si el cofre abre, si no lo hace, ir al último símbolo j donde aún no haya probado todas las posibilidades y elegir una nueva combinación con el prefijo desde la posición j.

Suponiendo que Teodoro utilizara esta estrategia como mecanismo de backtracking, ¿Cuántas hojas tendría el árbol que formaría?

1 ☐ $O(2^n)$ 3 ☒ $O(n!)$
2 ☐ $O(n^2)$ 4 ☐ $O(n^n)$

Aún sin haber comenzado a probar, Teodoro miró con más detenimiento el teclado y descubrió que los botones de ciertos símbolos aún tenían marcas de huellas digitales, indicando (suponiendo que nadie hizo un intento fallido de abrir el cofre) que al parecer la clave de longitud n apenas utiliza 4 símbolos distintos. En función de esto, cambió el diseño de su estrategia, que seguirá tratando de armar claves estableciendo prefijos pero ahora usando sólo estos 4 símbolos (pero permitiendo que se repitan, pues $n > 4$). De cambiar la estrategia de este modo, ¿Cuántas hojas tendría el nuevo árbol de backtracking?

5 ☒ $O(4^n)$ 7 ☐ $O(4 * n!)$
6 ☐ $O(n^4)$ 8 ☐ $O((4n)!)$

En función de sus respuestas anteriores, y sabiendo que $n \geq 10$, ¿con cuál de las estrategias termina quedando el árbol con menor cantidad de hojas?

9 ☐ La primera estrategia 10 ☒ La segunda estrategia 11 ☐ No es posible compararlas

Pregunta 2 Definimos el problema **subset sum minimal**: Dada una lista $A = [A_0, \ldots, A_{n-1}]$ de n valores naturales y un valor fijo $K \geq 0$, queremos saber el mínimo cardinal de un conjunto de índices de elementos de A que sumen exactamente K. Para esto, definimos una función recursiva $f(i, a, c, m)$, que toma un índice i, un acumulador a, una cantidad c y un minimo actual m, y hace lo siguiente:

- Si $i = n$ y $a = K$, entonces devuelve $\min\{c, m\}$.
- Si $i = n$ y $a \neq K$, entonces devuelve m.

- Si $i < n$, entonces devuelve $f(i+1, a, c, m')$ donde $m' = f(i+1, a + A_i, c+1, m)$.

De esta manera, la invocación para resolver el problema es $f(0,0,0,\infty)$. Para cada una de las siguientes propuestas de poda, categorizar según si constituye una poda por factibilidad, una poda por optimalidad, o no es una poda válida para esta formulación.

- Si $a > K$, directamente devolver m.

1 [X] Poda por factibilidad 2 [ ] Poda por optimalidad 3 [ ] No es una poda válida

- Si $a < K$, directamente devolver m.

4 [ ] Poda por factibilidad 5 [ ] Poda por optimalidad 6 [X] No es una poda válida

- Si $a + \sum_{j>i} A_j < K$, directamente devolver m.

7 [X] Poda por factibilidad 8 [ ] Poda por Optimalidad 9 [ ] No es una poda válida

- Si $a + \sum_{j>i} A_j > K$, directamente devolver m.

10 [ ] Poda por factibilidad 11 [ ] Poda por optimalidad 12 [X] No es una poda válida

- Si $a < m$, directamente devolver m.

13 [ ] Poda por Factibilidad 14 [ ] Poda por Optimalidad 15 [X] No es una poda válida

- Si $a > m$, directamente devolver m.

16 [ ] Poda por Factibilidad 17 [ ] Poda por Optimalidad 18 [X] No es una poda válida

Pregunta 3 Supongamos tres vectores de N enteros no negativos $p_1, \ldots, p_N$, $b_1, \ldots, b_N$ y $d_1, \ldots, d_N$, y sea Φ un problema que se puede resolver haciendo el llamado $f(N, K)$ a la función recursiva f definida como

$$f(i,k) = \begin{cases} -\infty & \text{si } k < 0 \\ i & \text{si } i \leq 1 \\ \max\{f(i-1, k-p_i) + f(i-2, k) + b_i, f(i-2, k-p_i) + f(i-1, k) + d_i\} & \text{c.c.} \end{cases}$$

Queremos implementar una solución por medio de programación dinámica para computar f, usando una tabla M como estructura de memoización. Queremos poder acceder a M en $O(1)$ tiempo, pero, una vez calculado el valor de $f(N, K)$, nos interesa también saber cuál de las dos expresiones recursivas resultó en el valor máximo para cada llamado. Con esta restricción, ¿Cuáles son las dimensiones mínimas que podríamos darle a M?

1 [X] $O(N \times K)$ 3 [ ] $O(N^2)$
2 [ ] $O(N)$ 4 [ ] $O(K)$

Pregunta 4 Agustín se fue de viaje, y a la vuelta espera comprar chocolates. 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 chocolates. Denotamos como $p_{i,k}$ a lo que cuesta comprar k chocolates en la ciudad i, para $1 \leq i \leq n$ y $1 \leq k \leq m$, y definimos $p_{i,0} = 0$. Agustín inicia el viaje con un capital P, y su objetivo es maximizar la cantidad de chocolates que tiene al llegar a Buenos Aires. Para resolver este problema propone implementar la siguiente función recursiva:

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

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

1 [X] La evaluación de cada nodo que no es caso base en el árbol de recursión tiene costo $O(m)$.

2 [X] La función f está bien definida y el llamado propuesto resuelve el problema pero su semántica no corresponde con la que dice Agustín.

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

4 [X] La implementación de la solución de Agustín basada en programación dinámica, que toma $p_{i,k}$ con $0 < i \leq n$, $1 \leq k \leq m$ y el valor P, tiene complejidad temporal pseudopolinomial respecto al tamaño de su entrada.

5 [X] La estrategia *bottom-up* que vimos en la materia permite una implementación que consume $O(P)$ espacio.

6 [ ] La evaluación de cada nodo que no es caso base en el árbol de recursión tiene costo $O(n)$.

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

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

9 [ ] Independientemente de n y P, este problema siempre presenta el fenómeno de superposición de subproblemas.

Pregunta 5 Matías es un niño que adora hacer travesuras. Un día, durante el recreo, Matías 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 un 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 tiempo $O(1)$ 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 [ ] $T(n) = 4T(n/4) + O(1)$

2 [ ] $T(n) = T(n/2) + O(n)$

3 [ ] $T(n) = T(n/4) + O(1)$

4 [X] $T(n) = T(n/2) + O(1)$

5 [ ] $O(n^2)$

6 [ ] $O(n)$

7 [ ] $O(n \log n)$

8 [X] $O(\log n)$

Pregunta 6 Para las siguientes recurrencias marque el orden de complejidad más ajustado.

- $T(n) = 8T(n/3) + n$

1 ☐ $O(n)$
2 ☐ $O(\log n)$
3 ☐ $O(n \log n)$
4 ☐ $O(n^3)$
5 ☒ $O(n^{\log_3 8})$
6 ☐ $O(n^2)$

- $T(n) = 7T(n/3) + n^2$

7 ☐ $O(n^3)$
8 ☐ $O(n)$
9 ☐ $O(n \log n)$
10 ☐ $O(n^2 \log n)$
11 ☐ $O(n^{\log_3 7})$
12 ☒ $O(n^2)$

- $T(n) = 16T(n/4) + n^2$

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

- $T(n) = 9T(n/3) + \log n$

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

Pregunta 7 En el **problema del cambio**, tenemos una suma n de dinero que queremos formar con la menor cantidad de monedas (o, en este ejemplo, billetes) posible. Consideremos que las denominaciones de billetes con las que contamos son 1000, 500 y 100. Podemos considerar una estrategia greedy para estipular los billetes que vamos a usar para sumar n (suponiendo a n múltiplo de 100), donde usamos la mayor cantidad de billetes de 1000 que podamos, y para el valor menor a 1000 restante usamos todos los billetes de 500 que podamos seguidos de los billetes de 100 que necesitemos para completar el cambio.

Podemos argumentar que esta estrategia es correcta presentando la siguiente demostración: Sea n el valor que queremos cambiar y sean (m_g, q_g, c_g) la cantidad de billetes de mil, de quinientos y de cien que usamos para el cambio en la estrategia greedy $\mathcal{G}$, y (m, q, c) las cantidades que usamos para el cambio en una estrategia cualquiera $\mathcal{E}$. Ciertamente, $m_g \geq m$, pues por ser estrategia greedy usará todos los billetes de 1000 posibles. Supongamos que $m_g > m$, es decir, que $\mathcal{G}$ utiliza más billetes de mil que $\mathcal{E}$. Notar que, entonces, $(500q+100c)-(500q_g+100c_g) \geq 1000$ pues $m_g+q_g+c_g = m+q+c = n$. Luego, o bien $q-q_g \geq 2$, o $q-q_g = 1$ y $c-c_g \geq 5$, o $q-q_g = 0$ y $c-c_g \geq 10$. Notar que en estos 3 escenarios es sencillo cambiar, respectivamente, 2 billetes de 500 por uno de 1000, o uno de 500 y 5 de 100 por uno de 1000, o 10 de 100 por uno de 1000. En todo caso, observamos que $\mathcal{G}$ utiliza menos billetes que $\mathcal{E}$; luego $\mathcal{G}$ usa la misma cantidad de billetes de 1000 que toda estrategia óptima. Podemos hacer un razonamiento similar con q_g y q; si suponemos que $m_g = m$, volvemos a poder afirmar que $q_g \geq q$, lo cual nuevamente nos lleva a que si $q_g > q$, entonces $c - c_g \geq 5$ y por lo tanto $\mathcal{G}$ utiliza menos billetes en total que $\mathcal{E}$. En consecuencia, la cantidad de billetes total usada en $\mathcal{G}$ es a lo sumo la misma que en $\mathcal{E}$, mostrando que la estrategia greedy da el cambio utilizando una cantidad mínima de billetes. En base a este argumento, decidir:

1 ☐ Usando argumentos más sofisticados se puede mostrar que esta estrategia funciona para billetes de cualquier denominación.
2 ☐ Usando una lógica similar se puede probar que la estrategia greedy descrita también funciona con los billetes de 2500, 1000, y 100.
3 ☐ La estrategia funciona, pero el argumento presentado es incorrecto.
4 ☒ El argumento es correcto.

5 ☒ Usando una lógica similar se puede probar que la estrategia greedy descrita también funciona con los billetes de 2500, 500, y 100.

¿Cuál es la complejidad más ajustada para aplicar este algoritmo (con las denominaciones originales), suponiendo operaciones aritméticas constantes?

6 ☐ $O(n \log n)$ 7 ☒ $O(1)$ 8 ☐ $O(n)$

Pregunta 8 Estela acaba de salir de la oficina a almorzar y se dirige a su lugar de comida por peso preferido, mientras piensa qué se va a servir. En este lugar hay una variedad de n platos de los cuales la gente se sirve en una bandeja y luego, según el peso de la bandeja, se les cobra. Estela mira en su teléfono una tabla de referencia que dice el peso p_i de una porción del plato i ($1 \leq i \leq n$) e imagina un valor g_i que corresponde con cuánto le gusta ese plato. Luego, mira su billetera y llega a la conclusión de que puede pagar una bandeja de peso a lo sumo P. Sabiendo que el lugar no permite servirse en la bandeja más de una porción de un mismo plato (pero sí admite servirse una fracción de porción, en ese caso tanto el peso como el gusto son proporcionales a la fracción servida), Estela se pregunta cuál es la máxima cantidad de gusto que puede llegar a obtener de una bandeja de peso a lo sumo P con los platos disponibles. Ella está convencida de que existe una estrategia greedy que la puede ayudar a conocer este valor, pero no está segura de cuál es.

¿Cuál(es) de estas estrategias greedy maximiza(n) el gusto que puede obtener Estela?

1 ☐ Calcular, para cada plato i, el cociente $q_i = \frac{p_i}{g_i}$. Luego, ir poniendo en la bandeja una porción de cada plato ordenando descendentemente por cociente q_i, y cuando el siguiente plato no entre, poner la fracción que entre.

2 ☐ Calcular, para cada plato i, el promedio $m_i = \frac{p_i+g_i}{2}$. Luego, ir poniendo en la bandeja una porción de cada plato ordenando descendentemente por promedio m_i, y cuando el siguiente plato no entre, poner la fracción que entre.

3 ☐ Ir poniendo en la bandeja una porción de cada plato ordenando descendentemente por gusto, y cuando el siguiente plato no entre, poner la fracción que entre.

4 ☐ Ir poniendo en la bandeja una porción de cada plato ordenando ascendentemente por peso, y cuando el siguiente plato no entre, poner la fracción que entre.

5 ☒ Calcular, para cada plato i, el cociente $c_i = \frac{g_i}{p_i}$. Luego, ir poniendo en la bandeja una porción de cada plato ordenando descendentemente por cociente c_i, y cuando el siguiente plato no entre, poner la fracción que entre.

Supongamos que una nueva regla en este lugar indica que ya no es posible servirse una fracción de un plato, se sirve una porción o no se sirve. ¿Sigue funcionando la estrategia elegida?

6 ☐ Sí. 7 ☒ No.

Pregunta 9 Contamos con una demostración de la afirmación "todo grafo conexo con $n \geq 2$ vértices posee dos vértices distintos v_1, v_2 tales que tanto $G \setminus v_1$ como $G \setminus v_2$ son conexos", por inducción global en $|V(G)|$:

Caso base: Para $|V(G)| = 2$, la afirmación es verdadera (un vértice aislado es conexo).

Paso inductivo: Sea G un grafo conexo con $|V(G)| > 2$. Si para todo vértice v de G vale que $G \setminus v$ es conexo, entonces en particular hay 2 vértices que al removerlos el grafo permanece conexo y la propiedad está probada. Sea v entonces tal que $G \setminus v$ no es conexo, y sean $H_1, \ldots, H_k$ ($k > 1$) las componentes conexas de $G \setminus v$. Para cualquier $i \in \{1, \ldots, k\}$, consideremos el subgrafo G'_i inducido por $H_i \cup v$. Es claro que $|V(G'_i)| < |V(G)|$, y más aún, G'_i es conexo. Luego, podemos aplicar hipótesis inductiva y afirmar que G'_i posee dos vértices distintos v_1^i y v_2^i tales que tanto $G'_i \setminus v_1^i$ como $G'_i \setminus v_2^i$ son conexos. Ciertamente uno de v_1^i o v_2^i no es v, supongamos que es v_1^i. Notar que si quitamos v_1^i de G, G sigue siendo conexo, pues para todo par de vértices w_1, w_2 de G: o bien ambos pertenecen a H_i, y tienen un camino que los conecta por definición de v_1^i, o w_1 es de H_i y w_2 no, pero entonces también tienen un camino porque se puede ir desde w_1 hasta v y desde v hasta w_2, o si no ni w_1 ni w_2 pertenecen a H_i y por lo tanto la remoción de v_1^i no cambia el camino que los conecta. Como $i \geq 2$, tenemos al menos los vértices v_1^1 y v_1^2 que cumplen que $G \setminus v_1^1$ es conexo y $G \setminus v_1^2$ es conexo, probando la afirmación.

¿Son la afirmación y su demostración correctas?

1 ☐ La demostración no es correcta porque el paso inductivo no está correctamente planteado.

2 ☒ La demostración es correcta.

3 ☐ La demostración no es correcta porque el paso inductivo requiere que el caso base sea distinto al elegido o se demuestren más casos base.

4 ☒ La afirmación es verdadera.

5 ☐ La afirmación no es verdadera.

Pregunta 10 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 para cada representación?

- Dados dos vértices v, w, determinar si en G hay un camino de v a w en $O(n+m)$.

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

- Calcular $N(v) \cap N(w)$ 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 conjunto independiente en $O(k^2)$.

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

- Listar todos los conjuntos independientes de G tamaño en $O(n^3)$.

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

Pregunta 11 Determine cuáles de estas afirmaciones respecto a árboles BFS y DFS son verdaderas:

1 ☒ En un grafo no dirigido G, la cantidad de back-edges (aristas fuera del árbol entre ancestro y descendiente) de cualquier árbol DFS y de cross-edges (aristas fuera del árbol entre vértices de ramas distintas) de cualquier árbol BFS es la misma, independientemente de qué vértices se elija para comenzar cada uno de los recorridos.

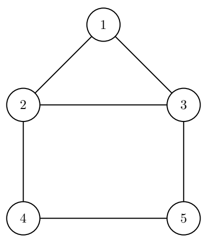
2 ☐ En BFS, toda arista no perteneciente al árbol es una arista entre dos vértices del mismo nivel en el arbol.

3 ☒ En DFS, toda arista no perteneciente al árbol es una arista entre dos vértices de niveles distintos en el árbol.

4 ☒ Si la ejecución tanto de BFS como de DFS de un grafo G derivan en el mismo árbol, entonces G es un árbol.

5 ☒ Tanto en BFS como en DFS vale que los puentes del grafo siempre son aristas del árbol.

Pregunta 12 Dado el grafo G de la figura y sabiendo que los vecindarios de los vértices se recorren de manera ordenada **ascendentemente** por etiqueta, ¿Cuál(es) arista(s) siempre son tree-edges del árbol DFS de G, independientemente de qué vértice se comience?



1 ☒ $(1,2)$

2 ☐ $(2,3)$

3 ☐ $(1,3)$

4 ☐ $(2,4)$

5 ☐ Ninguna

6 ☐ $(3,5)$

7 ☐ $(4,5)$