Técnicas de Diseño de Algoritmos
Algoritmos y Estructuras de Datos III

Primer Parcial - 2do Cuatrimestre 2024

Nombre y Apellido	LU	Nº orden	# hojas	Turno
.....	.....	.....	.....	.....

Duración : 4 horas
*Este examen es a **libro cerrado**.*

1. Preguntas opción múltiple (40 ptos.)

Las preguntas pueden tener cero, una, o varias respuestas correctas.

Pregunta 1 Tania tiene un arreglo de números, y quiere encontrar el subarreglo consecutivo más largo que repita siempre el mismo caracter. Propone el siguiente algoritmo de divide and conquer para resolver su problema:

1. Si el arreglo tiene un solo elemento, devolver el número 1.
2. Sino:
   a) Separar el arreglo en dos partes iguales.
   b) Llamar recursivamente al algoritmo en cada parte, obteniendo el subarreglo consecutivo más largo de cada parte.
   c) Devolver la suma de los dos.

¿Cuáles de las siguientes afirmaciones son verdaderas sobre el algoritmo de Tania?

- ☐ Tiene complejidad temporal más ajustada de $O(n^2)$.
- ☐ Tiene complejidad temporal más ajustada de $O(n)$.
- ☐ Tiene complejidad temporal más ajustada de $O(n \log(n))$.
- ☐ Resuelve correctamente el problema.

Pregunta 2
Sasha empezó a trabajar *fulltime* y se encuentra sobrepasado por la cantidad de tareas que debe realizar. Cada tarea $1 \leq i \leq n$ tiene un tiempo l_i que tarda en realizarla, junto a un peso w_i que denota la penalización en que incurre por cada minuto que pasa sin tenerla terminada. Sasha observó que dependiendo el orden en que haga las tareas la penalización final (i.e. la suma de las penalizaciones) cambia. A modo de ejemplo, supongamos que tiene dos tareas $\{(2,1),(1,4)\}$, donde la primera componente del par indica la duración y la segunda la penalización. Si realiza primero la tarea 1 y luego la 2 incurre en una penalización final de 14: la primera tarea la termina en la unidad de tiempo 2, incurriendo en una penalización de 2, mientras que la segunda la termina en la unidad 3, con un costo de 12. Por otro lado, si hace primero la segunda la penalización final es 7.
Sasha propuso distintas estrategias greedy para elegir qué tarea hacer a continuación, con el objetivo final de minimizar la penalización total en la que va a incurrir. Indicar cuáles son correctas:

- ☐ Hacer alguna de las tareas que maximice $w_i - l_i$.
- ☐ Hacer alguna de las tareas que maximice w_i.
- ☐ Hacer alguna de las tareas que minimice l_i.
- ☐ Hacer alguna de las tareas que maximice $\frac{w_i}{l_i}$.

Pregunta 3

Julieta estudia sociología, y está preparando el último final de la carrera. Para el mismo, debe leer una lista de n libros, y como está atrasada y no llega a leerlos todos, se propuso seleccionar un subconjunto de los mismos. Para hacer esto los ordenó cronológicamente por fecha de publicación en un vector l y se dispuso a escribir un algoritmo de backtracking que le permita imprimir en pantalla posibles subconjuntos de lectura que resultarían útiles. Está interesada en cumplir dos condiciones: debe elegir por lo menos $0 \leq k \leq n$ libros, y cada par de libros debe estar a distancia por lo menos t en orden cronológico. Propuso la siguiente función recursiva para resolver el problema:

```
procedure LIBROS(sol, i, t, k)
    if i = n then
        print(sol)
    else
        if ¬seleccionado_en_ultimos(sol, i, t) then
            LIBROS(sol, i + 1, t, k)
            sol ⊕ l_i
            LIBROS(sol, i + 1, t, k)
            sol ⊖ l_i
        else
            LIBROS(sol, i + 1, t, k)
        end if
    end if
end procedure
```

La solución `sol` la representa con un vector, e implementa la operación de agregado al final $\oplus$ como un `push_back`, y la de eliminado al final $\ominus$ como un `pop_back`. Para mplementar `seleccionado_en_ultimos` Julieta recorre las posiciones $i - t + 1, \ldots, i - 1$ del vector l revisando si alguno estos elementos está al final del vector `sol`, devolviendo `True` si es así. Finalmente, la función `print` imprime en consola, y la cantidad de operaciones que realiza es proporcional a la longitud de lo que se solicita imprimir. Piensa llamar la función como LIBROS$(\emptyset, 0, t, k)$.
Marcar las correctas:

- ☐ Existen valores para l, k y t que hacen que el árbol de recursión tenga 2^n hojas.
- ☐ La función recursiva propuesta por Julieta es correcta.
- ☐ La complejidad temporal del algoritmo puede acotarse por $O(2^n)$.

Julieta observó que puede mejorar su algoritmo: en vez de recorrer el vector para ver si se eligió uno de los últimos t puede mantener el índice del último elegido. Teniendo eso en cuenta, propone una nueva recursión:

```
procedure LIBROS(sol, i, ult, t, k)
    if i = n then
        print(sol)
    else
        if ult ≤ i − t then
            LIBROS(sol, i + 1, ult, t, k)
            sol ⊕ l_i
            LIBROS(sol, i + 1, i, t, k)
            sol ⊖ l_i
        else
            LIBROS(sol, i + 1, ult, t, k)
        end if
    end if
end procedure
```

Piensa llamar a su función como LIBROS$(\emptyset, 0, -t)$. Decidir:

- ☐ La cantidad de hojas del árbol de recursión es la misma con la nueva función.
- ☐ La complejidad temporal correspondiente al procesamiento de un nodo interno del árbol mejoró.
- ☐ La complejidad temporal de este nuevo algoritmo se puede acotar por $O(2^n)$.

Pregunta 4 Un *bosque* es un grafo sin ciclos (no necesariamente conexo). Supongamos que tenemos un bosque F y le agregamos dos aristas cualesquiera, obteniendo F'. Marcar las afirmaciones verdaderas.

- ☐ La sumatoria de los grados de los vértices de F' es menor o igual a $2 \cdot (n + 1)$.
- ☐ Si usamos $\mathsf{diam}(\mathsf{F}')$ para denotar la máxima distancia entre dos vértices conectados de F' (el "diámetro"), tenemos que $\mathsf{diam}(\mathsf{F}) < \mathsf{diam}(\mathsf{F}')$.
- ☐ F' es fuertemente conexo.
- ☐ F' no puede tener más de dos ciclos.

2. Ejercicios a desarrollar

2.1. Octavio y el esquí

A Octavio le encanta esquiar, y este año decidió competir en el TAP (Torneo Anual de Pekín). En el mismo se debe descender una montaña de N metros de alto, en una pista que se puede representar como una grilla cuadrada de dimensiones $N \times N$. Los participantes empiezan el recorrido en el centro de de la pista, a la máxima altura posible (que se corresponde con el casillero $(1, \frac{N+1}{2})$).[1]. A medida que desciende Octavio puede moverse a cualquiera de las posiciones que están a menos de k de distancia de su posición actual, en el eje x. Es decir, si está en la posición (i, j) entonces puede desplazarse a cualquiera de las posiciones $(i+1, j-k), (i+1, j-k+1), \ldots, (i+1, j+k-1), (i+1, j+k)$.

Octavio gana puntos a medida que recorre la montaña. Específicamente, se sabe el puntaje $P_{i,j}$ que él recibe por recorrer la posición (i, j) de la pista. Por cada casillero se pueden ganar a lo sumo 10 puntos (es decir, $1 \leq P_{i,j} \leq 10$ para todo $1 \leq i, j \leq n$). Sin embargo, hay algunas posiciones de la misma en que hay huecos, y en caso de entrar en contacto con ellas los puntos que haya acumulado hasta ese momento Octavio se dividen por dos.

Finalmente, en algunas posiciones de la pista hay *rampas*, las cuales puede usar para saltar y hacer trucos. Cada vez que toma una rampa su puntaje se incrementa en 100. Cuando se toma una rampa se avanzan L casilleros en dirección recta. Es decir, al tomar una rampa en la posición (i, j) se pasa directamente a la posición $(i + L, j)$, sin pasar por ninguna de las posiciones intermedias.

Octavio quiere calcular la máxima cantidad de puntos que puede obtener en esta competencia, considerando que inicia su trayecto en la posición $(1, \frac{N+1}{2})$ y que puede terminar en cualquier posición de la base de la montaña (es decir, cualquier posición de la forma (N, j) con $1 \leq j \leq N$). La entrada al problema consiste del valor N, el valor k, la matriz P de dimensiones $N \times N$, la lista de posiciones con huecos H, la lista de posiciones con rampas R, y el valor L.

a) Definir una función recursiva que sirva para resolver el problema de Octavio. Dar una cota inferior a su complejidad temporal de peor caso que sea exponencial en n, asumiendo que se la implementa sin ningún tipo de memorización. Indicar de qué formas se debe llamar a la función para obtener la respuesta al problema. **Sugerencia:** Proponer una función de la forma $f(i, j)$ que denote "El mayor puntaje al que puede llegar Octavio si comienza su recorrido en la posición $\left(1, \frac{N+1}{2}\right)$ y lo termina en la posición (i, j)".

b) Indicar cuál es la complejidad temporal del algoritmo que implementa la función del inciso anterior empleando programación dinámica. Justificar detalladamente qué estructuras se necesitan para alcanzar la complejidad esperada ¿Cuándo conviene emplear el algoritmo basado en programación dinámica?

c) Dar el pseudocódigo que implementa el algoritmo del inciso *b)*. Indicar qué estructuras son necesarias y cómo se las usa.

Criterio de aprobación: El algoritmo basado en programación dinámica debe tener una complejidad temporal $O(kN^2)$.

[1] Asumimos que N es impar.

2.2. Rabeirou y sus amigos

Rabeirou tiene un grupo de amigos en los que, como es de esperar, algunos son más amigos que otros. Rabeirou notó que para cada par de amigos en su grupo podía clasificar la relación entre ellos en dos categorías: o eran amigos *posta*, o eran amigos *casuales*. Se dio cuenta que los que eran amigos casuales eran amigos porque el primero era amigo posta de alguno, ese otro era amigo posta de otro... y así hasta llegar al último. En otras palabras, dos amigos son casuales si y sólo si no son amigos posta y hay una secuencia de amistades posta que llevan del primer amigo al último. Por ejemplo, si Alejandra es amiga posta de Brian, Brian es amigo posta de Cecilia, y Alejandra no es amiga posta de Cecilia, Alejandra es amiga casual de Cecilia.

Un día viene Daferade y, por envidioso, decide romperle la amistad posta entre dos amigos del grupo de Rabeirou, pero se dio cuenta que sin importar qué amistad posta rompiera, habría algunos pares de amigos (casuales o posta) para los que nunca podría romper la amistad. A estos los llamó *inseparables*. En otras palabras, dos amigos son inseparables si y sólo si, sin importar qué única amistad posta se rompa, de todas formas siguen siendo amigos (casuales o posta).

a) Dar un algoritmo de complejidad temporal $O(n + m)$ que determine la cantidad de pares diferentes[2] de amigos que son inseparables. Justificar su correctitud y complejidad.

Rabeirou se enteró de los planes de Daferade, y quiere asegurarse de no separarse de uno de sus amigos, llamado obocanegra, porque tiene pileta y se viene el verano. Desesperado por la situación, Rabeirou va a esforzarse para que dos amigos de su grupo se vuelvan amigos posta antes de que Daferade haga quilombo, y así lograr que él y obocanegra se vuelvan inseparables. Lamentablemente no tiene disponibilidad para reforzar la amistad de cualquier par de amigos, sino que tiene un conjunto C de pares de personas de los que puede elegir un solo par para convertirlo en un par de amigos posta.

b) Dar un algoritmo de complejidad temporal $O(n + m + |C|)$ que, dado el conjunto C de pares de amigos, determine si Rabeirou puede volver inseparable su amistad con obocanegra. Justificar su correctitud y complejidad.

[2]Dos pares de amigos son iguales si tienen los mismos dos elementos, sin importar el orden.