ALGORITMOS Y ESTRUCTURAS DE DATOS III - 1^{er} Parcial
Fecha de examen: 30-SEP-2022

	Nº Orden	Apellido y nombre		L.U.	# hojas [1]
Notas:	Ej1	Ej2	Ej3	Ej4	Final

Aclaraciones: El parcial se aprueba con 2 (dos) ejercicios aprobados. Cada hoja debe estar numerada y debe tener el número de orden y L.U. El parcial dura 4 horas y es a libro abierto.

1) Tuki está escribiendo un programa que imprime cadenas de n caracteres. Como es supersticioso, le gustaría evitar que el número 13 aparezca en su cadena más de v veces. A su vez, quiere que el 7, el número de la suerte, aparezca al menos k veces.

   Como todavía no programa muy bien no sabe cómo incorporar estas condiciones a su código, y prefiere estimar cuál es la probabilidad de que una cadena aleatoria sea de su agrado. Para eso, tenemos que ayudarlo a escribir una función que calcule cuántas cadenas de longitud n compuestas solo por los t caracteres permitidos por el lenguaje de programación de Tuki (que incluye a los dígitos de 0 a 9) cumplen las condiciones que él pide.

   a) Definir en forma recursiva la funcion $f\colon \mathbb{N}^3 \times \{\text{true}, \text{false}\} \to \mathbb{N}$ tal que $f(i, v, k, \mathit{uno})$ devuelve la cantidad de cadenas s tales que $|s| = i$, s solo contiene caracteres válidos dentro de los t permitidos y la cadena que resulta de agregar un caracter válido x a la izquierda de s cumple que tiene a lo sumo v subcadenas 13 y al menos k subcadenas 7, donde $x = 1$ si $\mathit{uno} = \text{true}$ y $x = 0$ en caso caso contrario. Indicar qué llamado(s) hay que hacer a esta función para resolver el problema.

   b) Demostrar que f tiene la propiedad de superposición de subproblemas.

   c) Definir un algoritmo *top-down* para calcular $f(i, v, k, \mathit{uno})$ indicando claramente las estructuras de datos utilizadas y la complejidad resultante.

   d) Escribir el (pseudo-)código del algoritmo top-down resultante.

   Complejidad: la complejidad temporal del algoritmo resultante para computar $f(i, v, k, \mathit{uno})$ debe ser $O(i \min\{v, i\} \min\{k, i\})$.

2) Un *modelo de intervalos* es una secuencia $\mathcal{I} = [s_1, t_1], \ldots, [s_n, t_n]$ de intervalos cerrados tales que $0 \leq s_1 \leq s_2 \leq \ldots \leq s_n$. El *grafo de intervalos* de $\mathcal{I}$ es el grafo $G(\mathcal{I})$ con n vertices $v_1, \ldots, v_n$ tal que v_i y v_j son adyacentes si y solo si $t_i \geq s_j$ para todo $1 \leq i < j \leq n$.

   a) Proponer un algoritmo goloso que, dado un modelo de intervalos $\mathcal{I}$ cuyo grafo $G(\mathcal{I})$ es conexo, encuentre un árbol v_1-geodésico de $G(\mathcal{I})$. Recordar que un árbol T es v_1-geodésico cuando la distancia entre v_1 a v_i en T es igual a la distancia entre v_1 y v_i en G para todo $1 \leq i \leq n$. **Ayuda:** recordar el trabajo práctico, observando qué propiedad cumplen los intervalos correspondientes a cualquier camino de v_1 a v_i.

   b) Demostrar que el algoritmo propuesto es correcto. **Ayuda:** recordar el trabajo práctico.

   Complejidad: la complejidad temporal del algoritmo resultante debe ser $O(n)$.

3) Decimos que un grafo pesado G es un *árbol enredado* si existe un ciclo C de 3 vértices tal que $G - E(C)$ (i.e., el grafo que resulta de sacarle a G las aristas de C) es árbol generador mínimo de G. Decimos que el ciclo C es un *nudo* de G.

   a) Mostrar un árbol enredado G para el cual alguna ejecución del algoritmo de Kruskal encuentra un AGM T' tal que las aristas en $G - E(T')$ no forman un nudo de G.

[1] Incluyendo esta hoja.

b) Sea X un árbol generador cualquiera de un árbol enredado G que tiene un nudo C. Demostrar que al menos una de las aristas de $G - E(X)$ pertenece a C, cualquiera sea el nudo C.

c) Dar un algoritmo para encontrar un nudo de G y su correspondiente AGM $T = G - E(C)$. **Sugerencia:** usar el item anterior para determinar aristas candidatas de C. ¿Cuántos candidatas puede haber?

Complejidad: El mejor algoritmo que conocemos para encontrar un nudo tiene complejidad temporal $O(n)$. El algoritmo propuesto debe tener complejidad temporal $O(n^2)$.[2]

4) Sea T un árbol y h_v su altura cuando T está enraizado en $v \in V(T)$.

a) Dar un algoritmo eficiente que calcule la distancia entre todo par de vértices de T.

b) Demostrar que si T está enraizado en v, entonces todo vértice w a distancia h_v de v pertenece a un camino de longitud máxima de T, independientemente de cuál sea la raíz v. **Ayuda:** Considere un camino de longitud máxima $x \rightsquigarrow y$ con $x, y \neq w$ y trate de construir otro camino máximo que empiece en w.

c) Se define el *diámetro* de T como la longitud del camino más largo entre dos vértices de T. Definir un algoritmo que, dado un árbol T, determine el diámetro d de T junto con dos vértices cuya distancia en T sea d.

Complejidad: el algoritmo del inciso *a*) debe tener complejidad temporal $O(n^2)$ y el del inciso *c*), $O(n)$.

[2]Recordar que $O(n) \subset O(n^2)$, con lo cual el algoritmo propuesto puede tener complejidad $O(n)$.