Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI5651 – Diseño de Algoritmos I
Septiembre–Diciembre 2025

Tarea 4

A continuación encontrará 4 preguntas, cada una dirá cuántos puntos vale en su preámbulo. Sea lo más detallado y preciso posible en sus razonamientos, algoritmos y demostraciones.

Además del informe expresando su solución, debe dar una implementación de su solución en el lenguaje de su elección (solamente como una función; el formato de entrada/salida no es relevante), para las preguntas 2 y 4.

La entrega se realizará únicamente por correo electrónico a rmonascal@gmail.com.

Fecha de entrega: Hasta las 11:59pm. VET del **Lunes, 20 de Octubre** (*Semana 5*).

1. (2 puntos) – Considere el algoritmo de *Programación Dinámica* propuesto en clase para el problema de distancia de edición entre dos cadenas de caracteres.

   Investigue el día de la semana en la que nació (puede usar un calendario online para esto).

   Construya la tabla correspondiente al proceso para decidir la distancia de edición entre su **día de nacimiento** y su **mes de nacimiento**, *sin ahorro de memoria*.

   Por ejemplo, si nació un *viernes* durante el mes de *marzo*, se desea que construya la tabla 7×5 para la distancia de edición entre `viernes` y `marzo`.

2. (2 puntos) – Sea $A[1..n]$ un arreglo de enteros positivos.

   Dado un $s > 0$, decimos que dos subarreglos contiguos de $s + 1$ elementos, $A[i..i + s]$ y $A[j..j + s]$, son *familiares* si se cumplen con las siguientes condiciones:

   - Los subarreglos son disjuntos (no comparten índices)
   - Uno a uno, cada par de posiciones debe tener elementos coprimos. Esto es:
     - $A[i]$ debe ser coprimo con $A[j]$
     - $A[i + 1]$ debe ser coprimo con $A[j + 1]$
     - $\cdots$
     - $A[i + s]$ debe ser coprimo con $A[j + s]$

   Nota: Recuerde que dos números son coprimos *si su máximo común divisor es 1.*

   Por ejemplo, considere el arreglo $A = [3, 10, 1, 8, 4, 5, 3, 6, 9, 2]$:

   - Los subarreglos $A[1..4] = [3, 10, 1, 8]$ y $A[6..9] = [5, 3, 6, 9]$ son *familiares* dado que: 3 y 5 son coprimos, 10 y 3 son coprimos, 1 y 6 son coprimos, 8 y 9 son coprimos.
   - Los subarreglos $A[1..5] = [3, 10, 1, 8, 4]$ y $A[6..10] = [5, 3, 6, 9, 2]$ **no** son *familiares* dado que 4 y 2 **no** son coprimos.

   Se desea que diseñe un algoritmo que tome tiempo $O(n^2)$ y memoria adicional $O(n)$ que halle la longitud del par de subarreglos *familiares* más largos. Para el ejemplo anterior, la respuesta sería 4.

   Nota: Puede suponer que todas las operaciones aritméticas, incluyendo multiplicaciones, divisiones y módulos, se hacen en $O(1)$.

3. (2 puntos) – Se desea que implemente, en el lenguaje de su elección, un cliente para probar el proceso de inicialización virtual de arreglos. Su programa debe cumplir con las siguientes características:

   - Al invocarse, debe recibir *como argumento del sistema* el tamaño del arreglo a utilizar.
   - El arreglo será indexado a cero (las posiciones válidas, para un arreglo de tamaño n, serán desde la 0 hasta la $n - 1$).
   - Debe presentar al usuario un cliente con cuatro opciones:
     - `ASIGNAR POS VAL`, que tiene el efecto de asignar el valor `VAL` en la posición `POS` del arreglo.
       Su programa debe reportar un error si la posición `POS` no es una posición válida del arreglo.
     - `CONSULTAR POS`, que debe reportar si la posición `POS` está inicializada o no. En caso de estar inicializada, debe devolver el valor asociado a esa posición.
       Su programa debe reportar un error si la posición `POS` no es una posición válida del arreglo.
     - `LIMPIAR`, que tiene el efecto de limpiar la tabla y hacer que **todas** las posiciones queden sin inicializar.
     - `SALIR`, que sale del programa.
   - Todas las operaciones involucradas en su programa deben tomar tiempo $\Theta(1)$.
   - Debe implementar estas operaciones siguiendo el proceso de inicialización virtual visto en clase, usando los dos arreglos auxiliares, **no** con métodos alternativos (como tablas de hash o tipos conjuntos de la librerías de su lenguaje de escogencia).

4. (3 puntos) – ¡Esta aerolinea es un desastre! El transporte automatizado de equipaje se ha descompuesto y ha dejado las maletas de los pasajeros regadas por toda la pista. Es tu trabajo volver a recogerlas todas y colocarlas en el avión. Pero, ¡de prisa! Mientras mas tiempo tomes, más se retrasará el vuelo en salir.

   Decides hacer una lista de todas las cosas que sabes, para organizarte mejor:

   - La cantidad, n, de equipajes que se han caído. También sabes que $1 \leq n \leq 24$.
   - La posición (x, y) de cada equipaje. Además, dadas las dimensiones del aeropuerto, sabes que $|x| \leq 100$ y $|y| \leq 100$ y que ambas dimensiones son **números enteros**.
   - El avión está en la posición $(0, 0)$ y es a donde quieres llevar todas las maletas que se han dispersado.
   - Transitar entre dos puntos a y b, toma tanto tiempo como **el cuadrado de la distancia cartesiana** entre ellos.
   - Solamente te puedes detener en la posición de una maleta o en la del avión, por temor a que te multen las autoridades del aeropuerto.
   - Puedes cargar a lo sumo dos maletas a la vez (una en cada mano) y una vez que tomas una maleta, solamente puedes soltarla en el avión.

   ¿Cuál es la mínima cantidad de tiempo necesaria para recoger todas las maletas?

   Se desea que diseñe un algoritmo usando *Programación Dinámica*, que resuelva este problema en tiempo y memoria $O(n \times 2^n)$.

   Nota: Puede suponer que todas las operaciones aritméticas, incluyendo multiplicaciones, divisiones y módulos se hacen en $O(1)$.

Ricardo Monascal / Octubre 2025