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

Tarea 6

A continuación encontrará 3 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 todas las preguntas.

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

Fecha de entrega: Hasta las 11:59pm. VET del **Lunes, 10 de Noviembre** (*Semana 8*).

1. (3 puntos) – Considere un arreglo `A[1..N]`, que representa una permutación de los números de 1 a N.

   Se desea que ejecute N acciones de la forma `multiswap(a, b)`. Esta acción consite en:

   (a) Intercambiar el valor en la posición a con el de la posición b.

   (b) Invocar `multiswap(a+1,b+1)`

   (c) El proceso termina cuando b sale del rango del arreglo o a alcanza el primer valor de b utilizado.

   A continuación se presenta una implementación en pseudo–Python para `multiswap(a,b)`:

```
def multiswap(A, a, b):
    i, j = a, b
    while i < b and j <= N:
        swap(A, i, j)
        i += 1
        j += 1
```

   Si A inicia como la permutación identidad (números del 1 al N, de menor a mayor) y se ejecutan N operaciones `multiswap(ai, bi)` (donde los valores de ai y bi vienen dados en una lista de tuplas), se desea que imprima el arreglo resultante.

   Diseñe un algoritmo que pueda ejecutar esta acción en tiempo promedio $O(N \log N)$, usando memoria adicional $O(N)$.

   Pistas:

   - Una estructura de datos que permita *dividir* o *reunir* subarreglos eficientemente con una *alta probabilidad*, puede llevar al camino del bien.
   - ¿Realmente hay que mover los elementos uno a uno?

2. (3 puntos) – Sea $A = (N, C)$ un árbol (notemos que $|C| = |N| - 1$) y un predicado $p : C \to \{true, false\}$. Queremos realizar consultas y acciones que pueden ser:

   - Consulta $forall(x, y)$: Para $x, y \in N$, que indique si evaluar p para *todas* las conexiones entre los nodos x e y resulta en $true$.
   - Consulta $exists(x, y)$: Para $x, y \in N$, que indique si evaluar p para *alguna* de las conexiones entre los nodos x e y resulta en $true$.
   - Acción $toggle(x, y)$: Para $x, y \in N$, que, para cada conexión entre los nodos x e y, invierta su valor de verdad (es decir, que, si antes del *toggle* la conexión tenía $true$, ahora tendrá $false$ y viceversa).

   Diseñe un algoritmo que pueda realizar Q consultas y acciones de cualquiera de estas formas en tiempo $O(|N| + Q \log^2 |N|)$, usando memoria adicional $O(|N|)$

   Pistas:

   - Realice un precondicionamiento adecuado en $O(|N|)$, que le permita responder cada consulta o realizar cada acción en $O(\log^2 |N|)$.
   - Si les da pereza implementar *toggle*, van por buen camino.

3. (3 puntos) – Considere un arreglo `A[1..N]`, que representa una permutación de 1 a N.

   Se desea que responda Q consultas de la forma `seleccion(i, j, k)`. Esta consulta pide calcular el k–ésimo elemento del subarreglo `A[i..j]`, si ese subarreglo estuviera ordenado.

   Tomemos, por ejemplo, `A = [2,6,3,1,8,4,7,9,5]`:

   - Al hacer `consulta(2,5,3)`, se refiere al subarreglo comprendido entre las posiciones 2 y 5; es decir: `[6,3,1,8]`. Si ordenáramos este sub–arreglo, el resultado sería `[1,3,6,8]` y el tercero (3–ésimo elemento) sería 6.
   - Al hacer `consulta(3,7,1)`, se refiere al subarreglo comprendido entre las posiciones 3 y 7; es decir: `[3,1,8,4,7]`. Si ordenáramos este sub–arreglo, el resultado sería `[1,3,4,7,8]` y el primero (1–ésimo elemento) sería 1.
   - Al hacer `consulta(1,9,5)`, se refiere al subarreglo comprendido entre las posiciones 1 y 9; es decir: `[2,6,3,1,8,4,7,9,5]`. Si ordenáramos este sub–arreglo, el resultado sería `[1,2,3,4,5,6,7,8,9]` y el quinto (5–ésimo elemento) sería 5.

   Se desea que diseñe un algoritmo que responda a todas las consultas en tiempo $O((N + Q) \log N)$ y en memoria $O(N \log N)$.

   Pistas:

   - Consideremos un arreglo de ocurrencias, donde la i–ésima posición representa la presencia del valor i (1 si está presente y 0 si no). Consideremos el mismo subarreglo del primer ejemplo: `[6,3,1,8]`. Su arreglo de ocurrencias sería $[1, 0, 1, 0, 0, 1, 0, 1, 0]$.
   - En el arreglo de ocurrencias anterior, ¿cuántas veces aparecen los números del 2 al 5? O, en general, ¿cuántas veces aparecen los números del i al j? ¿Hay alguna estructura que permita responder este tipo de consultas *de forma eficiente*?
   - En algún momento hablamos sobre arreglos cumulativos para resolver consultas del estilo `suma(i, j)`. Una idea en particular que usamos ahí podría ser de utilidad.
   - Cuando sientan que el problema se vuelve muy difícil, sean *persistentes*.
   - El tiempo y el espacio son uno.

Ricardo Monascal / Noviembre 2025