Auxiliar 7: Repaso C1

Profesores: Iván Sipiran
Nelson Baloian
Patricio Poblete

Auxiliares: Alonso Almendras, Albani Olivieri
Vicente Olivares, Ricardo Valdivia
Sebastián Acuña, Martín Paredes

Nota: Este auxiliar está basado en el Control 1 del semestre 2022-2.

P1. Preguntas teóricas. Responda brevemente cada una de las siguientes preguntas. Fundamente su respuesta.

a) ¿Cuál es la altura máxima que puede tener un árbol binario con n nodos internos? Dibuje un árbol que tenga dicha altura.

b) ¿Qué ecuación de recurrencia describe la cantidad de movimientos necesarios para mover una torre de Hanoï de altura n? ¿Cuál es su solución exacta?

c) ¿Qué son los nodos internos y externos de un árbol binario? ¿Qué relación hay entre la cantidad i de nodos internos y la cantidad e de nodos externos en un árbol binario?

d) ¿Cuál es la altura h de un árbol binario completo con n nodos (i.e. en que todos sus niveles están llenos)? ¿De un árbol quasi-completo con n nodos (i.e. todos sus niveles excepto quizás el ultimo están llenos)?

e) Un algoritmo recursivo divide un problema de tamaño n en tres subproblemas de tamaño $\frac{n}{2}$ con la misma estructura que el problema original, y realiza trabajo extra $3n$ para encontrar la solución final. El caso base $n = 1$ se resuelve en tiempo constante. ¿Cual es la complejidad temporal del algoritmo?

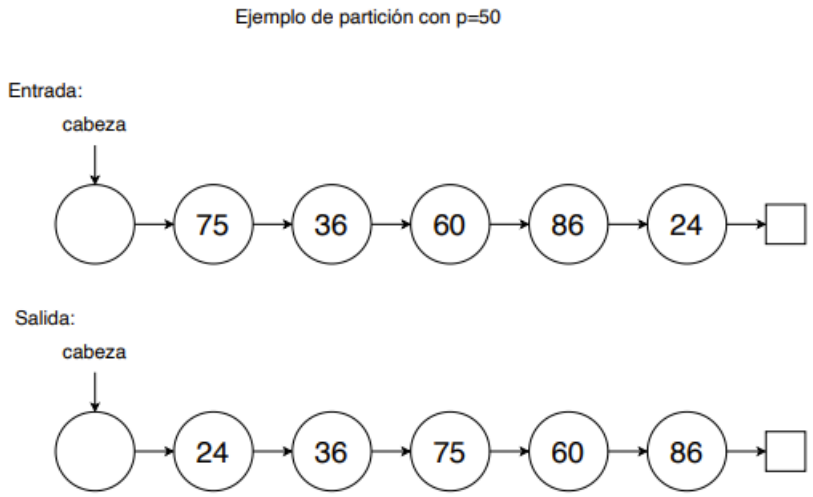
f) Considere la siguiente afirmación: "En Programación Dinámica, la técnica conocida como *memoización* no guarda valores en una tabla, ya que corresponde a un algoritmo recursivo y no a uno iterativo." ¿La afirmación es verdadera o falsa? ¿Por qué?

P2. Buscar elementos en arreglos ordenados. Suponga que se dispone de un arreglo A de capacidad ilimitada (es decir, $\text{len}(A) = \infty$) que contiene números enteros y en donde los índices del arreglo comienzan en 1. Los primeros n casilleros del arreglo (n es un valor desconocido) contienen números enteros en orden ascendente, y a partir del casillero $n + 1$ en adelante se almacena el valor ∞. Se propone el siguiente algoritmo para buscar el índice del arreglo donde se almacena un cierto valor x:

- Se verifica si el primer casillero del arreglo (índice $= 1$) contiene x. Si lo contiene, retornar dicho índice.
- Continuar revisando casilleros de la siguiente manera: si en una iteración se revisa el casillero de índice i, en la siguiente se debe revisar el casillero de índice $2i$ (esto es: $i = 1, 2, 4, 8, 16, ...$). Repetir mientras el número almacenado en el casillero de índice i sea menor que x.
- Si en el casillero de índice i se encuentra un número mayor que x, hacer búsqueda binaria entre los casilleros $[i//2, i]$.
- Si en el casillero de índice i se encuentra almacenado ∞, hacer búsqueda binaria entre los casilleros $[i//2, i]$.

¿Cuál es la complejidad temporal en el peor caso de este algoritmo de búsqueda en función de n? Fundamente su respuesta.

P3. Particionar una lista enlazada. Escriba una función `particion(cabeza,p)`, que reciba como parámetros un puntero cabeza al comienzo de una lista enlazada con nodo cabecera, y un entero `p`, y reordene los nodos de la lista de modo que queden primero todos los menores que `p` y luego los mayores que `p`. Suponga que ningún nodo de la lista contiene un valor igual a `p`. No es obligatorio preservar el orden relativo en que venían los nodos originalmente. La siguiente figura muestra un ejemplo de esto:



Su algoritmo debe ser iterativo y no debe crear nuevos nodos, sino solo reemplazar los nodos existentes.

P4. Distancia de edición en *strings*. Dado dos cadenas de texto S (la fuente) de tamaño n y T (el objetivo) de tamaño m, y un conjunto S de operadores modificando un solo símbolo en una cadena de texto (e.g. *Insertar* una ocurrencia de un símbolo, *Borrado* de una ocurrencia de un símbolo), la distancia de edición de S a T por los operadores S corresponde al tamaño de la secuencia más corta de aplicaciones de operadores de S para transformar S en T. Por ejemplo, la cadena $S =$ "*VICHO*" se transforma en la cadena $T =$ "*RICHI*" haciendo 2 *inserciones* y 2 *borrados*, por lo cual la distancia de edición de S a T por los operadores {*inserción*,*borrado*} es 4.

a) Describa una relación de recurrencia que permita calcular la distancia de edición de S a T por los operadores {*inserción*,*borrado*}. *Hint:* Ojo que este problema es similar pero distinto al problema de calcular la distancia de edición de S a T por los operadores {*inserción*,*borrado*,*reemplazo*}!

b) Describa una solución usando programación dinámica para calcular la distancia de edición de S a T por los operadores {*inserción*,*borrado*} en tiempo $O(nm)$. Justifique el tiempo de su solución.