PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC 2133 — Estructuras de Datos y Algoritmos
Guía de Ejercicios I1

1. Desafortunadamente, un arreglo que contiene elementos con un misma clave repetida muchas veces lleva a QuickSort a desempeñarse cercano a su peor caso. En el extremo, con un arreglo con un único valor para la clave de ordenación, el algoritmo es cuadrático.

   a) Muestre cómo modificar $\mathrm{Partition}(A, p, r)$ visto en clases **sin agregar** un *keyword* de iteración adicional[1], de tal forma de evitar malos casos con muchas claves repetidas.

   b) Argumente por qué su algoritmo resuelve el problema y por qué es correcto.

2. *a*) Use inducción para demostrar que Merge Sort es correcto.

   b) Use inducción en el número de dígitos máximo de los números a ordenar para demostrar que Radix Sort es correcto.

   c) ¿Bajo qué tipos de input RadixSort es cuadrático? Demuestrelo.

   d) Demuestre que la implementación de Quick Sort vista en clases no es estable.

   e) Identifique cuáles de los algoritmos vistos en clases son estables y demuestre lo más detalladamente posible.

3. La idea detrás de *Bucket Sort* se puede usar para diseñar una estructura de cola de prioridades, que, bajo ciertas condiciones, puede ser más eficiente que un Heap binario. Cuando insertamos un elemento con clave K, lo ubicamos en una lista ligada asociada a un bucket, igual que en Bucket Sort. Para que la operación Extract-Min (o Extract-Max) sea eficiente, mantenemos siempre un puntero al bucket que contiene el elemento menor (o mayor). Escriba un pseudocódigos para todas las operaciones estándar de las colas de prioridades.

4. Use la estructura de datos anterior para implementar un algoritmo de ordenación.

5. Un string de paréntesis balanceados contiene solo los carácteres "(" y ")". Inductivamente, se definen de esta forma:

   a) el string vacío es un string de paréntesis balanceados,

   b) si s y t son strings de paréntesis balanceados entonces st y (s) también lo son.

   Dé un algoritmo de programación dinámica, que reciba a n como parámetro, y que retorne el número de strings con paréntesis balanceados de tamaño menor o igual a n.

6. El problema de *alinear dos strings s y t* construidos sobre caracteres de un conjunto Σ, consiste en insertar el carácter "-" cero o más veces entre carácteres de s y t para obtener, respectivamente, dos strings s' y t' donde:

   a) el tamaño de s' y t' son el mismo.

   b) nunca ocurre que $s'[i] =$ - y $t'[i] =$ - para el mismo i.

[1] Esta restricción tiene sentido hacerla por eficiencia.

La función de *puntaje* para un cierto alineamiento (s', t') está dada por $P(s', t') = \sum_{0 \leq i < |s|} P(s[i], s[j])$, tal que cuando c y c' son caracteres:

$$P(c, c') = \begin{cases} 4 & \text{si } c = \text{- o } c' = \text{-} \\ 0 & \text{si } c = c' \text{ pero } c \neq \text{-} \\ 2 & \text{en caso contrario} \end{cases}$$

Escriba el pseudocódigo de un algoritmo de programación dinámica que encuentra el alineamiento de menor puntaje dados dos strings. Analice el algoritmo en términos del tamaño de s y t.

7. Un *heap flojo* es uno en donde $A[Parent(Parent(i))] \geq A[i]$ cuando $i \geq 4$, y tal que $A[Parent(i)] \geq A[i]$, cuando $1 \leq i < 4$. Así, se cumple que todo heap también es un heap flojo, pero no al revés.

   a) Dé un pseudocódigo para la inserción de un elemento en un heap flojo y argumente que es correcta y más eficiente en la práctica que la misma operación en heaps tradicionales.

   b) Explique en detalle (no es necesario un pseudocódigo) cómo se debe implementar la operación *Heapify*. ¿Es esta operación más eficiente que su homónima en heaps? Justifique.

8. El número de caminos más cortos entre dos nodos u y v de un grafo $G = (V, E)$ con función de pesos w, es denotado por $M_{G,w}(u, v)$ y se define formalmente como $|\{p : u \overset{p}{\rightsquigarrow} v, w(p) = \delta(u, v)\}|$. Escriba un pseudocódigo detallado de un algoritmo que, dados G, u y v, encuentre $\max_{u,v \in V}\{M_{G,w}(u, v)\}$ en $O(|V|^3)$.

   Definición
   La relación $u \overset{p}{\rightsquigarrow} v$ se lee en castellano como "u está conectado con v a través del camino p".
   Si esa definición intuitiva no le es suficientemente clara, le puede ayudar mirar su definición inductiva:

   - Si p es la secuencia vacía y $u \in V$, entonces $u \overset{p}{\rightsquigarrow} u$.
   - Si $p = up'$, $(u, v) \in E$, y $v \overset{p'}{\rightsquigarrow} w$, entonces $u \overset{p}{\rightsquigarrow} w$.

9. *a*) Deduzca una cota lo más ajustada posible (usando notación Θ) para el tiempo de ejecución de un algoritmo cuyo tiempo de ejecución promedio está dado por

$$T(n) = \begin{cases} 1 & \text{cuando } n = 1 \\ n^2 + \frac{2}{n} \sum_{i=2}^{n-1} T(n) & \text{cuando } n > 1 \end{cases}$$

   b) ¿Es verdadero que el tiempo de ejecución de *insertion sort*, en el mejor caso es $\Omega(n^2)$? Argumente usando el pseudocódigo del algoritmo.