Rensso Victor Hugo Mora Colque
Email: rvhmora@ucsp.edu.pe
Course: Análisis y Diseño de Algoritmos
Universidad Católica San Pablo

Práctica - 1
Semestre: 2022-1
Fecha de entrega: 6 de mayo 2022

Ejercicio 1[2pt]
Utilizando las reglas de notación asintóticas, demuestre las expresiones son verdaderas o falsas, justificando con la propia demostración matemática y también con un breve comentario en caso sea necesario. Asuma que toda función presentada es positiva.

Contenido:

- $max(f(n), g(n)) = \Theta(f(n) + g(n))$
- $(n + a)^b = \Theta(n^b) \mid a, b \in R^+, b > 0$
- $2^{n+1} = O(2^n)$
- $2^{2n} = O(2^n)$
- $log_2 log_2(n) = O(loglog(n))$
- Probar por inducción:
  $\sum_{i=1}^{n} \frac{1}{i^2} \leq 2 - \frac{1}{n}$
- Probar por inducción:
  $\lceil \frac{n}{2} \rceil = \begin{cases} \frac{n}{2} \ si \ n \ es \ par \\ \frac{n+1}{2} \ si \ n \ es \ impar \end{cases}$
- Probar por inducción:
  $\forall n \geq 0$ se cumple que $n^5 - n$ es divisible por 5.

Ejercicio 2[2pt]
La recurrencia $T(n) = 7T(\frac{n}{2}) + n^2$ representa la función de complejidad del algoritmo A. Un algoritmo alternativo A' para el mismo problema tiene la siguiente ecuación $T'(n) = aT'(\frac{n}{4}) + n^2$. Cuál es el mayor número entero a, que hace que el algoritmo A' sea asintótica-mente más rápido que A.

Contenido: Describa detalladamente el proceso para encontrar el valor de a y brinde como respuesta a^2.

Ejercicio 3[5pt]
Dada las siguientes ecuaciones de recurrencia, **resolver utilizando expansión de recurrencia y teorema maestro**. En ambos casos colocar todo el desarrollo no obviar partes. Si en caso no se pueda aplicar el teorema maestro, explique el motivo. Asuma que $T(n)$ es constante para $n < 2$.

Contenido:

- $T(n) = 4T(\frac{n}{2}) + n$
- $T(n) = 4T(\frac{n}{2}) + n^2$
- $T(n) = 4T(\frac{n}{2}) + n^3$
- $T(n) = T(\frac{9n}{10}) + n$

- $T(n) = 16T(\frac{n}{4}) + n^2$
- $T(n) = 7T(\frac{n}{3}) + n^2$
- $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$
- $T(n) = 4T(\frac{n}{2}) + n^2 logn$
- $T(n) = T(n-1) + n$
- $T(n) = \sqrt{n} + 1$

Ejercicio 4[5pt]
Se sabe que, para resolver el problema de ordenación de números, el algoritmo Mergesort posee una complejidad de $\Theta(n\ logn)$ y el Insertion Sort $O(n^2)$. Sin embargo, los factores constantes del algoritmo Insertion lo tornan más veloz para vectores de tamaño n pequeños. Por ende tiene sentido realizar una combinación de ambos, y utilizar el algoritmo de inserción cuando los problemas se tornen lo suficientemente pequeños. Considere la siguiente modificación del Mergesort: $\frac{n}{k}$ sub listas de tamaño k son ordenadas utilizando el algoritmo de inserción, y luego combinadas utilizando el mecanismo del Mergesort (*Merge*), siendo k la variable a ser determinada.

Contenido: Cada item vale un punto.

- Presente el algoritmo, y en anexo coloque el código.
- En la práctica (realice un análisis estadístico) cual es el valor de k.
- Muestre que las $\frac{n}{k}$ sub-listas, cada una de tamaño k, pueden ser ordenadas por el algoritmo Insertion en el peor caso en $O(nk)$.
- Muestre que las listas pueden ser combinadas en el peor caso en tiempo $O(n\ log(\frac{n}{k}))$.
- Sea A= [1...n] un vector de números distintos. Si se cumple que $i < j$ y $A[i] > A[j]$, entonces el par (i, j) es llamado de "inversión" del vector A. Modifique el algoritmo del Mergesort para encontrar las inversiones de un vector. En anexo del mismo problema coloque el código.

Ejercicio 5[5pt]
Para los problemas presentados, cree un código en $C++$ [4pts], presente la complejidad del algoritmo del cual es instancia el código presentado [1pt]. Cada código deberá estar en latex es decir deben utilizar la biblioteca listings [a] [b]. Adicionalmente, deberán correr y colocar *screenshots* del código corriendo con las instancias que se les pida por cada problema.

[a] `https://nasa.github.io/nasa-latex-docs/html/examples/listing.html`
[b] `https://texdoc.org/serve/listings.pdf/0`

Contenido:

- Desarrolle un programa ***"Recursivo"*** para generar la descomposición de un número entero positivo, en la suma de todos los posibles factores. La presentación de los factores debe estar ordenada de mayor a menor. Por ejemplo para $n = 5$

  5
  4 + 1
  3 + 2
  3 + 1 + 1

$2 + 2 + 1$
$2 + 1 + 1 + 1$
$1 + 1 + 1 + 1 + 1$

Obs. Note que los números se encuentran ordenados de forma decreciente en cada fila y de igual forma el número inicio de cada fila también esta ordenado con respecto a la fila anterior.

Instancias del problema: $n = [5, 7, 11, 19, 23]$.

- Dado un vector con n números enteros, determine la máxima suma en un subvector contiguo de ese vector. Si todos los números fueran negativos asuma que la suma es 0. Por ejemplo en el siguiente vector $A = [31, -41, 59, 26, -53, 58, 97, -93 - 23, 84]$ la mayor suma de elementos contiguos es 187 que se encuentra en el rango de 3 a 7. El algoritmo del código a presentar ***debe tener complejidad lineal***.

  Instancias del problema :

  1. $A = [31, -41, 59, 26, -53, 58, 97, -93 - 23, 84]$
  2. $A = [0, 0, 0, 0, 0, 0, 0, 0, -1, 1]$
  3. $A = [1, 1, 1, 1, 1, 1, 0, 1, -1, 2]$
  4. $A = [-1, 1, -1, 1, -1, 1, -1, 1, -1, 1000]$
  5. $A = [23, 1, 12, 11, 41, 22, 18, 4, 2, 6]$
