Práctico 2

Análisis de algoritmos

(I) Imprescindible
(R) Recomendado
(C) Complementario

Notas: Para el análisis del tiempo de ejecución se manejarán los siguientes casos:

- **Exacto**: Análisis en base a la cantidad de operaciones realizadas, asignándole costo 1 a cada operación.
- **En base a Op**: Análisis basándose en el estudio de una operación (**Op básica**)
- **Orden**: Análisis que se basa en dar una expresión para el orden lo más exacta posible

EJERCICIO 1 (I)

Haga un análisis del tiempo de ejecución para cada uno de los siguientes seis fragmentos de programa:

1. en forma exacta (cantidad de operaciones)
2. definiendo sum++ como **Op** básica
3. dando el orden de ejecución.

```
(1)   sum = 0;
      for (i = 1; i <= n; i++)
            sum++;
(2)   sum = 0;
      for (i = 1; i <= n; i++)
            for (j = 1; j <= n; j++)
                  sum++;
(3)   sum = 0;
      for (i = 1; i <= n; i++)
            for ( j = 1; j <= n * n; j++)
                  sum++;
(4)   sum = 0;
      for (i = 1; i <= n; i++)
      for (j = 1; j <= i; j++)
            sum++;
(5)   sum = 0;
      for (i = 1; i <= n; i++)
      for (j = 1; j <= i * i; j++)
            for (k = 1; k <= j; k++)
                  sum++;
(6)   sum = 0;
      for (i = 1; i <= n; i++)
      for (j = 1; j <= i * i; j++)
            if (j % i == 0)
                  for (k = 1; k <= j; k++)
                        sum++;
```

EJERCICIO 2 (I)

Una implementación de un cierto algoritmo lleva un tiempo acotado superiormente por:

$t(n) = 3\ s - 18\ n\ ms + 27\ n^2\ \mu s$

para resolver cualquier instancia de tamaño *n*.

Encuentre una función f: $N \rightarrow R^*$, lo más simple posible, tal que el algoritmo tenga un tiempo en el orden de f(n). Pruebe que $t(n) \in O(f(n))$.

EJERCICIO 3 (I)

Demostrar la **regla del límite**.

Dadas dos funciones arbitrarias f: $N \rightarrow R^*$ y g: $N \rightarrow R^*$

1) Si lím $\{n \rightarrow +\infty\}$ $(f(n) / g(n)) \in R^+$ entonces $f(n) \in O(g(n))$ y $g(n) \in O(f(n))$
2) Si lím $\{n \rightarrow +\infty\}$ $(f(n) / g(n)) = 0$ entonces $f(n) \in O(g(n))$ pero $g(n) \notin O(f(n))$
3) Si lím $\{n \rightarrow +\infty\}$ $(f(n) / g(n)) = +\infty$ entonces $f(n) \notin O(g(n))$ pero $g(n) \in O(f(n))$

EJERCICIO 4 (I)

Demostrar que:

1) $f(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$.

2) $O(f(n)+g(n)) = O(\max \{f(n),g(n)\})$

considerando:

- f y g son funciones que retornan únicamente valores positivos
- $f > g$ si $\exists n_0 \in N, \forall n \in N, n > n_0, f(n) > g(n)$

EJERCICIO 5 (I)

Dado el problema de buscar un elemento en un arreglo desordenado de tamaño n. Sabiendo que el elemento se encuentra y que las probabilidades de que se encuentre en cada posición son las mismas:

1. Implementar la rutina de búsqueda en C/C++.
2. Calcule el tiempo en forma exacta en el peor caso, mejor caso y caso promedio del algoritmo.

EJERCICIO 6 (R)

Considere el siguiente algoritmo (conocido como regla de Horner) para evaluar $f(x)= \sum_{i=0}^{n} a_i x^i$:

```
poli = 0;
for (i = n; i >= 0; i--)
    poli = x * poli + a[i];
```

1. Muestre como se realizan los pasos para este algoritmo con x = 3, $f(x) = 4x^4 + 8x^3 + x + 2$.
2. Calcule en forma exacta (cantidad de operaciones) el tiempo de ejecución de este algoritmo.

EJERCICIO 7 (R)

Definiciones:

- f(n) crece más lento que g(n) si f(n) ∈ O(g(n)) y f(n) ∉ Θ(g(n)).
- f(n) y g(n) crecen con las misma tasa si: f(n) ∈ Θ(g(n))

Ordene las siguientes funciones por orden de crecimiento, indicando cuales crecen con la misma tasa.

1. $n, \sqrt{n}, n^{1.5}, n^2, n\log(n), n\log(\log(n)), n\log^2(n), n\log(n^2), \frac{2}{n}, 2^n, 2^{\frac{n}{2}}, 37, n^2\log(n), n^3$
2. $\frac{n}{\log(n)}, n, \frac{\log(n)}{n}, n^n, n^2\log(n), \frac{n^2}{\log(n)}, n\log(n), \log(\log(n)), n\log^2(n), n^2, 2^n, \frac{1}{2}^2$

EJERCICIO 8 (R)

¿Cuáles de las siguientes afirmaciones son ciertas?

1. $n^2 \in O(n^2)$
2. $n^3 \in O(n)$
3. $2^{n+1} \in O(2^n)$
4. $(n + 1)! \in O(n!)$
5. Para cualquier función f: $N \to R^*$, $f(n) \in O(n) \Rightarrow f^2(n) \in O(n^2)$
6. Para cualquier función f: $N \to R^*$, $f(n) \in O(n) \Rightarrow 2^{f(n)} \in O(2^n)$

EJERCICIO 9 (R)

Se analizan los programas A y B y se encuentra que tienen un tiempo de ejecución en el peor caso no mayor que $150.n.log_2(n)$ y n^2 respectivamente. Responda las siguientes preguntas, si es posible:

1. ¿Qué programa tiene mejor tiempo de ejecución, para valores grandes de n (n > 10.000)?
2. ¿Qué programa tiene mejor tiempo de ejecución, para valores pequeños de n (n < 100)?
3. ¿Qué programa se ejecutará más rapidamente para n = 1000?

EJERCICIO 10 (R)

Escribir un programa C/C++ de ordenación por selección que ordene un arreglo de enteros en orden ascendente mediante la localización del valor más pequeño en cada iteración y el intercambio de este con el componente apropiado.

Calcule el orden del tiempo de ejecución, tomando como operación básica la comparación de 2 elementos del arreglo, en el mejor caso, caso promedio y peor caso, dando un ejemplo de entrada para cada caso. Por último, compare los resultados obtenidos.

EJERCICIO 11 (R)

Considere las siguientes funciones de n:

$$f_1(n) = n^2$$

$$f_2(n) = n^2 + 1000n$$

$$f_3(n) = \begin{cases} n & n\ impar \\ n^3 & n\ par \end{cases}$$

$$f_4(n) = \begin{cases} n & n \leq 100 \\ n^3 & n > 100 \end{cases}$$

Indique para cada par de distintos valores de i y j cuando $f_i(n) \in O(f_j(n))$ y cuando $f_i(n) \in \Omega(f_j(n))$.

EJERCICIO 12 (R)

Encuentre dos funciones f(n) y g(n) tales que $f(n) \notin O(g(n))$ y $g(n) \notin O(f(n))$.

EJERCICIO 13 (R)

Supóngase que $T_1(n) \in O(f(n))$ y $T_2(n) \in O(f(n))$. ¿Cuáles de las siguientes afirmaciones son ciertas?

1. $T_1(n) + T_2(n) \in O(f(n))$
2. $T_1(n) - T_2(n) \in O(f(n))$
3. $T_1(n)/T_2(n) \in O(1)$
4. $T_1(n) \in O(T_2(n))$

EJERCICIO 14 (C)

La criba de Eratostenes es un método para obtener todos los primos menores que n. Se empieza haciendo una tabla de enteros entre 2 y n. Se encuentra el entero más pequeño i, que no este marcado, se imprime i y se marcan i, 2i, 3i, … El algoritmo termina cuando $i > \sqrt{n}$. Verifique que cuando $i > \sqrt{n}$ solo quedan sin marcar números primos. Escriba un programa C/C++ para la criba de Eratostenes. ¿Cuál es el orden del tiempo de ejecución de este programa?

EJERCICIO 15 (C)

Demostrar que si f(n) es estrictamente positiva $\forall n \in N$, son equivalentes las siguientes definiciones:

1. $O_1(f) = \{g: N \to R^* / \exists c \in R^+, \exists n_0 \in N, \ \forall n \in N, n \geq n_0, g \leq cf\}$
2. $O_2(f) = \{g: N \to R^* / \exists c \in R^+, \forall n \in N, g \leq cf\}$

EJERCICIO 16 (C)

Suponga $T_1(n) \in \Omega(f(n))$ y $T_2(n) \in \Omega(g(n))$, $f(n), g(n) \geq 0 \ \forall n$. ¿Cuáles de las siguientes sentencias son verdaderas?

1. $T_1(n) + T_2(n) \in \Omega(\max(f(n), g(n)))$.
2. $T_1(n)/T_2(n) \in \Omega(f(n)g(n))$.

EJERCICIO 17 (C)

Probar que las relaciones O y Ω son transitivas, es decir:

1. $f(n) \in O(g(n)), g(n) \in O(h(n)) \Rightarrow f(n) \in O(h(n))$
2. $f(n) \in \Omega(g(n)), g(n) \in \Omega(h(n)) \Rightarrow f(n) \in \Omega(h(n))$

EJERCICIO 18 (C)

Probar:

1. $f(n) \in \Theta(g(n)) \Leftrightarrow g(n) \in \Theta(f(n))$
2. Θ define una relación de equivalencia R entre las funciones, es decir, $f(n)Rg(n) \Leftrightarrow f(n) \in \Theta(g(n))$

EJERCICIO 19 (C)

Dado el ejercicio 18 del práctico 1, determinar cual es el mejor caso y el peor caso para este problema y analizar los tiempos de ejecución en cada uno de ellos.

EJERCICIO 20 (C)

Escribir un programa C/C++ de ordenación por inserción que ordene un arreglo de enteros en orden ascendente. Este algoritmo consiste en un filtrado descendente de los elementos del arreglo. Se comienza por el principio del arreglo considerando este elemento como el menor. Un paso consiste en tomar el siguiente elemento y filtrarlo hacia la izquierda hasta colocarlo ordenado en ese sector del arreglo. De esta forma en el paso k del algoritmo se tiene el arreglo ordenado desde 0 hasta k. El paso se repite N-1 veces para un arreglo de N elementos.

Calcule el orden del tiempo de ejecución, tomando como operación básica la comparación de 2 elementos del arreglo, en el mejor caso, caso promedio y peor caso, dando un ejemplo de entrada para cada caso. Por último, compare los resultados obtenidos.