PRACTICA 1 ADA

Harold Alejandro Villanueva Borda

Mayo 2022

---

1. 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:

- $\text{máx}(f(n), g(n)) = \Theta(f(n) + g(n))$

  Solución:

  Para probar esto, demostramos que existen las constantes $C_1, C_2, n_0 > 0$ tales que, para todo $n \geq n_0$,

  $$0 \leq C_1(f(n) + g(n)) \leq max(f(n), g(n)) \leq C_2(f(n) + g(n))$$

  Como las funciones son asitóticamente no negativa, podemos suponer que para ningun $n_0 > f(n) \geq 0$, por lo tanto, $n \geq n_0$,

  $$f(n) + g(n) \geq max(f(n), g(n))$$

  Tener en cuenta que $f(n) \leq max(f(n), g(n)) y g(n) \leq max(f(n), g(n))$

  $$\begin{aligned} f(n) &\leq max(f(n), g(n)) && (1)\\ g(n) &\leq max(f(n), g(n)) && (2)\\ f(n) + g(n) &\leq 2max(f(n), g(n)) && (3)\\ \frac{1}{2}(f(n) + g(n)) &\leq max(f(n), g(n)) && (4) \end{aligned}$$

  Por lo tanto podemos combinar las desigualdades anteriores:

  $$0 \leq \tfrac{1}{2}(f(n) + g(n)) \leq max(f(n), g(n)) \leq (f(n) + g(n)), \text{ para } n \geq n_0$$

  Entonces $max(f(n), g(n)) = \theta(f(n), g(n))$ porque existe $C_1 = \frac{1}{2}$ y $C_2 = 1$

- $(n + a)^b = \Theta\left(n^b\right) \mid a, b \in R^+, b > 0$

  Solución:

  Para probar esto, demostramos que existen las constantes $C_1, C_2, n_0 > 0$ tales que, para todo $n \geq n_0$,

  $$0 \leq C_1 n^b \leq (n + a)^b \leq C_2 n^b$$

Note que $n + a \leq n+ |a| \leq 2n$ cuando $|a| \leq n$, y $n + a \geq n-|a| \geq \frac{1}{2}n$ cuando $|a| \leq \frac{1}{2}n$. Así cuando $n \geq 2\,|a|$

$$\begin{aligned}
(n+a)^b &\leq (n+ \mid a \mid)^b, n > 0 && (5)\\
&\leq (n+n)^b, n \geq \mid a \mid && (6)\\
&= (2n)^b && (7)\\
&= C_1 n^b, C_1 = 2^b && (8)\\
& && (9)
\end{aligned}$$

$$\begin{aligned}
(n+a)^b &= \Omega(n^b) && (10)\\
(n+a)^b &\geq (n+ \mid a \mid)^b, n > 0 && (11)\\
&\geq (C_2 n)^b, C_2 = \frac{1}{2}, n \geq 2 \mid a \mid && (12)\\
& && (13)
\end{aligned}$$

Como $\frac{n}{2} \leq n- |a|$, para $n \geq 2\,|a|$, entonces $0 \leq \frac{n}{2} \leq n + a \leq 2n$
Ya que b ¿0, la desigualdad sigue siendo valida cuando todas las partes se elevan a la potencia b.

$$0 \leq (\frac{n}{2})^b \leq (n+a)^b \leq (2n)^b \quad (14)$$

$$0 \leq (\frac{n}{2})^b n^b \leq (n+a)^b \leq 2^b n^b \quad (15)$$

$\therefore C_1 = \frac{1}{2^b} = 2^- b$, y $n_0 = 2\,|a|$ satisface la definición.

- $2^{n+1} = O\,(2^n)$

  Solución:
  $2^{n+1} = 2.2^{n+1} \leq C2^n$, para cualquier $C \leq 2$ se cumple

  $\therefore 2^{n+1} = O\,(2^n)$ es verdadera.

- $2^{2n} = O\,(2^n)$

  Solución:
  $2^{2n} = 2^2.2^n \leq C2^n$

  $2^n \leq C$

  vemos que no tiene tal constante C, ya que 2^n no está acotado

  $\therefore 2^{2n} \neq O\,(2^n)\, esfalso.$

- $\log_2 \log_2(n) = O(\log\log(n))$

  Solución:
  $\log_2 \log_2(n) \leq C \log\log(n)$

  $C \log((\log(n))) \geq \frac{\log(\frac{\log(n)}{\log(2)})}{\log(2)}$

  $\frac{\log(\log(n)-\log(2))}{\log(2)} \leq C \log(\log(n))$

$C\log(2) - 1 \neq 0$, $C > \frac{1}{\log(2)}$

$\therefore O(\log\log(n))$ es erdadera.

- Probar por inducción:
  $\sum_{i=1}^{n} \frac{1}{i^2} \leq 2 - \frac{1}{n}$

  Solución:

  Prueba (por inducción matemática):

  1. Paso Base:
     $P(n_0) = P(1)$, para $n_0 = 1$

     $\sum_{i=1}^{1} \frac{1}{1^2} \leq 2 - \frac{1}{1}$

     $1 \leq 2 - \frac{1}{1}$

     $1 \leq 1$, es verdadera para $n_0 = 1$.
  2. Paso inductivo:
     Si la fórmula es verdadera para $n = k$, entonces también debe ser verdadera para $n = k + 1$, i.e., $P(k) \longrightarrow P(k + 1)$.

     – Suponga que la formula sea verdadera para $n = k$, i.e.,

     $P(k) = \sum_{i=1}^{k} \frac{1}{i^2} \leq 2 - \frac{1}{k}$

     para algún entero $k \geq 1$. [hipótesis inductiva]

     – Se debe demostrar que

     $P(k+1) = \sum_{i=1}^{k+1} \frac{1}{i^2} \leq 2 - \frac{1}{k+1}$

$$\sum_{i=1}^{k+1} \frac{1}{i^2} = \sum_{i=1}^{k} \frac{1}{i^2} + \frac{1}{(k+1)^2} \tag{16}$$

$$= (2 - \frac{1}{k}) + \frac{1}{(k+1)^2} \tag{17}$$

$$= \frac{2k-1}{k} + \frac{1}{k^2+2k+1} \tag{18}$$

$$= \frac{(2k-1)(k^2+2k+1)+k}{(k)(k^2+2k+1)} \tag{19}$$

$$= \frac{2k3+3k^2+k-1}{k(k+1)^2} \tag{20}$$

$$= \frac{1}{(k+1)^2} - \frac{1}{k} + 2 \tag{21}$$

     – Entonces $\frac{1}{(k+1)^2} - \frac{1}{k} + 2 \neq 2 - \frac{1}{k+1}$, por lo tanto P(k) no implica P(k + 1), eso quiere decir que la desigualdad indica que el paso inductivo ha fallado.

     $\therefore$ Se ha demostrado que la afirmación es falsa.

- Probar por inducción:

$\lceil \frac{n}{2} \rceil = \begin{cases} \frac{n}{2} \ si \ n \ es \ par \\ \\ \frac{n+1}{2} si \ n \ es \ impar \end{cases}$

Solución:

Prueba (por inducción matemática):

1. Paso base: $P(n_0) = P(1)$, para $n_0 = 1$ cuando n es impar

   $\lceil \frac{1}{2} \rceil = \left(\frac{1+1}{2}\right)$

   $\lceil \frac{1}{2} \rceil = 1$

   $1 = 1$

   $P(n_0) = P(1)$, para $n_0 = 2$ cuando n es par

   $\lceil \frac{2}{2} \rceil = \left(\frac{2}{2}\right)$

   $1 = 1$

   Se prueba que el caso base cumple para ambas condiciones.

2. Paso inductivo:

   Si la fórmula es verdadera para $n = k$, entonces también debe ser verdadera para $n = k + 1$, i.e., $P(k) \longrightarrow P(k + 1)$.

   Para n par, $n = 2k$

   $\lceil \frac{2k}{2} \rceil = \frac{2k}{2}$

   $k = k$

   $k = \frac{n}{2}$

   Para n impar, $2k + 1$

   $\lceil \frac{2k}{2} \rceil = \frac{2k+1}{2} + 1$

   $k + 1 = \frac{2k+1}{2} + 1$

   $k = \frac{n+1}{2}$

   $\therefore$ se compuerba que $\lceil \frac{n}{2} \rceil = \begin{cases} \frac{n}{2} \ si \ n \ es \ par \\ \\ \frac{n+1}{2} si \ n \ es \ impar \end{cases}$ es verdadera-

- Probar por inducción:

  $\forall n \geq 0$ se cumple que $n^5 - n$ es divisible por 5.

  Solución:

  Prueba (por inducción matemática):

  1. Paso base:

$P(n_0) = P(1)$, para $n_0 = 1$

$1^5 - 1 = 0$ y 0 es divisible por 5

$n^5 - n$ es verdadera para $n_0 = 1$.

2. Paso inductivo:
Si la fórmula es verdadera para $n = k$, entonces también debe ser verdadera para $n = k + 1$, i.e., $P(k) \longrightarrow P(k + 1)$.

$-P(k) : k^5 - k$ es divisible por 5. [hipotesis inductiva]

− Se debe demsotrar que $P(k + 1) : (k + 1)^5 - (k + 1)$ es divisible por 5.

$$\begin{aligned}(k+1)^5 - (k+1) &= k^5 + 5k^4 + 10k^3 + 10k^2 + 5k - k && (22)\\ &= (k^5 - k) + (5k^4 + 10k^3 + 10k^2 + 5k) && (23)\\ &= (k^5 - k) + 5k(k+1)(k^2 + k + 1) && (24)\\ & && (25)\end{aligned}$$

− La hipótesis de inducción establece que $k^5 - k$ es divisible por 5:

$$(k^5 - k) \bmod 5 = 0$$

− $5k(k + 1)(k^2 + k + 1)$ tiene a 5 como factor, explicitamente:

$$(5k(k+1)(k^2 + k + 1)) \bmod 5 = 0$$

− $k^5 - k \bmod 5 = 0$ implica $(k + 1)^5 - (k + 1) \bmod 5 = 0$, por lo tanto el paso inductivo queda verificado.

$\therefore$ La afirmación $n^5 - n$ is divisible por 5, para $n > 0$ es verdadera.

---

2. Ejercicio $2[2pt]$

Ejercicio 2[2pt]
La recurrencia $T(n) = 7T\left(\frac{n}{2}\right) + n^2$ representa la función de complejidad del algoritmo A. Un algoritmo alternativo A' para el mismo problema tiene la siguiente ecuación $T'(n) = aT'\left(\frac{n}{4}\right) + n^2$. 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.

Solución:
− Consideramos la siguente relacón de recurrencia para el algoritmo A:

$T(n) = 7T\left(\frac{n}{2}\right) + n^2$

− Usando el caso 1 del teorema maestro tenemos:

$$\begin{aligned}a = 2, b &= 2, f(n) = n^2 && (26)\\ n^{\log_b(a)} &= n^{\log_2(7)} = n^{2,81} && (27)\end{aligned}$$

$$
\begin{aligned}
f(n) &= O\left(n^{\log_b(a)-\varepsilon}\right) && (28)\\
&= O\left(n^{2,81-0,81}\right) \ ,donde\ \varepsilon = 0{,}81 && (29)\\
&= O\left(n^2\right) && (30)\\
& && (31)
\end{aligned}
$$

– Consideramos la siguente relacón de recurrencia para el algoritmo A':

$T(n) = a'T\left(\frac{n}{4}\right) + n^2$

– Haciendo uso del teorema maestro tenemos:

$b = 4, f(n) = n^2$

– Se desconoce el valor de a. Por lo tanto, no es posible analizar la relación.

$f(n) = O\left(n^{\log_b(a)+\varepsilon}\right) \Longrightarrow O\left(n^{\log_(\ a)+\varepsilon}\right)$

– Aplicamos expansión:

$$
\begin{aligned}
T(n) &= 7T\left(\frac{n}{2}\right) + n^2 && (32)\\
&= 7\left[7T\left(\frac{n}{\frac{2}{2}}\right) + \left(\frac{n}{2}\right)^2\right] n^2 && (33)\\
&= 7^2T\left(\frac{n}{2^2}\right) + \frac{7n^2}{2^2} + n^2 && (34)\\
&= 49T\left(\frac{n}{4}\right) + \frac{11n^2}{4} && (35)\\
& && (36)
\end{aligned}
$$

– Entonces, ahroa tenemos que:

$$
\begin{aligned}
a = 49, b = 4, f(n) = n^2 \ ,donde\ f(n)\ es\ la\ misma\ para\ el\ algoritmo & && (37)\\
f(n) &= O\left(n^{\log_b(a)+\varepsilon}\right) && (38)\\
&= O\left(n^{\log_4(49)+\varepsilon}\right) && (39)\\
& && (40)
\end{aligned}
$$

– Sean las mismas relaciones de recurrencia de $T(n)$ y $T'(n)$ que:

$T(n) \geq (T'(n))$ si $a > 49$

$n^{\log_4(49)} \in n^{\log_4(a)}$

$$
\begin{aligned}
n^{\log_2(7)} &> n^{\log_4(a)} && (41)\\
\log_2(7) &> \log_4(a) && (42)\\
\log_4(a) &< \log_2(7) && (43)\\
a &< 4^{\log_2(7)} && (44)\\
a &< 49 && (45)\\
& && (46)
\end{aligned}
$$

Aquí, $T(n) \in (T'(n))$ si $a > 49$

– Por lo tanto, A' es asintoticamente mas rápido que A hasta que el valor de $a = 49$

– Expresando el resultado en a^2: $49^2 = 2401$.

---

3. 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:

- $T(n) = 4T\left(\frac{n}{2}\right) + n$

  Solución por expansión:

$$
\begin{aligned}
T(n) &= 4T\left(\frac{n}{2}\right) + n && (47)\\
&= 4\left[4T\left(\frac{n}{\frac{2}{2}}\right) + \frac{n}{2}\right] + n && (48)\\
&= 4^2T\left(\frac{n}{2^2}\right) + 2n + n && (49)\\
&= 4^2\left[4T\left(\frac{n}{\frac{2^2}{2}}\right) + \frac{n}{2^2}\right] + n && (50)\\
&= 4^3T\left(\frac{n}{2^3}\right) + 4n + 2n + n && (51)\\
&= ... && (52)\\
&= 4^kT\left(\frac{n}{2^k}\right) + \left(\sum_{i=0}^{k-1} 2^i\right) n && (53)\\
&= 4^kT\left(\frac{n}{2^k}\right) + (2^k - 1)n && (54)\\
& && (55)
\end{aligned}
$$

$\frac{n}{2^k} = 1 \Longrightarrow n = 2^k \Longrightarrow \log_2(n) = \log_2(2^k) \Longrightarrow \log_2(n) = k\log_2(2) \Longrightarrow \log_2(n) = k$

$$
\begin{aligned}
&= 4^{\log_2(n)}T(1) + (2^{\log_2(n)} - 1)n && (56)\\
&= n^{\log_2(4)}T(1) + (n^{\log_2(2)} - 1)n && (57)\\
&= n^2T(1) + n^2n && (58)\\
& && (59)
\end{aligned}
$$

$\therefore T(n) = O\left(n^2\right)$

Solución aplicando el teorema maestro:

$a = 4, b = 2, f(n) = n$

$n^{\log_b(a)} = n^{\log_2(4)} = n^2$

$n^2 > f(n)$

$n^2 > n$, ya que $f(n) = O\left(n^{2-1}\right)$ para $\varepsilon = 1$

aplica el caso 1:

$T(n) = O\left(n^{\log_b(a)}\right)$

$T(n) = O\left(n^{\log_2(4)}\right)$

$\therefore T(n) = O\left(n^2\right)$

- $T(n) = 4T\left(\frac{n}{2}\right) + n^2$

Solución:

$$
\begin{aligned}
T(n) &= 4T\left(\frac{n}{2}\right) + n^2 && (60)\\
&= 4\left[4T\left(\frac{n}{\frac{2}{2}}\right) + \left(\frac{n}{2}\right)^2\right] + n^2 && (61)\\
&= 4^2T\left(\frac{n}{2^2}\right) + n^2 + n^2 && (62)\\
&= 4^2\left[4T\left(\frac{n}{\frac{2^2}{2}}\right) + \left(\frac{n}{2^2}\right)^2\right] + n^2 + n^2 && (63)\\
&= 4^3T\left(\frac{n}{2^3}\right) + n^2 + n^2 + n^2 && (64)\\
&= ... && (65)\\
&= 4^kT\left(\frac{n}{2^k}\right) + kn^2 && (66)\\
& && (67)
\end{aligned}
$$

$\frac{n}{2^k} = 1 \Longrightarrow n = 2^k \Longrightarrow \log_2(n) = \log_2(2^k) \Longrightarrow \log_2(n) = k\log_2(2) \Longrightarrow \log_2(n) = k$

$$
\begin{aligned}
&= 4^{\log_2(n)}T(1) + n^2\log_2(n) && (68)\\
&= n^{\log_2(4)}T(1) + n^2\log_2(n) && (69)\\
&= n^2T(1) + n^2\log_2(n) && (70)\\
& && (71)
\end{aligned}
$$

$\therefore T(n) = O\left(n^2\log(n)\right)$

Solución aplicando el teorema maestro:

$a = 4, b = 2, f(n) = n^2$

$n^{\log_b(a)} = n^{\log_2(4)} = n^2$

$n^2 = f(n)$

$n^2 = n^2$, ya que $f(n) = O\left(n^{2-0}\right)$ para $\varepsilon = 0$

aplica el caso 2:

$T(n) = O\left(n^{\log_b(a)}\log(n)\right)$

$T(n) = O\left(n^{\log_2(4)}\log(n)\right)$

$\therefore T(n) = O\left(n^2\log(n)\right)$

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

  Solución:

$$\begin{aligned}
T(n) &= 4T\left(\frac{n}{2}\right) + n^3 && (72)\\
&= 4\left[4T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2}\right)^3\right] + n^2 && (73)\\
&= 4^2T\left(\frac{n}{2^2}\right) + \frac{n^3}{2^3} + n^3 && (74)\\
&= 4^2\left[4T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2}\right)^3\right] + \frac{n^3}{2^3} + n^2 && (75)\\
&= 4^3T\left(\frac{n}{2^3}\right) + \frac{n^3}{2^6} + \frac{n^3}{2^3} + n^3 && (76)\\
&= 4^3\left[4T\left(\frac{n}{2^4}\right) + \left(\frac{n}{2^3}\right)^3\right] + \frac{n^3}{2^6} + \frac{n^3}{2^3} + n^3 && (77)\\
&= 4^4T\left(\frac{n}{2^4}\right) + \frac{n^3}{2^9} + \frac{n^3}{2^6} + \frac{n^3}{2^3} + n^3 && (78)\\
&= 4^4T\left(\frac{n}{2^4}\right) + \left(1 + \frac{1}{2^3} + \frac{1}{2^6} + \frac{1}{2^9}\right)n^3 && (79)\\
&= ... && (80)\\
&= 4^kT\left(\frac{n}{2^k}\right) + \left(\sum_{i=0}^{k-1}\frac{1}{2^{3i}}\right)n^3 && (81)\\
& && (82)
\end{aligned}$$

haciendo uso de $\left(\sum_{i=0}^{k-1}\frac{1}{a^i}\right) = \frac{a(1-a^{-k})}{a-1}$

$$= 4^kT\left(\frac{n}{2^k}\right) + \left(\frac{8(1-8^{-k})}{7}\right)n^3 \quad (83)$$

$\frac{n}{2^k} = 1 \Longrightarrow n = 2^k \Longrightarrow \log_2(n) = \log_2(2^k) \Longrightarrow \log_2(n) = k\log_2(2) \Longrightarrow \log_2(n) = k$

$$\begin{aligned}
4^{log_2(n)}T(1)\left(\frac{8(1-8^{log_2(n)})}{7}\right)n^3 &= n^{log_2(4)}T(1)\left(\frac{8(1-n^{log_2(8)})}{7}\right)n^3 && (84)\\
&= n^2T(1) + \left(\frac{8(1-n^3)}{7}\right)n^3 && (85)\\
& && (86)
\end{aligned}$$

$\therefore T(n) = O\left(n^3\right)$

Solución aplicando el teorema maestro:

$a = 4, b = 2, f(n) = n^3$

$n^{\log_b(a)} = n^{\log_2(4)} = n^2$

$n^2 < f(n)$

$n^2 < n^3$, ya que $f(n) = O\left(n^{2+1}\right)$ para $\varepsilon = 1$

aplica el caso 3:

$T(n) = O\left(f(n)\right)$

$\therefore T(n) = O\left(n^3\right)$

- $T(n) = T\left(\frac{9n}{10}\right) + n$

Solución:

$$
\begin{aligned}
T(n) &= T\left(\frac{9n}{10}\right) + n && (87)\\
&= 1\left[1T\left(\frac{9n}{\frac{10}{2}}\right) + \left(\frac{n}{2}\right)\right] + n && (88)\\
&= 1^2T\left(\frac{9n}{20}\right) + \frac{n}{2} + n && (89)\\
&= 1^2\left[1T\left(\frac{9n}{\frac{20}{2}}\right) + \left(\frac{n}{2^2}\right)\right] + \frac{n}{2} + n && (90)\\
&= 1^3T\left(\frac{9n}{40}\right) + \frac{n}{2^2} + \frac{n}{2} + n && (91)\\
&= 1^3\left[1T\left(\frac{9n}{\frac{40}{2}}\right) + \left(\frac{n}{2^3}\right)\right] + \frac{n}{2^2} + \frac{n}{2} + n && (92)\\
&= 1^4T\left(\frac{9n}{80}\right) + \frac{n}{2^3} + \frac{n}{2^2} + \frac{n}{2} + n && (93)\\
&= 1^4T\left(\frac{9n}{80}\right) + \left(1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3}\right)n && (94)\\
&= ... && (95)\\
&= 1^kT\left(\frac{9n}{10(2^{k-1})}\right) + \left(\sum_{i=0}^{k-1}\frac{1}{2^i}\right)n && (96)\\
&= 1^kT\left(\frac{9n}{10(2^{k-1})}\right) + \left(2 - \frac{1}{2^k}\right)n && (97)\\
& && (98)
\end{aligned}
$$

$\frac{9n}{10(2^{k-1})} = 1 \implies 9n = 10(2^{k-1}) \implies 9n = 5(2^k) \implies \frac{9n}{5} = 2^k \implies \log_2(\frac{9n}{5}) = \log_2(2^k) \implies \log_2(\frac{9n}{5}) = k\log_2(2) \implies \log_2(\frac{9n}{5}) = k$

$$
\begin{aligned}
&= T(1) + \left(2 - \frac{1}{2^{\log_2(\frac{9n}{5})}}\right)n && (99)\\
&= T(1) + \left(\frac{2^{\log_2(\frac{9n}{5})} - 1}{2^{\log_2(\frac{9n}{5})}}\right)n && (100)\\
&= T(1) + \left(\frac{\frac{9n}{5} - 1}{\frac{9n}{5}}\right) && (101)\\
&= T(1) + n - \frac{5}{9} && (102)\\
& && (103)
\end{aligned}
$$

$\therefore T(n) = O\left(n\right)$

Solución aplicando el teorema maestro:

$a = 1, b = \frac{10}{9}, f(n) = n$

$n^{\log_b(a)} = n^{\log_{\frac{10}{9}}(1)} = n^0 = 1$

aplica el caso 3:

$T(n) = O\left(f(n)\right)$

$\therefore T(n) = O\left(n\right)$

- $T(n) = 16T\left(\frac{n}{4}\right) + n^2$

Solución:

$$
\begin{aligned}
T(n) &= 16T\left(\frac{n}{4}\right) + n^2 && (104)\\
&= 16\left[16T\left(\frac{n}{\frac{4}{4}}\right) + \left(\frac{n}{2}\right)^2\right] + n^2 && (105)\\
&= 16^2T\left(\frac{n}{4^2}\right) + \frac{16n^2}{2^2} + n^2 && (106)\\
&= 16^2\left[16T\left(\frac{n}{\frac{4^2}{4}}\right) + \left(\frac{n}{2^2}\right)^2\right] + \frac{16n^2}{2^2} + n^2 && (107)\\
&= 16^3T\left(\frac{n}{4^3}\right) + \frac{16^2n^2}{2^4} + \frac{16n^2}{2^2} + n^2 && (108)\\
&= 16^3\left[16T\left(\frac{n}{\frac{4^3}{4}}\right) + \left(\frac{n^2}{2^3}\right)^2\right] + \frac{16^2n^2}{2^4} + \frac{16n^2}{2^2} + n^2 && (109)\\
&= 16^4T\left(\frac{n}{4^4}\right) + \frac{16^3n^2}{2^6} + \frac{16^2n^2}{2^4} + \frac{16n^2}{2^2} + n^2 && (110)\\
&= 16^4T\left(\frac{n}{4^4}\right) + \left(\frac{16^3}{2^6} + \frac{16^2}{2^4} + \frac{16}{2^2} + 1\right)n^2 && (111)\\
&= ... && (112)\\
&= 16^kT\left(\frac{n}{4^k}\right) + \left(\sum_{i=0}^{k-1}\frac{16^i}{2^{2i}}\right)n^2 && (113)\\
&= 16^kT\left(\frac{n}{4^k}\right) + \left(\frac{1}{3}(4^k - 1)\right)n^2 && (114)\\
& && (115)
\end{aligned}
$$

$\frac{n}{4^k} = 1 \Longrightarrow n = 4^k \Longrightarrow \log_4(n) = \log_4(4^k) \Longrightarrow \log_4(n) = k\log_4(4) \Longrightarrow \log_4(n) = k$

$$
\begin{aligned}
&= 16^{\log_4(n)}T(1) + \left(\frac{1}{3}(4^{\log_4(n)} - 1)\right)n^2 && (116)\\
& && (117)
\end{aligned}
$$

$\therefore T(n) = O\left(n^2\log(n)\right)$

Solución aplicando el teorema maestro:

$a = 16, b = 4, f(n) = n^2$

$n^{\log_b(a)} = n^{\log_4(16)} = n^2$

$n^2 = f(n)$

$n^2 = n^2$

aplica el caso 2:

$T(n) = O\left(f(n)\log(n)\right)$

$\therefore T(n) = O\left(n^2 \log_(n)\right)$

- $T(n) = 7T\left(\frac{n}{3}\right) + n^2$

Solución:

$$\begin{aligned}
T(n) &= 7T\left(\frac{n}{3}\right) + n^2 && (118)\\
&= 7\left[7T\left(\frac{n}{\frac{3}{3}}\right) + \left(\frac{n}{3}\right)^2\right] + n^2 && (119)\\
&= 7^2T\left(\frac{n}{3^2}\right) + \frac{7n^2}{3^2} + n^2 && (120)\\
&= 7^2\left[7T\left(\frac{n}{\frac{3^2}{3}}\right) + \left(\frac{n}{3^2}\right)^2\right] + \frac{7n^2}{3^2} + n^2 && (121)\\
&= 7^3T\left(\frac{n}{3^3}\right) + \frac{7^2n^2}{3^4} + \frac{7n^2}{3^2} + n^2 && (122)\\
&= 7^3T\left(\frac{n}{3^3}\right) + \left(\frac{7^2}{3^4} + \frac{7^1}{3^2} + 7\right)n^2 && (123)\\
&= ... && (124)\\
&= 7^kT\left(\frac{n}{3^k}\right) + \left(\sum_{i=0}^{k-1}\frac{7^i}{3^{2i}}\right)n^2 && (125)\\
&= 7^kT\left(\frac{n}{3^k}\right) + \left(-\frac{9}{2}\left(\frac{7}{2}\right)^k - 1\right)n^2 && (126)
\end{aligned}$$

$\frac{n}{3^k} = 1 \Longrightarrow n = 3^k \Longrightarrow \log_3(n) = \log_3(3^k) \Longrightarrow \log_3(n) = k\log_3(3) \Longrightarrow \log_3(n) = k$

$$\begin{aligned}
&= 7^{\log_3(n)}T(1) + \left(-\frac{9}{2}\left(\frac{7}{2}\right)^{\log_3(n)} - 1\right)n^2 && (127)\\
&= 7^{\log_3(n)}T(1) - n^2 - 9n^2 2^{-\log_3(n)-1}7^{\log_3(n)} && (128)\\
& && (129)
\end{aligned}$$

$\therefore T(n) = O\left(n^2\right)$

Solución aplicando el teorema maestro:

$a = 7, b = 3, f(n) = n^2$

$n^{\log_b(a)} = n^{\log_3(7)} = n^{1,77..}$

$n^{1,77..} < f(n)$

$n^{1,77..} < n^2$

aplica el caso 3:

$T(n) = O\left(f(n)\right)$

$\therefore T(n) = O\left(n^2\right)$

- $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$
  Solución:

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{4}\right) + \sqrt{n} && (130)\\ &= 2\left[2T\left(\frac{n}{\frac{4}{4}}\right) + \sqrt{\frac{n}{4}}\right] + \sqrt{n} && (131)\\ &= 2^2T\left(\frac{n}{4^2}\right) + \frac{2\sqrt{n}}{\sqrt{n}} + \sqrt{n} && (132)\\ &= 2^2\left[2T\left(\frac{n}{\frac{4^2}{4}}\right) + 2^2\sqrt{\frac{n}{4^2}}\right] + \frac{2\sqrt{n}}{\sqrt{n}} + \sqrt{n} && (133)\\ &= 2^3T\left(\frac{n}{4^3}\right) + 2^2\sqrt{\frac{n}{4^2}} + \frac{2\sqrt{n}}{\sqrt{n}} + \sqrt{n} && (134)\\ &= 2^3T\left(\frac{n}{4^3}\right) + \sqrt{n} + \sqrt{n} + \sqrt{n} && (135)\\ &= 2^kT\left(\frac{n}{4^k}\right) + k\sqrt{n} && (136)\\ & && (137) \end{aligned}$$

$\frac{n}{4^k} = 1 \Longrightarrow n = 4^k \Longrightarrow \log_4(n) = \log_4(4^k) \Longrightarrow \log_4(n) = k\log_4(4) \Longrightarrow \log_4(n) = k$

$$\begin{aligned} &= 2^{\log_4(n)}T(1) + \log_{_{4(n)\sqrt{n}}} && (138)\\ &= n^{\log_4(2)}T(1) + \log_{_{4(n)\sqrt{n}}} && (139)\\ &= \sqrt{n}T(1) + \log_{_{4(n)\sqrt{n}}} && (140)\\ & && (141) \end{aligned}$$

$\therefore T(n) = O\left(\sqrt{n}\log(n)\right)$
Solución aplicando el teorema maestro:
$a = 2, b = 4, f(n) = \sqrt{n}, k = \frac{1}{2}$

$n^{\log_b(a)} = n^{\log_4(2)} = n^{\frac{1}{2}}$

$b^k = 4^{\frac{1}{2}} = 2$

$a = b^k \Longrightarrow 2 = 4^{\frac{1}{2}}$

$2 = 2$

aplica el caso 2:

$T(n) = O\left(n^k\log(n)\right)$

$\therefore T(n) = O\left(\sqrt{n}\log(n)\right)$

- $T(n) = 4T\left(\frac{n}{2}\right) + n^2\log n$
  Solución:

$$\begin{aligned} T(n) &= 4T\left(\frac{n}{2}\right) + n^2\log n && (142)\\ &= 4^2T\left(\frac{n}{2^2}\right) + n^2\log(n) + n^2\log(\frac{n}{2}) && (143) \end{aligned}$$

$$= \quad ... \tag{144}$$
$$= \quad 4^k T\left(\frac{n}{2^k}\right) + n^2\left(\sum_{i=0}^{k-1} \log \frac{n}{2^i}\right) \tag{145}$$
$$\tag{146}$$

$\frac{n}{2^k} = 1 \Longrightarrow n = 2^k \Longrightarrow \log_2(n) = \log_2(2^k) \Longrightarrow \log_2(n) = k\log_2(2) \Longrightarrow \log_2(n) = k$

$$\left(\sum_{i=0}^{k-1} \log \frac{n}{2^i}\right) = \left(\sum_{i=0}^{k-1} (\log(n) - i)\right) \tag{147}$$
$$= (k-1)\log(n) - \frac{(k-1)(k-2)}{2} \tag{148}$$
$$= k\log(n) \tag{149}$$
$$\tag{150}$$

Reemplazando:

$$= \quad 4^k T\left(\frac{n}{2^k}\right) + n^2(k\log(n)) \tag{151}$$
$$= \quad 4^{\log_2(n)} T(1) + n^2(\log_2(n)\log(n)) \tag{152}$$
$$= \quad n^{\log_2(4)} T(1) + n^2(\log_2(n)\log(n)) \tag{153}$$
$$= \quad n^2 T(1) + n^2(\log^2(n)) \tag{154}$$

$\therefore T(n) = O\left(n^2(\log^2(n))\right)$

Solución aplicando el teorema maestro:

$a = 4, b = 2, f(n) = n^2(\log(n))$

$n^{\log_b(a)} = n^{\log_2(4)} = n^2$

$i.e. f(n) = \Omega(n^{\log_b(a)})$

Ahora, para aplicar el caso 3 del teorema maestro, necesitamos una constante positiva tal que $f(n) = \Omega(n^{\log_b(a)+\varepsilon})$ Pero aquí no podemos obtener tal constante y, por lo tanto, no podemos aplicar el teorema maestro., sin embargo podemos aplicar el teorema maestro extendido:

$a = 4, b = 2, k = 2, p = 1$

$si a = b^k entonces$

$si p > -1$ entonces $T(n) = O\left(n^{\log_b(a)\log^{p+1} n}\right)$

$4 = 2^2$

$\therefore T(n) = O\left(n^2(\log^2(n))\right)$

- $T(n) = T(n-1) + n$

  Solución:

$$T(n) = \quad T(n-1) + n \tag{155}$$
$$= \quad T(n-2) + (n-1) + n \tag{156}$$
$$= \quad T(n-3) + (n-2) + (n-1) + n \tag{157}$$
$$= \quad ... \tag{158}$$

$$
\begin{aligned}
&= T(n-k) + (n-(k-1)) + ... + n && (159)\\
&= 0 + 1 + 2 + 3... + n - 1 + n && (160)\\
\sum_{i=0}^{k} n &= \frac{n(n+1)}{2} && (161)\\
&= \frac{n^2}{2} + \frac{n}{2} && (162)\\
& && (163)
\end{aligned}
$$

$\therefore T(n) = O\left(n^2\right)$

Solución aplicando el teorema maestro:

La fórmula no tiene la forma para poder aplicar el teorema maestro, por lo tanto no se podrá usar el teorema en esta recurrencia.

- $T(n) = \sqrt{n} + 1$

  Solución:

$$
\begin{aligned}
T(n) &= \sqrt{n} + 1 && (164)\\
&= T(n^{\frac{1}{2}}) + 1 && (165)\\
&= T(n^{\frac{1}{2}})^{\frac{1}{2}} + 1 + 1 && (166)\\
&= T(n^{\frac{1}{4}}) + 1 + 1 && (167)\\
&= T(n^{\frac{1}{4}})^{\frac{1}{2}} + 1 + 1 + 1 && (168)\\
&= T(n^{\frac{1}{8}}) + 1 + 1 + 1 && (169)\\
&= ... && (170)\\
&= T(n^{\frac{1}{2^k}}) + k && (171)\\
& && (172)
\end{aligned}
$$

$n^{\frac{1}{2^k}} = 2 \Longrightarrow \log(n^{\frac{1}{2^k}}) = \log 2 \Longrightarrow \frac{1}{2^k} = \frac{\log 2}{\log n} \Longrightarrow 2^k = \log_2(n)$

$\log 2^k = \log \log_2(n) \Longrightarrow k \log(2) = \log \log_2(n)$

$k = \frac{\log \log_2(n)}{\log(2)} \Longrightarrow k = \log_2 \log_2(n)$

$$
\begin{aligned}
& T(2) + \log_2 \log_2(n) && (173)\\
& && (174)
\end{aligned}
$$

$\therefore T(n) = O\left(\log \log(n)\right)$

Solución aplicando el teorema maestro:

La fórmula no tiene la forma para poder aplicar el teorema maestro, por lo tanto no se podrá usar el teorema en esta recurrencia.

---

4. Ejercicio $4[5pt]$

Se sabe que, para resolver el problema de ordenación de números, el algoritmo Mergesort posee una complejidad de $\Theta(n \log n)$ y el Insertion Sort $O\left(n^2\right)$. 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.

Algorithm 1 InsertionSort(array[...], p, r)

Require: vector arr[...] desordenado
Ensure: vector arr[...] ordenado

```
for j ← p + 1 to r do
    key ← arr[j]
    //insert arr[j] into the sorted sequence arr[1..j-1]
    i ← j − 1
    while i ≥ p and arr[i] > key do
        arr[i + 1] ← arr[i]
        i ← i − 1
    end while
    arr[i + 1] = key
end for
```

Algorithm 2 merge(array[...], p, q, r)

```
n_1 ← q − p + 1
n_2 ← r − q
create arrays L[1..n_1 + 1] and R[1..n_2 + 1]
for i ← 1 to n_1 do
    L[i] ← arr[p + i − 1]
end for
for j ← 1 to n_2 do
    R[j] ← arr[q + j]
end for
L[n_1 + 1] ← ∞
R[n_2 + 1] ← ∞
i ← 1
j ← 1
for k ← p to r do
    if L[i] ≤ R[j] then
        arr[k] ← L[i]
        i ← i + 1
    else
        arr[k] ← R[j]
        j ← j + 1
    end if
end for
```

Algorithm 3 mergeSort(array[...], p , r)

Require: vector arr[...] desordenado
Ensure: vector arr[...] ordenado
1: **if** $p < r$ **then**
2: $q \leftarrow \lfloor (p+r)/2 \rfloor$
3: $mergeSort(arr, p, q)$
4: $mergeSort(arr, q+1, r)$
5: $merge(arr, p, q, r)$
6: **end if**

Algorithm 4 HybridMergeSort(array[...], p , r)

Require: vector arr[...] desordenado
Ensure: vector arr[...] ordenado
1: **if** $r - p < k$ **then**
2: $InsertionSort(arr, p, r)$
3: **else**
4: $q \leftarrow \lfloor (p+r)/2 \rfloor$
5: $mergeSort(arr, p, q)$
6: $mergeSort(arr, q+1, r)$
7: $merge(arr, p, q, r)$
8: **end if**

Código en C++

```cpp
#include <iostream>
#include <cstdlib>

using namespace std;

int K = 33;

void print(int *arr, int size){
    for(int i = 0; i < size - 1; ++i)
        cout << *(arr+i) << " | ";
}

void insertionSort(int *arr, int p, int r) {
    int i, j;
    int key;
    for (i = p + 1; i <= r; ++i) {
        key = *(arr+i);
        j = i - 1;
        while (j >= p and *(arr+j) > key) {
            arr[j + 1] = *(arr+j);
            --j;
        }
        arr[j + 1] = key;
    }
}

void Merge(int *arr, int p, int q, int r){
    int n1 = q - p + 1;
    int n2 = r - q;

    int *L = new int[n1 + 1];
    int *R = new int[n2 + 1];

    unsigned i = 0;
    for(; i < n1; ++i)
        L[i] = arr[p + i];

    unsigned j = 0;
    for(; j < n2; ++j)
```

```
        R[j] = arr[q + 1 + j];

    L[n1] = INT_MAX;
    R[n2] = INT_MAX;
    i = j = 0;

    unsigned k = p;
    for(; k <= r; ++k){
        if(L[i] <= R[j])
            arr[k] = L[i++];
        else
            arr[k] = R[j++];
    }
}

void HybridMergesort(int *arr, int p, int r){
    if(r - p < K)
        insertionSort(arr, p, r);
    if(p < r){
        int mid = (r + p) >> 1;
        HybridMergesort(arr, p, mid);
        HybridMergesort(arr, mid + 1, r);
        Merge(arr, p, mid, r);
    }
}

int main(){

    int size = 200, arr[size];

    for(long long i = 0; i < size; ++i)
        *(arr+i) = rand()%size;
    //print(arr, size); cout << endl;
    HybridMergesort(arr, 0, size - 1);
    print(arr, size);
}
```

- En la práctica (realice un análisis estadístico) cual es el valor de k.

  En la práctica, k debería ser la longitud de lista más grande en la que el insertion sort es más rápida que el merge sort.

  La complejidad temporal del insertion sort es $= c_1 n^2$ y la complejidad temporal del merge sort es $c_2 n \log(n)$. Para encontrar el valor de k.

  $c_1 k^2 \leq c_2 k \log(k)$

  $k \leq (\frac{c2}{c1}) \log(k)$

  Divide:

  – Primero dividimos estos N elementos en $\frac{n}{k}$ grupos de tamaño K

  Clasificación:

  – Para cada división de subarreglo de tamaño K, realice la operación de ordenación por inserción para ordenar este subarreglo.

  – El costo total del insertion sort para un solo bloque de K elementos es de $O(k)$ para el mejor de los casos y $O(k^2)$ para el peor de los casos.

  – Dado que hay $\frac{n}{k}$ bloques de este tipo, cada uno de tamaño K, obtenemos el costo total de aplicar el insertion sort como $\frac{n}{k} k = O(n)$ para el mejor de los casos y $\frac{n}{k} k^2 = O(nk)$

  Fusionando:

– Tomamos i iteraciones para el merge sort. Entonces, para que el ciclo se detenga se iguala lo sigueinte:

$$\begin{aligned} 2^i k &= n && (175)\\ 2^i &= \frac{n}{k} && (176)\\ i^{\log(2)} &= \log(\frac{n}{k}) \textit{tomando log en ambos lados} && (177)\\ i &= \log(\frac{n}{k}) && (178)\\ &= \log(\frac{n}{k}n) && (179)\\ &= n\log(\frac{n}{k}) && (180)\\ & && (181) \end{aligned}$$

– El costo total del algoritmo (merge + insertion) es $n + n\log(\frac{n}{k})$ para elmejor caso y $nk + n\log(\frac{n}{k})$ para el pero caso.

– Si $k = 1$, entonces es un merge sort completo, que es mejor en términos de complejidad de tiempo

– Si $k = n$, entonces es un insertion sort completo, que es mejor en términos de complejidad de espacio

- Muestre que las $\frac{n}{k}$ sub-listas, cada una de tamaño k, pueden ser ordenadas por el algoritmo Insertion sort en el peor caso en $O(nk)$.

  Tenga en cuenta que ordenar cada lista toma $ak^2 + bk + c$ para algunas constantes a, b y c. Tenemos $\frac{n}{k}$, por lo tanto:

$$\frac{n}{k}(ak^2 + bk + c) = ank + bn + \frac{cn}{k} = \theta(nk) \quad (182)$$

- Muestre que las listas pueden ser combinadas en el peor caso en tiempo $O\left(n\log\left(\frac{n}{k}\right)\right)$.

  $$T(n) = \begin{cases} 0 \textit{ ,si } a = 1 \\ \\ 2T(\frac{a}{2}) + ak \textit{ ,si } a = 2^p, \textit{ si } p > 0 \end{cases}$$

  Esto tiene sentido, ya que fusionar una sublista es trivial y fusionar a sublistas significa dividirlas en dos grupos de $\frac{a}{2}$ listas, fusionar cada grupo recursivamente y luego combinar los resultados en k pasos, ya que tiene dos matrices, cada una de longitud $\frac{a}{2}k$.

  Probamos por induccion:

  $T(1) = 1k\log(1) = k * 0 = 0$
  Supongamos que $T(a) = ak\log(a)$ y calculamos $T(2a)$:

$$\begin{aligned} T(2a) &= 2T(a) + 2ak && (183)\\ &= 2(T(a) + ak) && (184)\\ &= 2(ak\log(a) + ak) && (185)\\ &= 2ak(\log(a) + 1) && (186)\\ &= 2ak(\log(a) + \log(2)) && (187)\\ &= 2ak\log(2a) && (188) \end{aligned}$$

Esto lo prueba. Ahora, si sustituimos el número de sublistas $\frac{n}{k}$ por a:

$$T(\frac{n}{k}) = \frac{n}{k}k\log(\frac{n}{k}) = n\log(\frac{n}{k}) \tag{189}$$

Si bien esto es exacto solo cuando $\frac{n}{k}$ es una potencia de 2, nos dice que la complejidad temporal general de la fusión es $\Theta(n\lg(\frac{n}{k}))$.

- Sea A = $[1 \dots n]$ un vector de números distintos. Si se cumple que $i < j$ y $A[i] > A[j]$, entonces el par (i, j) es llamado de ïnversión"del vector A. Modifique el algoritmo del Mergesort para encontrar las inversiones de un vector. En anexo del mismo problema coloque el código.

Algorithm 5 mergeInv(array[...], p, q, r)

```
n1 ← q − p + 1
n2 ← r − q
create arrays L[1..n1 + 1] and R[1..n2 + 1]
for i ← 1 to n1 do
    L[i] ← arr[p + i − 1]
end for
for j ← 1 to n2 do
    R[j] ← arr[q + j]
end for
L[n1 + 1] ← ∞
R[n2 + 1] ← ∞
i ← 1
j ← 1
count ← 0
for k ← p to r do
    if L[i] ≤ R[j] then
        arr[k] ← L[i]
        i ← i + 1
    else
        arr[k] ← R[j]
        j ← j + 1
        count ← count + (n1 − i + 1)
    end if
end for
```

Algorithm 6 countInv(array[...], p , r)

Require: vector arr[...] desordenado
Ensure: vector arr[...] ordenado, numero de inversiones

```
if p < r then
    q ← ⌊(p + r)/2⌋
    a ← mergeSort(arr, p, q)
    b ← mergeSort(arr, q + 1, r)
    c ← merge(arr, p, q, r)
    return (a + b + c)
else
    return 0
end if
```

Código en C++

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
```

```
5
6  void print(vector<int>& vec) {
7      for (auto ele : vec)
8          cout << ele << " ";
9      cout << endl;
10 }
11
12 int mergeInv(vector<int>& nums, vector<int>& left, vector<int>& right, int mid) {
13     int i = 0, j = 0, k = 0;
14     int inv_count=0;
15     while ((j < left.size()) && (k < right.size())) {
16         if (left[j] <= right[k]) {
17             nums[i] = left[j];
18             i++;
19             j++;
20         } else {
21             inv_count +=  (left.size() - j);
22             nums[i] = right[k];
23             i++;
24             k++;
25         }
26     }
27     while(j < left.size())  {
28         nums[i] = left[j];
29         i++;
30         j++;
31     }
32
33     while(k < right.size()) {
34         nums[i] = right[k];
35         i++;
36         k++;
37     }
38     return inv_count;
39 }
40
41 int countInv(vector<int>& nums){
42     int inv_count=0;
43     if (nums.size() > 1) {
44         int mid = nums.size()/2;
45         vector<int> left(mid);
46         vector<int> right(nums.size()-mid);
47         copy(nums.begin(), nums.begin() + mid, left.begin());
48         copy(nums.begin() + mid, nums.end(), right.begin());
49         inv_count+=countInv(left);
50         // cout<<inv_count;
51         inv_count+=countInv(right);
52         // cout<<inv_count;
53         inv_count+=mergeInv(nums, left, right, mid);
54         // cout<<inv_count;
55         if(inv_count==14)
56             print(nums);
57     }
58     return inv_count;
59 }
60
61 int main(){
62     int n;
63     vector<int> numvec{4, 5, 6, 1, 2, 3};
64     n = countInv(numvec);
65     cout << "numero de inversiones: " << n << endl;
66     //print(numvec);
67     numvec = {1, 2, 3, 4, 5, 6};
68     n = countInv(numvec);
69     cout << "numero de inversiones: " << n << endl;
70     // print(numvec);
71     numvec = {6, 5, 4, 3, 2, 1};
72     n = countInv(numvec);
73     cout << "numero de inversiones: " << n << endl;
74     // print(numvec);
75     numvec = {0, 0, 0, 0, 0, 0};
```

```
76      n = countInv(numvec);
77      cout << "numero de inversiones: " << n << endl;
78  }
```

5. Ejercicio $5[5pt]$

Para los problemas presentados, cree un código en $C++[4$ pts $]$, 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 *ab*. Adicionalmente, deberán correr y colocar screenshots del código corriendo con las instancias que se les pida por cada problema.
'https://nasa.github.io/nasa-latex-docs/html/examples/listing.htmlhttps://nasa.github.io/nasa-latex-docs/html/examples/listing.html
https://texdoc.org/serve/listings.pdf/0https://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: $n = [5, 7, 11, 19, 23]$.

  Solución

Algorithm 7 Descomponer(arr[...], suma, max, b)

Require: vector vacio arr[...], suma, maximo max, tope b
Ensure: vector arr[...] con factores descompuestos

```
1:  if suma == 0 then
2:      print(arr, b)
3:  end if
4:  for i ← max to 1 do
5:      if i ≥ max then
6:          continue
7:      else if i ≤ suma then
8:          arr[b] = i
9:          Descomponer(arr, suma − i, i, b + 1)
10:     end if
11: end for
```

Análisis:
línea 1: O(1)
línea 2: O(n)
línea 4 al 8: O(n)
línea 9: T(n-1)

$$T(n) = \begin{cases} C \ ,si \ n = 1 \\ \\ T(n-1) + 2n \ ,si \ n > 1 \end{cases}$$

$$\begin{aligned} T(n) &= T(n-1) + 2n && (190)\\ &= [T(n-1-1) + 2n - 1] + 2n && (191)\\ &= [T(n-2) + 2n - 1 + 2n] && (192)\\ &= [T(n-2-1) + 2n - 1 - 1] + 2n - 1 + 2n && (193)\\ &= [T(n-3) + 2n - 2 + 2n - 1 + 2n] && (194)\\ &= T(n-3) + 6n - 3 && (195)\\ &= T(n-k) + 2kn - k && (196) \end{aligned}$$

$n - k = 1 \Longrightarrow n - 1 = k$

$$\begin{aligned} &= T(1) + 2(n-1) - (n-1) && (197)\\ &= T(1) + 2n - 2 - n - 1 && (198)\\ & && (199) \end{aligned}$$

$\therefore T(n) = O(n)$

Código en C++

```cpp
#include <iostream>

using namespace std;

void print(int *arr, int b){
    for (int i = 1; i < b; ++i)
        cout << *(arr + i) << " ";
    cout << endl;
}

void algorithm1(int *arr, int suma, int max, int b){
    if(suma == 0){
        print(arr, b);
        return;
    }
    for (int i = max; i >= 1; --i){
        if (i > suma)
            continue;
        else if(i <= suma){
            *(arr + b) = i;
            algorithm1(arr, suma - i, i, b + 1);
        }
    }
}

int main(){
    int a = 5, b = 1, arr[300];
    algorithm1(arr, a, a, b);
    /*int b = 7;
    algorithm1(b, b, 1);
    int c = 11;
    algorithm1(c, c, 1);
    int d = 19;
    algorithm1(d, d, 1);
    int e = 23;
    algorithm1(e, e, 1);*/
}
```

Capturas de pantalla:

Figura 1: ejecución del código de descomposición con a = 5

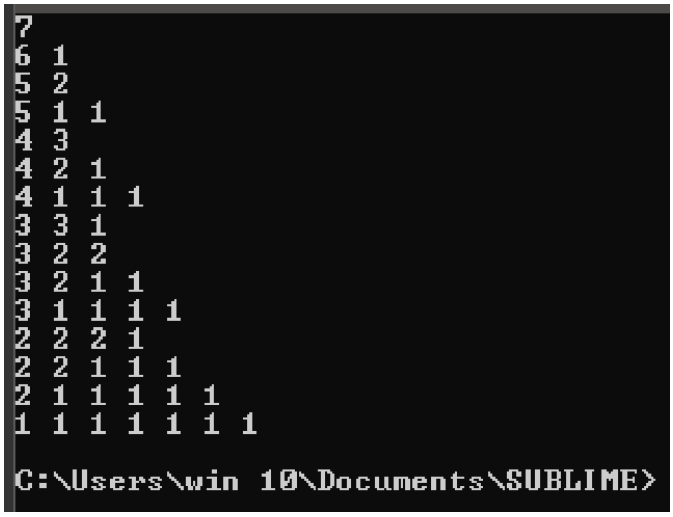

Figura 2: ejecución del código de descomposición con a = 7

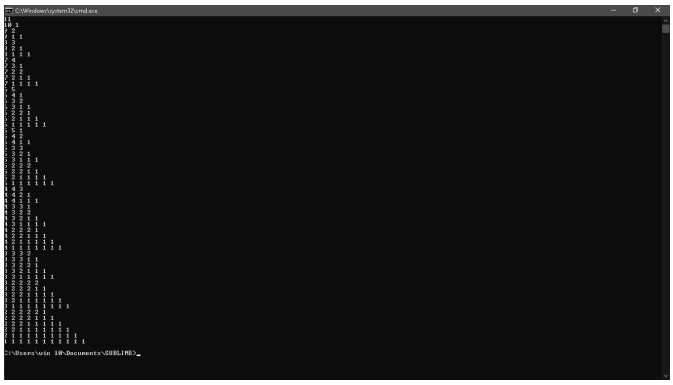

Figura 3: ejecución del código de descomposición con a = 11

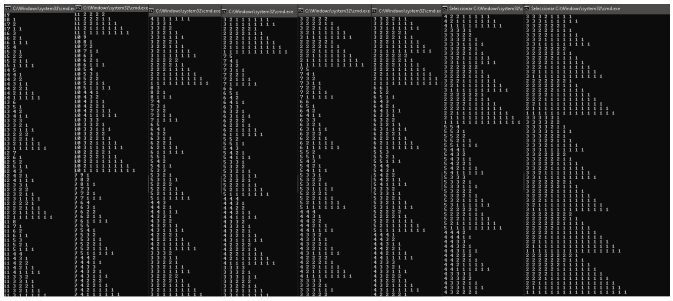

Figura 4: ejecución del código de descomposición con a = 19

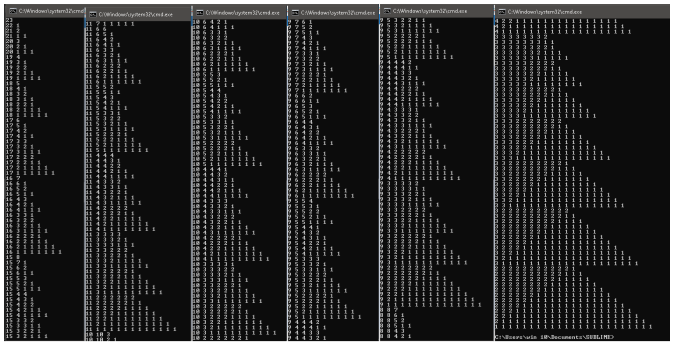

Figura 5: ejecución del código de descomposición con a = 23

- 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.

  Solución:

Algorithm 8 maxSum(arr[...], size)

Require: vector lleno arr[...], tamaño size
Ensure: numero con la suma máxima

1: $r \leftarrow 0$
2: $t \leftarrow arr[0]$
3: $p \leftarrow 0$
4: $q \leftarrow 0$
5: **for** $i = 1$ to $size$ **do**
6: $t \leftarrow t + arr[i]$
7: **if** $t > r$ **then**
8: $r \leftarrow t$
9: $q \leftarrow i$
10: **else if** $t < 0$ **then**
11: $t \leftarrow 0$
12: $p \leftarrow i + 1$
13: **end if**
14: **end for**
15: **return** r

Código en C++

```
#include <iostream>
using namespace std;
```

```
3
4  int maxSuma(int *a, int n){
5
6      int r = 0, t = a[0], p = 0, q = 0;  ////  C
7
8      for (int i = 1; i < n; ++i){        ////  n
9          t += *(a+i);                    ////  n
10         if(t > r){                      ////  n
11             r = t;                      ////  n
12             q = i;                      ////  n
13         }else if (t < 0){               ////  n
14             t = 0;                      ////  n
15             p = i + 1;                  ////  n
16         }                               ////  n
17     }                                   ////  n
18
19     return r;                           ////  C
20 }
21
22 int main(){
23     //int A[11] = {31, -41, 59, 26, -53, 58, 97, -93, -23, 84}, n = 11;
24     //int A[11] = {0, 0, 0, 0, 0, 0, 0, 0, -1, 1}, n = 11;
25     //int A[11] = {1, 1, 1, 1, 1, 1, 0, 1, -1, 2}, n = 11;
26     //int A[11] = {  1  , 1,   1  , 1,   1  , 1,   1  , 1,   1  , 1000}, n = 11;
27     //int A[11] = {23, 1, 12, 11, 41, 22, 18, 4, 2, 6}, n = 11;
28     cout << maxSuma(A, n);
29 }
```

Análisis:
Línea 1 al 4: O(1)
Línea 5 al 12: O(n)
Línea 1: O(1)

$\therefore O(n)$

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. $A = [23, 1, 12, 11, 41, 22, 18, 4, 2, 6]$

Captura de pantalla:

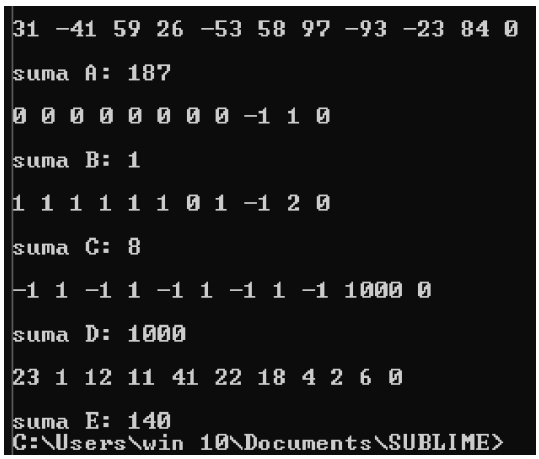

Figura 6: ejecución del código de máxima suma