TEORÍA DE ALGORITMOS I

TP1

ALUMNO: MARÍA PAULA BRÜCK

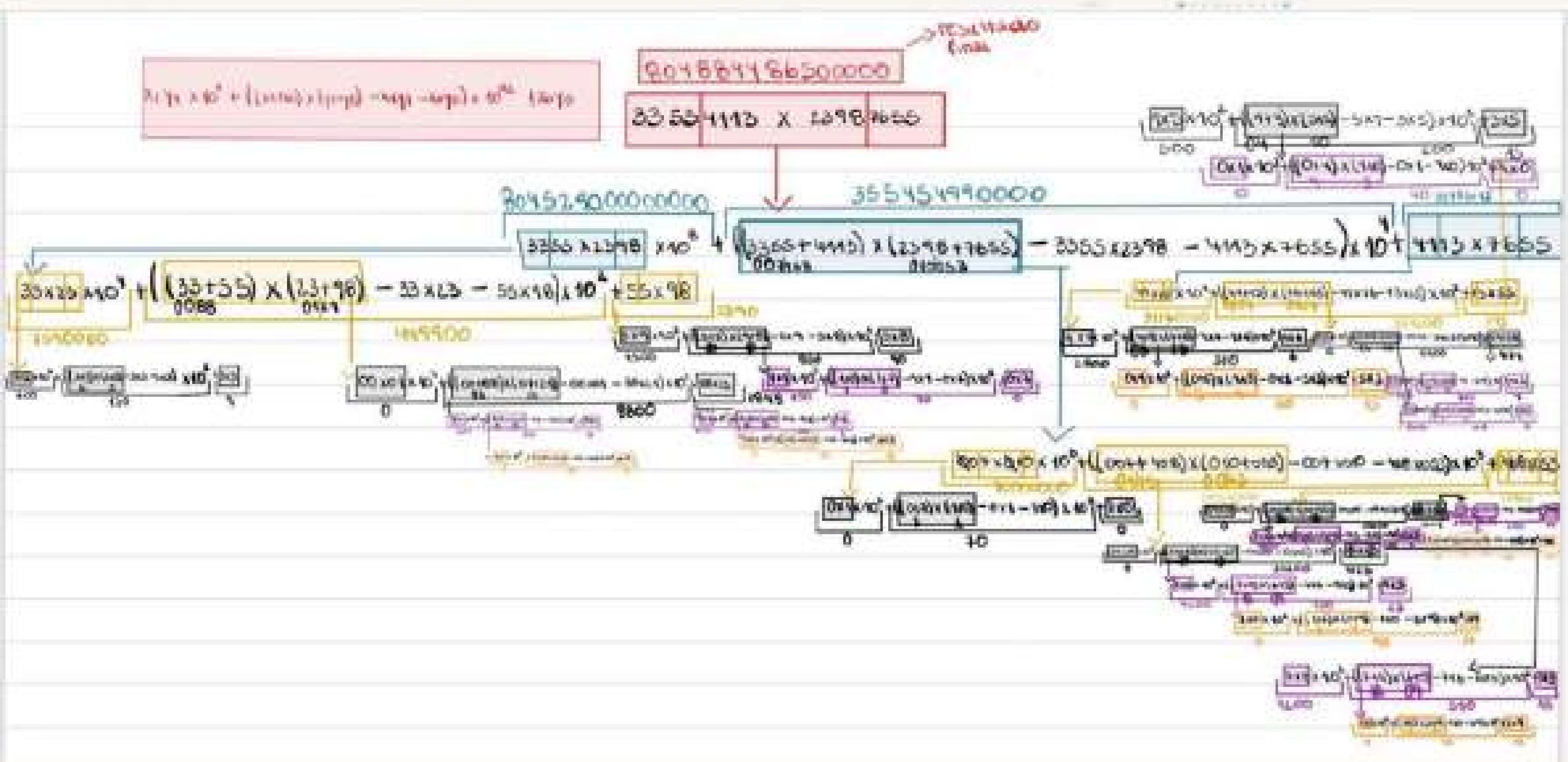
PADRON: 107533

FECHA DE ENTREGA: 27/09/2021

PARTE 1

- NUMEROS: 33554113
  23987655

1) MULTIPLICACION KARATSUBA



2) COMPLEJIDAD TEMPORAL

* El metodo de multiplicacion de Karatsuba conlleva reducir las multiplicaciones aumentando el numero de sumas. En vez de hacer 4 multiplicaciones como en el metodo tradicional se reducen a $\boxed{3}$ sin embargo el numero de sumas es mayor siendo este $\boxed{4}$. Este algoritmo reduce la complejidad temporal de $O(n^2)$ como es utilizando el metodo tradicional a $O(n^{1.59})$. Esto se puede demostrar a traves del teorema maestro:

→ al tener 13 casos base se realizaron 39 multiplicaciones como base cuyos resultados fueron combinandose para asi poder llegar a una unica solucion general

por lo tanto la cantidad total de sumas en los casos base $13 \times 4 = \boxed{52}$

Nuestra relacion de recurrencia seria: $T(n) = \underbrace{3}_{}T(n/2) + cn$

siendo este el numero de multiplicaciones
$$\begin{cases} x_1 y_1 \\ (x_1+x_2)(y_1+y_2) \\ x_2 y_2 \end{cases}$$

$$\boxed{(10^n)\, x_1 y_1 + ((x_1 y_2) + (x_2 y_1))\, 10^{n/2} + x_2 y_2}$$

$$T(n) = 3^k \times 1 + \sum_{i=0}^{k} \frac{3^i \times n}{2^i}$$

$k = \log_2 n$ } cant de veces que hago la sumatoria

$$T(n) = 3^{\log(n)} + n \times \sum_{i=0}^{\log(n)} \left(\frac{3}{2}\right)^i$$

usando $\sum_{i=0}^{k} r^i = \frac{1 - r^{\log(n)+1}}{1-r}$

$$T(n) = 3^{\log(n)} + n \times \frac{\left(1 - (3/2)^{\log(n)+1}\right)}{(1 - 3/2)}$$

$$T(n) = 3^{\log(n)} + 2n \times \left((3/2)^{\log(n)+1} - 1\right)$$

$$T(n) = 3^{\log(n)} + 3n \times (3/2)^{\log(n)} - 2n$$

$$T(n) = 3^{\log(n)} + 3n \, \frac{3^{\log(n)}}{2^{\log(n)}} - 2n$$

$$T(n) = 3^{\log(n)} + 3 \times 3^{\log(n)} - 2n$$

$$T(n) = 4 \times 3^{\log(n)} - 2n \longrightarrow \text{usando } a^{\log_2 n} = n^{\log_2 a}$$

$$T(n) = 4 \times n^{\log 3} - 2n$$

si $n \rightarrow \infty$

$$\boxed{O(n^{\log_2 3}) = O(n^{1.59})}$$

3) si comparamos el metodo de Karatsuba con el metodo tradicional podemos afirmar que es mas eficiente ya que para los n grandes reduce la complejidad temporal de $O(n^2)$ a $O(n^{1.59})$.

En el caso resuelto en el punto 1 si hubieramos usado el metodo tradicional hubieramos hecho:

$$\begin{array}{r} 33554113 \\ \times\ 23987655 \\ \hline \end{array} \longrightarrow n = 8 \longrightarrow 64 \text{ multiplicaciones digito x digito}$$

$$O(8^2) \longrightarrow O(n^2)$$

[illegible]

4) El algoritmo de Karatsuba es del tipo divide y conquista ya que el problema se resuelve de manera recursiva dividiendo el problema inicial en 2 subproblemas siendo estos de menor tamaño hasta llegar a un caso base y luego combina las soluciones de los subproblemas en una unica solucion general.

PARTE 2

- Ecuacion de Recurrencia: $T(n) = 2T(n/5) + O(n^2)$

1) Para utilizar teorema maestro nos falta decir que cuando se llegue al caso base este ultimo problema tiene que tener como complejidad de ejecucion una constante $(O(1)) \rightarrow \boxed{T(0) = cte}$

2, 3) Complejidad temporal aplicando Teorema Maestro

* teniendo esta ecuacion de recurrencia podemos decir:

$T(n) = 2T(n/5) + O(n^2)$ con $T(0) = cte$

$a = 2$ (cantidad de subproblemas)

$b = 5$ (fraccion de elementos por subproblema)

$f(n) = n^2$ (separacion y unión de los subproblemas)

* Para poder determinar la complejidad temporal debemos analizar cual de estos 3 casos se cumple

$\boxed{\text{Caso 1}}$ si $\underset{\text{tiene que estar}}{f(n)} = \underbrace{O(n^{\log_b a - e})}_{\text{acotada superiormente}}$, $e > 0 \rightarrow T(n) = \Theta(n^{\log_b a})$

$\boxed{\text{Caso 2}}$ si $\underset{\text{tiene que estar}}{f(n)} = \underbrace{\Theta(n^{\log_b a})}_{\text{acotada superiormente e inferiormente}} \rightarrow T(n) = \Theta(n^{\log_b a} \times \log n)$

$\boxed{\text{Caso 3}}$ si $\underset{\text{tiene que estar}}{f(n)} = \underbrace{\Omega(n^{\log_b a + e})}_{\text{acotada inferiormente}}$, $e > 0 \rightarrow T(n) = \Theta(f(n))$

$a f(n/b) \leq c f(n)$, $c < 1$ para un n suficientemente grande

* Comenzamos probando el $\boxed{\text{caso 2}}$

$n^2 = \Theta(n^{\log_5 2}) \rightarrow n^2 = \underbrace{\Theta(n^{0{,}43})}_{\text{puede ser acotado inferiormente pero no superiormente por lo tanto este caso no nos sirve}}$

*Probamos el $\boxed{\text{caso } 1}$

$n^2 = O(n^{\log_5 2 - e}) \rightarrow n^2 = O(n^{0{,}43 - e})$ si $e = 0{,}43 \rightarrow n^2 = \underbrace{O(n)}$

no se puede acotar superiormente con $e > 0$
por lo tanto este caso tampoco nos sirve

*Probamos finalmente el $\boxed{\text{caso } 3}$

$n^2 = \Omega(n^{\log_5 2 + e})$ si $e = 0{,}1 \rightarrow \underbrace{n^2 = \Omega(n^{0{,}53})}_{\text{puede acotar inferiormente}}$

$\underbrace{\exists c < 1}_{\substack{\text{valor c de} \\ \text{y} < 1}}$, $\underbrace{n >>}_{\substack{\text{n lo} \\ \text{suficientemente} \\ \text{grande}}}$ / $a . f(n/b) \leq c \times f(n) \rightarrow 2((n/5)^2) \leq cn^2$ si $c = 2/5$

$\hookrightarrow \underbrace{2 \frac{n^2}{25} \leq \frac{2}{5} n^2}_{\text{se cumple}}$

*Por lo tanto podemos decir que la complejidad temporal de $T(n) = 2T(n/5) + O(n^2)$ es $T(n) = \Theta(f(n))$, es decir $\boxed{T(n) = \Theta(n^2)}$