

# Tarea 5

## Informatica Teorica

Matias peñaloza 202373037-8

2024-2

Concepto	Tiempo [min]
Revisión	120
Desarrollo	120
Informe	90

## 1 Enunciado

Para los siguientes dé una estimación de lo que significa el tamaño del problema, y justifique que el costo del algoritmo relevante es polinomial. Puede suponer operaciones similares a un lenguaje como C. Suponga que operaciones aritméticas con números de  $k$  bits tienen costo  $O(k)$ , igualmente el costo de acceder a un arreglo con índice de  $k$  bits tiene costo  $O(k)$ .

1. El problema COMPOSITE da un entero  $n$  escrito en binario, pregunta si es compuesto (puede escribirse como  $n = a \cdot b$ , con  $a, b \neq 1$ ). Sabiendo que el producto  $a \cdot b$  de  $a$  y  $b$  expresados en binario puede calcularse en tiempo  $O(\log a \cdot \log b)$ , demuestre que COMPOSITE está en NP.
2. El problema 3CLIQUE da un grafo  $G = (V, E)$ , expresado como matriz de adyacencia, y pregunta si contiene un  $K_3$  (un triángulo) como subgrafo. Demuestre que 3CLIQUE está en P.

## 2 Desarrollo

### 2.1 Problema Composite

Primero transformaremos el concepto a algo más conocido, tomaremos el conjunto de números enteros representados en binario que son compuestos y lo llamaremos el lenguaje  $L_{com}$ . También diremos que el algoritmo que se utiliza para resolver el problema es una máquina de Turing  $M_{com}$  tal que  $L_{com} = L(M_{com})$ . Por último, nuestra máquina  $M_{com}$  tendrá como input una palabra  $\sigma$  que será un número entero representado en binario, y como output tendrá Sí en caso de que  $\sigma \in L_{com}$  y No en caso de que  $\sigma \notin L_{com}$ .

### 2.1.1 Tamaño del problema

El tamaño para las instancias del problema COMPOSITE está dado por el número de bits que se necesitan para representar el número  $n$  en binario; a este tamaño lo llamaremos:

$$N = \log_2(n) = |\sigma|$$

### 2.1.2 Costo del algoritmo relevante

El algoritmo que utilizará nuestra máquina  $M_{com}$  para determinar si aceptar o no será:

1. En una cinta guardaremos una lista de los números enteros menores a  $n$  y mayores a 1.
2. La máquina copiará **de forma no determinista** 2 números de la lista a otra cinta.
3. Se calculará el producto de los números copiados.
4. Se verificará si el producto es finalmente  $n$ .

Note que para cada uno de los pasos los costos relevantes asociados son:

1. Escribir  $n - 2$  números de  $N$  bits.  $O(N(n - 2))$
2. Copiar 2 números de  $N$  bits.  $O(2N)$
3. Calcular el producto de 2 números que son a lo más  $n - 1$ .  $O((\log_2(n - 1))^2)$
4. Verificar si 2 números son iguales es trivial.  $O(1)$

Sumando, obtenemos que el costo del algoritmo es:

$$N(n - 2) + 2N + (\log_2(n - 1))^2 + 1$$

Donde claramente no se aceptan números menores a 4, ya que nuestra lista debe contener al menos dos números. Podemos decir que, para efectos prácticos,  $n \leq N$ . Ahora procederemos a acotar la complejidad algorítmica de la máquina:

$$N(n - 2) + 2N + (\log_2(n - 1))^2 + 1 \leq N \cdot N + 2N + (\log_2(n))^2 \leq N^2 + 2N + N^2$$

Quedando finalmente así el tiempo acotado por el polinomio:

$$T(N) = 2N^2 + 2N$$

### 2.1.3 Demostración Composite en NP

Dado el lenguaje  $L_{com}$  que representa las soluciones a COMPOSITE y nuestra máquina de Turing **no determinista**  $M_{com}$  que acepta  $\sigma \in L_{com}$  (decide) en un tiempo acotado por un polinomio  $T(N)$  donde  $N = |\sigma|$ , podemos decir que COMPOSITE está en NP.

## 2.2 Problema 3Clique

Al igual que en el problema anterior, tendremos nuestro grafo representado en una palabra  $\sigma$  de forma que, por ejemplo, la matriz:

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad (1)$$

es representada de la forma:

$$011.100.100$$

Donde las listas horizontales están separadas por un punto. También tendremos nuestro lenguaje  $L_3$ , que será el conjunto de grafos representados de la forma antes mencionada que contienen al menos un subgrafo  $K_3$ . Finalmente, nuestra máquina de Turing  $M_3$  tal que  $L_3 = L(M_3)$ .

\*Note que, a través de contadores, se puede determinar la posición en  $x$  e  $y$  del enlace en la matriz.

### 2.2.1 Tamaño del problema

El tamaño de las instancias del problema está dado por:

$$N = V^2 + V - 1$$

Donde  $V^2$  es la cantidad de 1's o 0's de la matriz (huecos de la matriz para los enlaces) y  $V - 1$  es la cantidad de puntos (separación vertical de la matriz en listas horizontales).

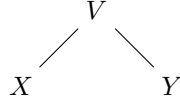
### 2.2.2 Costo del algoritmo relevante

El algoritmo que utilizará nuestra máquina  $M_3$  para determinar si aceptar o no será:

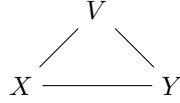
1. Por cada vértice (vértice  $V$ ):
  - (a) Por cada vecino de  $V$  (vecino  $X$ ):
    - i. Por cada vecino siguiente de  $V$  distinto de  $X$  (vecino  $Y$ ):
      - A. Buscaremos al vecino  $Y$  en los enlaces de  $X$ .
      - B. Si encuentra, termina y acepta.
    - ii. Al terminar con el vecino  $Y$ , se toma el siguiente vecino.

- (b) Al terminar con el vecino  $X$ , se toma el siguiente vecino.
- 2. Al terminar con el vértice  $V$ , se toma el siguiente vértice.
- 3. Al terminar con todos los vértices y no haber aceptado, se rechaza.

Lo que se trata de hacer en este algoritmo es buscar dos vecinos ( $X$  e  $Y$ ) de un vértice  $V$ :



Tal que  $X$  e  $Y$  estén conectados para formar un  $K_3$ .



Ahora, para los costos asociados:

1. Por cada lista horizontal.  $O(V)$ 
  - (a), i. Por cada lista horizontal  $X$  e  $Y$ , donde se comparan una vez cada  $X$  con cada  $Y$  sin repetir a los vecinos, siendo el peor de los casos la cantidad de vecinos igual a  $V$ .  $O(\sum_{i=1}^V i)$ 
    - A. Buscar el elemento en la lista  $X$  en el índice  $Y$ , donde el índice máximo es  $V$ .  $O(\log_2(V))$
    - B. Comparar si el elemento es un 1 es trivial.  $O(1)$
  - (b), ii. Se sigue la iteración.
2. Se sigue la iteración, si termina, simplemente no acepta.

Sumando, obtenemos:

$$V * \left( \sum_{i=1}^V i * (\log_2(V) + 1) \right) = V * \frac{V * (V + 1)}{2} * (\log_2(V) + 1)$$

Luego, siendo  $\log_2(V) + 1 \leq V^2$ , procedemos a acotar:

$$V * \frac{V * (V + 1)}{2} * (\log_2(V) + 1) \leq V * V * (V + 1) * V^2 \leq V^3 * V^2$$

Ahora, tomando en cuenta que  $V \neq 0$ , ya que el algoritmo necesita por lo menos 3 nodos, entonces  $V^2 \leq N$  y  $V^3 \leq N^2$ :

$$V^3 * V^2 \leq N^3$$

Quedándonos finalmente el tiempo acotado por el polinomio:

$$T(N) = N^3$$

### 2.2.3 Demostración 3Clique en P

Dado el lenguaje  $L_3$  que representa las soluciones a 3CLIQUE y nuestra máquina de Turing **determinista**  $M_3$ , que acepta  $\sigma \in L_3$  (decide) en un tiempo acotado por un polinomio  $T(N)$  donde  $N = |\sigma|$ , podemos decir que 3CLIQUE está en P.