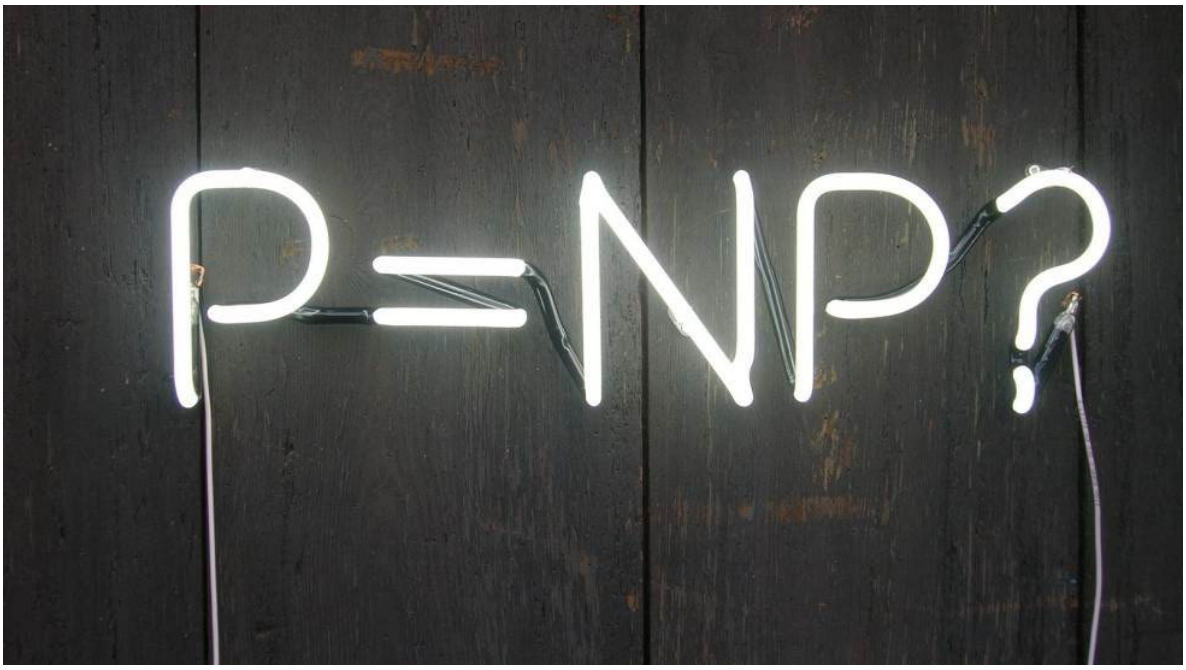
Universidad Nacional Autónoma de México

Facultad de ingeniería



Alumno: Monroy Salazar Diego Gustavo

Materia: Estructura de datos y algoritmos I

Grupo:13

Fecha:07/06/2020

El problema P=NP surge de la necesidad de analizar cuan compleja se vuelve la solución a un problema conforme el tamaño de este vaya creciendo, cosa que es muy importante en la computación y en las matemáticas. Con el pasar de los años fue dando forma a lo que se conoce como clases de complejidad, las cuales engloban ciertos conjuntos de problemas, los cuales tienen subconjuntos que son estrictamente eso, subconjuntos, pero hay otros subconjuntos en los cuales no estamos seguros ni hemos podido probar que son estrictos y no pueden ser iguales. Los problemas P y NP son uno de estos subconjuntos que no conocemos si pueden ser iguales o no.

¿Pero qué es P y qué es NP? El conjunto P viene de “Polynomial time” y se refiere al conjunto de problemas en los que podemos encontrar una respuesta a estos en un tiempo razonable de orden polinomial.

Ejemplos de problemas P pueden ser la multiplicación de matrices o decir si un numero es mayor a otro:

```
void FunMultiplicacion(float arreglo1[][t],float arreglo2[][t], int filas, int columnas)
{
    int f, c, k;
    float temporal, resultado[f][c];
    printf("La multiplicacion de las matrices es:\n");
    for (f=0;f<filas;f++) //i para las filas de la matriz resultante
    {
        for(k=0;k<columnas;k++)
        {
            temporal=0;
            for (c=0;c<filas;c++) //j para realizar la multiplicacion de
            {                                    //los elementos   de la matriz

                temporal=temporal+arreglo1[f][c]*arreglo2[c][k];
                resultado[f][k]=temporal;
            }
        }
    }
```

Aquí se puede ver que el problema se resuelve en un tiempo polinomial $O(n^3)$.

```
double mayor (double x, double y
{
    if ((x>=y))
    {
        return x;//O(1)
    }else
    {
        return y;//O(1)
    }

}
```

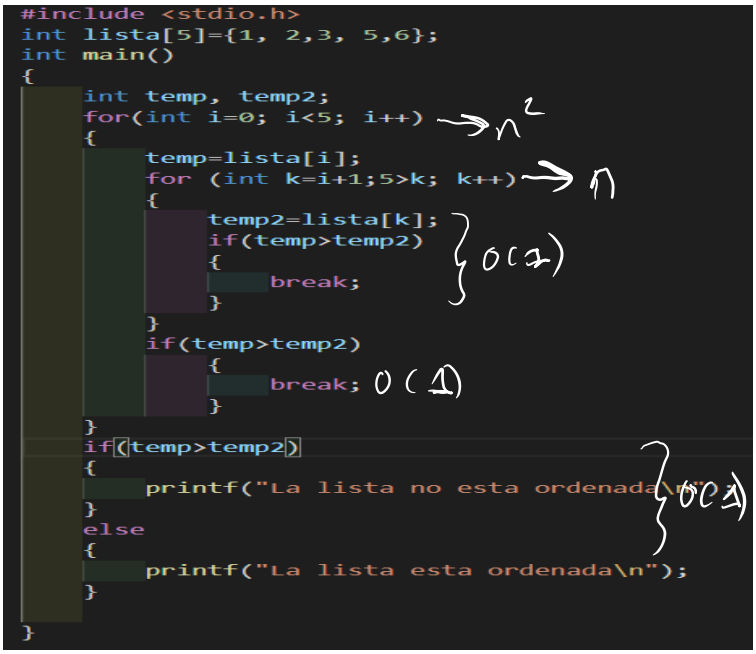
Aquí podemos ver que el problema se resuelve en un tiempo prácticamente constante con una eficiencia O(1).

En cambio e subconjunto NP se refiere a “Non-deterministic Poynomial time” y engloba el conjunto de problemas en los que podemos comprobar si una respuesta es correcta en un tiempo razonable (polinomial).

Ejemplos de estos problemas pueden ser la comprobación de la raíz cuadrada de un número o checar si una lista de números está ordenada.

```
def raiz_numero(numero,raiz)
    ans=raiz*raiz
    if ans==numero:
        return True
```

Aquí podemos fácilmente ver que la comprobación es muy rápida. Con una eficiencia O(1)



Aquí la eficiencia del algoritmo es de $O(n^2)$, por lo que se comprueba en tiempo polinomial la respuesta.

El gran problema con los problemas P y NP tiene que ver con que se hace la siguiente pregunta: Si se puede comprobar la veracidad de una solución de un problema de manera sencilla ¿También se puede encontrar de manera sencilla una solución a este? Ésta es la pregunta de uno de los más grandes problemas a los que se ha enfrentado los científicos computacionales y matemáticos. De hecho, a quien sea capaz de comprobar o refutar el problema P=NP se le premiara con una cantidad bastante generosa de dinero.

A día de hoy mucha gente piensa que la respuesta obvia es que P no es igual a NP, y que simplemente hay problemas en los que no se puede encontrar una solución de manera sencilla. Sin embargo, es importante comprender la implicación de todo esto ya que si se logra responder a esta gran interrogante cambiaria mucho el paradigma de cómo vemos las cosas hoy.