Mejia Valdiviezo Eduardo Javier
Grupo: 13

# Problemas P y NP

El objetivo de este trabajo es dar a conocer específicamente, qué son los problemas P y NP, pero antes de comenzar a hablar de esto, debemos tener bien claro qué es un algoritmo, ya que mediante estos es que se puede resolver los tipos de problemas que se nos presentan

Un algoritmo se puede definir como una secuencia de instrucciones que representan un modelo de solución para determinado tipo de problemas. O bien como un conjunto de instrucciones que realizadas en orden conducen a obtener la solución de un problema.

Para realizar un programa es conveniente el diseño o definición previa del algoritmo. El diseño de algoritmos requiere creatividad y conocimientos profundos de la técnica de programación. Luis Joyanes, programador experto y autor de muchos libros acerca de lógica y programación nos dice “en la ciencia de la computación y en la programación, los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es sólo un medio para expresar un algoritmo y una computadora es sólo un procesador para ejecutarlo”.

Loa algoritmos son independientes de los lenguajes de programación. En cada problema el algoritmo puede escribirse y luego ejecutarse en un lenguaje diferente de programación. El algoritmo es la infraestructura de cualquier solución, escrita luego en cualquier lenguaje de programación.

Características de los algoritmos

- Preciso. Definirse de manera rigurosa, sin dar lugar a ambigüedades.
- Definido. Si se sigue un algoritmo dos veces, se obtendrá el mismo

  resultado.

- Finito. Debe terminar en algún momento.
- Puede tener cero o más elementos de entrada.
- Debe producir un resultado. Los datos de salida serán los resultados de

  efectuar las instrucciones.

Se concluye que un algoritmo debe ser suficiente para resolver el problema. Entre dos algoritmos que lleven a un mismo objetivo, siempre será preferible el más corto (se deberá analizar la optimización de tiempos y / o recursos).

**Problemas P y NP**

Son dos grupos (formalmente dos clases de complejidad) que agrupan problemas decidibles distintos. Es decir, tanto para cualquier problema P como para cualquier NP hay un algoritmo que lo resuelve en un tiempo finito.

Ante un problema podemos hacer dos cosas: calcular los resultados o verificarlos. Por ejemplo, ante el problema “dado x obtener su cuadrado”, podemos hacer dos cosas: crear un algoritmo que encuentre el cuadrado de cualquier x o bien hacer un algoritmo que verifique si un y determinado es el cuadrado de un x.

**Problemas P:**

Hay problemas que son fáciles de solucionar y de verificar. En este contexto “fáciles” significa mediante un algoritmo de orden polinomial. A esos problemas los llamamos problemas de tipo P. Un algoritmo tiene orden polinomial (o es polinómico) cuando dado unos datos de entrada de tamaño N, resuelve el problema en un tiempo que es un polinomio de N.

Así por ejemplo, si tengo un problema que para ordenar listas de N elementos tarda un tiempo que es, pongamos por ejemplo, 10N4+2N2+100N entonces ese agoritmo es polinómico. Ahora bien, si el tiempo que tarda fuese 2N, entonces ya no (ese sería exponencial). Los algoritmos polinómicos los consideramos sencillos en el sentido de que los podemos computar y el tiempo no explota cuando los tamaños de datos aumentan.

| N | $10N^4+2N^2+100N$ | (años) | $2^N$ | (años) |
|---|---|---|---|---|
| 1 | 112 | 3.5515E-06 | 2 | 6.34196E-08 |
| 2 | 368 | 1.16692E-05 | 4 | 1.26839E-07 |
| 10 | 101200 | 0.003209031 | 1024 | 3.24708E-05 |
| 11 | 147752 | 0.004685185 | 2048 | 6.49417E-05 |
| 50 | 62510000 | 1.982179097 | 1.126E+15 | 35702051.84 |
| 100 | 1000030000 | 31.71074328 | 1.268E+30 | 4.01969E+22 |

Observa que como, incluso aunque nuestro algoritmo polinómico aumenta bastante (no en vano es un polinomio de cuarto nivel) se puede ver como la función exponencial aumenta… bueno, eso, exponencialmente (para un elemento tarda 2 segundos y para 100 más de 10 elevado a 22 años).

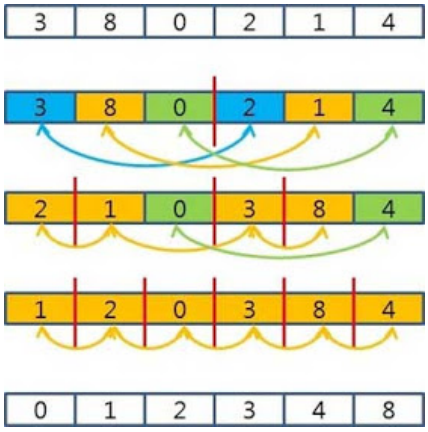
Por lo tanto: aquellos problemas qué podemos solucionar y verificar con un algoritmo de orden polinómico, los llamamos problemas P. Estos son los que podemos solucionar fácilmente.

## Problemas NP:

Los problemas NP son aquellos problemas cuyo algoritmo de solución es de orden superior al polinómico. Una cosa muy característica de estos problemas es que se puede elaborar un algoritmo de orden de complejidad polinómica para determinar si esta solución es en verdad una solución. Estos son problemas difíciles de solucionar pero fáciles de verificar.

Un ejemplo informal es un puzzle: difícil de solucionar pero basta un vistazo para saber si el puzzle está bien resuelto o no. Pues a estos tipos de problemas los llamamos NP.

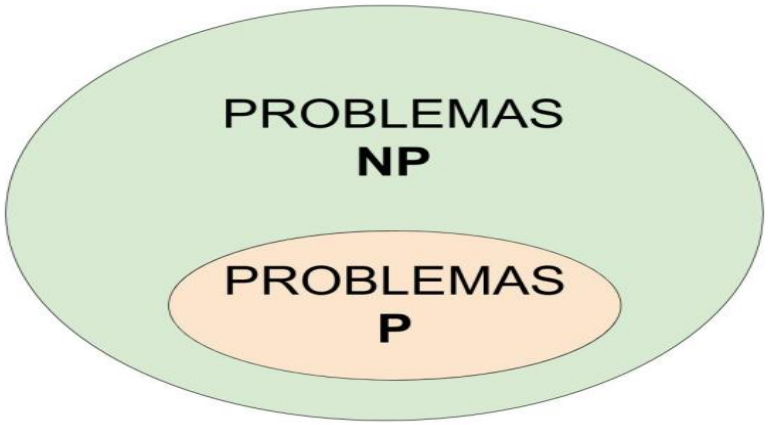
Un ejemplo más formal es la ordenación por el método Shell:




Consiste en dividir un arreglo (o la lista de elementos) en intervalos (o bloques) de varios elementos para organizarlos después por medio del ordenamiento de inserción directa.

Todos los problemas P son a su vez problemas NP, es decir, P es un subconjunto de NP. Debido a que un problema P es aquél que tiene un algoritmo de orden polinómico que lo resuelve. Mientras que un problema NP tiene un algoritmo de orden polinómico que lo verifica. Pues bien, entonces dado un problema de tipo P

puedo verificarl, corriendo el algoritmo que lo soluciona y comprobando la solución. Por lo tanto, un problema de tipo P es, por definición, un NP.




En conclusión, es importante como programadores, conocer las distintas vías quese tienen a la hora de resolver ciertos problemas, debemos saber cómo y cuándo llevarlas a cabo. Se tiene que tener bien presente cómo funcionan los algoritmos ya que son una parte fundamental de los problemas de tipo P y NP.