**Problemas Computables y No Computables**

Es importante saber que existen problemas para los cuales no se pueden conseguir programas que los resuelvan, es decir, que no son computables. Más aun, existen algunos problemas que a pesar de ser computables son tan difíciles, que sus soluciones son demasiado ineficientes (y no pueden ser resueltos eficientemente).

La noción de computabilidad no es (necesariamente) absoluta. Esta noción depende de un modelo de computación elegido.

Un modelo de computación es, a grandes rasgos, una noción abstracta de maquina de cómputo y de algoritmo. Un problema P es computable (bajo un modelo de computación dado) si y solo su existe un algoritmo (sobre la máquina de computo correspondiente) que lo resuelva.

Si nos restringimos a problemas para los cuales existen soluciones algoritmicas, podemos preguntarnos algunas cuestiones importantes. Una de estas preguntas es la siguiente:

*Dado un problema (computable) Prob, cuan eficiente puede Prob ser resuelto?*

Para poder responder esta pregunta necesitamos una noción de costo y paso de computo- Esto no es difícil de definir para cada uno de los modelos de computación:

* número de avances/movimiento sobre la cinta en una máquina de Turing.
* número de reducciones en cálculo lamba
* …

De acuerdo a cuán eficiente pueden ser resueltos los problemas, estos pueden clasificarse en clases.

**p:** Clase de todos los problemas computables que pueden resolverse en un número polinomial de pasos (con respecto al tamaño de la entrada) en una máquina de Turing determinista .

**NP:** Clase de todos los problemas computables que pueden resolverse en un número polinomial de pasos (con respecto al tamaño de la entrada) en una máquina de Turing no determinista. Equivalentemente, esta clase puede definirse como aquella compuesta de todos los problemas que , dada una potencial solución al mismo, la validez de esta puede chequearse en tiempo polinomial por una máquina de Turing determinista.

**EXP:** Clase de todos los problemas computables que pueden resolverse en un número de pasos proporcional a 2^p(n), donde n es el tamaño de la entrada y p es un polinomio, por una máquina de Turing determinista.

**En el caso de que para cada par (estado, símbolo) posible exista a lo sumo una posibilidad de ejecución, se dirá que es una máquina de Turing determinista, mientras que en el caso de que exista al menos un par (estado, símbolo) con más de una posible combinación de actuaciones se dirá que se trata de una máquina de Turing no determinista.**

**X-Completitud:**

Un problema Pr se dice X-completo ⇔ Pr pertenece a X y todo problema en la clase de X puede reducirse a Pr en tiempo polinomial por una máquina de turing determinista.

**Existen Pr EXP-COMPLETOS que no pertenecen a P, son Pr computables que no pueden resolverse en tiempo polinomial.**

**------------------------------------------------------------------------------------------------------------------------**

¿ P=NP?

Una respuesta a la pregunta **P** = **NP** determinaría si los problemas que se pueden verificar en tiempo polinomial, también se pueden resolver en tiempo polinomial.

Pues bien, **P es la clase de complejidad** que contiene problemas de decisión que se pueden resolver en un **tiempo polinómico**. P contiene a la mayoría de problemas naturales, algoritmos de programación lineal, funciones simples,... Por ejemplo la suma de dos números naturales se

resuelven en tiempo polinómico (para ser más exactos es de orden 2n).

La **clase de complejidad NP** contiene problemas que **no pueden resolverse** en un tiempo polinómico. Cuando se dice que un algoritmo no puede obtener una solución a un problema en tiempo polinómico siempre se intenta buscar otro procedimiento que lo consiga mejorar.

-------------------------------------------------------------------------------------------------------------------------