Análisis de Algoritmos

Es difícil realizar un análisis simple de un algoritmo que determine la cantidad exacta de tiempo requerida para ejecutarlo. La primera complicación es que la cantidad exacta de tiempo dependerá de la implementación del algoritmo y de la máquina real en que se ejecuta. El análisis normalmente debe ser independiente de la computadora (hardware y software) y del lenguaje o máquina que se utilice para implementar el algoritmo. La tarea de calcular el tiempo exacto requerido suele ser bastante pesado.

Es conveniente utilizar una función T(n) para representar el número de unidades de tiempo (o tiempo de ejecución del algoritmo) tomadas por un algoritmo de cualquier entrada de tamaño n. La evaluación se podrá hacer desde diferentes puntos de vista:

- **Peor caso.** Se puede hablar de T(n) como el tiempo para el peor caso. Se trata de aquellos ejemplares del problema en los que el algoritmo es menos eficiente (no siempre existe el caso peor). Ejemplos: insertar al final de una lista, ordenar un vector que está ordenado en orden inverso, etc. Nos interesa mucho.
- **Mejor caso.** Se habla de T(n) como el tiempo para el mejor caso. Se trata de aquellos ejemplares del problema en los que el algoritmo es más eficiente; por ejemplo: insertar en una lista vacía, ordenar un vector que ya está ordenado, etc. Generalmente no nos interesa.
- **Caso medio.** Se puede computar T(n) como el tiempo medio de ejecución del programa sobre todas las posibles ejecuciones de entradas de tamaño n. El tiempo de ejecución medio es aveces una medida más realista del rendimiento en la práctica, pero es, normalmente, mucho más difícil de calcular que el tiempo de ejecución en el pero caso, ya que requiere definir una distribución de probabilidades de todo el conjunto de datos de entrada, el cuál típicamente es una tarea difícil.

Función de Complejidad

La función de complejidad de un algoritmo es el número de operaciones elementales que utiliza un algoritmo cualquiera para resolver un problema de tamaño n. Matemáticamente se define la Función de complejidad asi: Sea A un algoritmo, la función de complejidad del algoritmo A T(n) se define como el número máximo de operaciones elementales que utiliza el algoritmo para resolver un problema de tamaño n.

T(n) = Max {nx: nx es el número de operaciones que utiliza A para resolver una instancia x de tamaño n}

Tipos de Operaciones elementales:

- *Operación Lógica:* Son operaciones del tipo a > b, o por ejemplo los indicadores que se suelen utilizar en los condicionales que si se cumpla esta condición o esta otra haga esto. Ese o es una operación lógica.
- *Operacion Aritmetica:* Son operaciones del tipo a + b, o a / b, etc.
- *Asignación:* Es cuando asignamos a una variable un valor, ejemplo: int a = 20+30, el igual (=) en este caso es la operación de asignación.
- *Invocación a un Método:* Como su nombre lo dice es cuando llamamos, cuando invocamos a un método.

Cálculo del T(n)

Para hallar la función de complejidad (t(n)) de un algoritmo se puede evaluar el algoritmos desde tres puntos de vista:

- Peor Caso: Se puede hablar de T(n) como el tiempo de ejecución para el peor de los casos, en aquellos ejemplares del problema en el que el algoritmo es Menos Eficiente.
- Caso Medio: Se puede comportar el T(n) como el tiempo medio de ejecución del programa sobre todas las posibles ejecuciones de entrada de tamaño n. Es una medida más realista del rendimiento del algoritmo en la práctica, pero es mucho más difícil del cálculo, ya que requiere una distribucion de probabilidades de todo el conjunto de entrada lo cual es una tarea difícil.
- Mejor Caso: Se puede hablar de T(n) como el tiempo de ejecución para el mejor de los casos, en aquellos ejemplares del problema en el que el algoritmo es Más Eficiente.

Bibliografia

1- José Fager, W. Libardo Pantoja Yépez (2014), Estrcuturas de Datos, LATIn, Mexico. Pags 95-97