Tiempo de ejecución

Una vez que se diseñó el algoritmo y funciona correctamente, es necesario definir criterios para medir su rendimiento. Para ello, se pondrá el foco en:

- La simplicidad.
- El uso eficiente de los recursos.

El uso eficiente de los recursos se mide en función de dos parámetros: **el espacio**, memoria que utiliza, y **el tiempo**, lo que tarda en ejecutarse. Estos parámetros van a servir para comparar los algoritmos entre sí, lo que va a permitir determinar el más adecuado entre varios que solucionan un mismo problema.

El tiempo que consume un algoritmo para resolver un problema se llama **complejidad temporal,** y a la idea de la memoria que necesita el algoritmo se llama **complejidad espacial**.

La complejidad espacial, en general, tiene mucho menos interés. El tiempo es un recurso mucho más valioso que el espacio. Así que cuando se habla de **complejidad** a secas se hace referencia prácticamente siempre a complejidad temporal.

El tiempo que tarda en ejecutarse un algoritmo va a depender de diferentes factores, como, por ejemplo, los datos de entrada.

Hay dos estudios posibles sobre el tiempo:

1. Uno proporciona una medida **teórica (a priori),** que consiste en obtener una función que acote el tiempo de ejecución de un algoritmo para unos valores de entrada de datos.
2. Y otro que ofrece una medida **real (a posteriori),** que consiste en medir el tiempo de ejecución para unos valores en una computadora.

Ambas medidas son importantes, ya que la primera proporciona elementos independientes de la computadora donde serán implementados, y la segunda representa las medidas reales del comportamiento del algoritmo.

Este comportamiento puede cambiar notablemente para diferentes entradas. De hecho, para muchos programas, el tiempo de ejecución es en realidad una función de la entrada

específica, y no sólo del tamaño de ésta. Así, suelen estudiarse tres casos para un mismo algoritmo: caso peor, caso mejor y caso medio.

El **caso mejor** corresponde a la traza (secuencia de sentencias) del algoritmo que realiza menos instrucciones. Análogamente, el **caso peor** corresponde a la traza del algoritmo que realiza más instrucciones. Respecto al **caso medio**, corresponde a la traza del algoritmo que realiza un número de instrucciones igual a la esperanza matemática de la variable aleatoria definida por todas las posibles trazas del algoritmo para un tamaño de la entrada dado, con las probabilidades de que éstas ocurran para esa entrada.

Para entender bien este concepto, es necesario pensar qué es lo peor que puede pasar para un problema de tamaño n, y qué es lo mejor que puede pasar para un problema de tamaño n. Si existe un arreglo de n elementos y se desea encontrar el valor máximo, en el peor de los casos, se deberá recorrer todo el arreglo si el valor máximo se encuentra en la última posición. En el mejor de los casos, el valor máximo se debería encontrar en la primera posición.

Tiempo de ejecución

Una medida que suele ser útil conocer es el **tiempo de ejecución** de un programa en función de N, lo que se denomina T(n). Esta función se puede medir físicamente (ejecutando el programa, reloj en mano), o calcularse sobre el código contando instrucciones a ejecutar y multiplicando por el tiempo requerido por cada instrucción.

Se mide contando las operaciones elementales, es decir, aquellas instrucciones básicas que se ejecutan. Las operaciones aritméticas básicas, asignaciones a variables de tipo predefinido por el compilador, los saltos (llamadas a funciones y procedimientos, retorno desde ellos, etc.), las comparaciones lógicas y el acceso a estructuras básicas, como son los vectores y matrices, son operaciones elementales. Resumiendo, el tiempo de ejecución de un algoritmo va a ser una función que mide el número de operaciones elementales que realiza el algoritmo para un tamaño de entrada dado.

¿Cómo se calcula el tiempo de ejecución?

Si se tiene:

Asignaciones y expresiones simples

Se considera una operación elemental (OE) la asignación, las operaciones matemáticas, el acceso al vector, las comparaciones lógicas, entre otras.

Condicional

El tiempo de ejecución de la sentencia:

```
if  (C)
{ D}
else
                {H}
```

Se calcula como:

T= T(C) +max { T(D),T(H) }

Ciclos

El tiempo de ejecución de la sentencia:

```
while( C )
{ D  }
```

Se calcula como:

T= T(C) + número de iteraciones * {T(C)+T(D) }

Los otros ciclos repetitivos se expresan como el *while* previamente.

Funciones

El tiempo de ejecución de una llamada a una función es 1 más el tiempo de evaluación de los parámetros.

Ejemplo: la siguiente función busca un elemento en un vector:

```
int Buscar(int a[];int c)
{
int j;
{
```

```
  j=1;
  while (a[j]<c && j<n)
  {
    j=j+1 ;
  }
  if (a[j]==c)
    return j;
else
return 0;
}
}
```

Para determinar el tiempo de ejecución, se calcula primero el número de operaciones elementales (OE) que se realizan:

```
int Buscar(int a[];int c)
{
int j;
{
  j=0;
  while (a[j]<c && j<n)
  {
    j=j+1 ;
  }
  if (a[j]==c)
    return j;
else
return 0;
}
```

1 OE (operación elemental)

4 OE (dos comparaciones, un acceso al vector y el conector *and*)

2 OE (incremento y asignación)

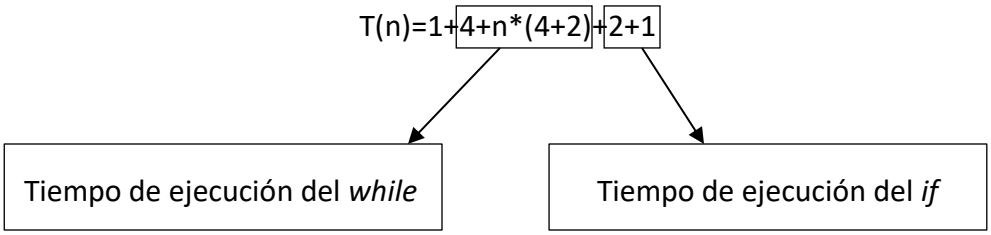
2 OE (acceso al arreglo y una condición)

1 OE (operación elemental)

1 OE (operación elemental)

}

El tiempo de ejecución, en el peor de los casos, es:



T(n)=6n+8