

ANÁLISIS Y DISEÑO DE ALGORITMOS

DIVIDE Y VENCERÁS

Práctica 5 de laboratorio

Entrega: Hasta el domingo 20 de marzo, 23:55h. A través de Moodle

La técnica de diseño de algoritmos *Divide y vencerás* consiste en resolver un problema reduciéndolo sucesivamente a subproblemas más sencillos hasta que la solución de estos resulte trivial. La complejidad temporal del algoritmo resultante depende, entre otros factores, del número de subproblemas que se generan.¹ El objetivo de esta práctica es comprobar empíricamente con un problema concreto cómo influye en la eficiencia del algoritmo resultante la forma en la que el problema inicial se divide en subproblemas.

El trabajo a realizar en esta práctica es el siguiente:

1. Sin hacer uso de la función `pow()` de C++ (ni de sus variantes), implementa tres funciones distintas que obtengan el valor de la potencia enésima de 2 (2^n donde $n \in \mathbb{N}$) de manera que cada una de ellas presente una complejidad temporal asintótica distinta a las demás. Incluye las tres funciones en un único fichero fuente con nombre `pow2.cc`. Los prototipos de las funciones han de ser los siguientes:

- `unsigned long pow2_1(unsigned)`
- `unsigned long pow2_2(unsigned)`
- `unsigned long pow2_3(unsigned)`

Asegúrate de que las funciones escritas obtienen el resultado correcto. Para esta comprobación sí puedes hacer uso de la función `pow()` o cualquiera de sus variantes.

2. En la línea inmediatamente anterior al encabezado de cada una de las funciones, escribe, entre comentarios, el coste temporal asintótico de la función; por ejemplo:

```
// coste: \Theta(2^n)
unsigned long pow2_1 (unsigned n)
{
    ....
}
```

Para que la función implementada se dé por válida, han de ser correctos tanto el coste temporal indicado (que debe ser diferente a las otras dos) como el resultado que obtiene.

¹El coste de obtener los subproblemas y el de combinar sus soluciones parciales también pueden afectar a la complejidad del algoritmo resultante.

3. De manera similar al trabajo realizado en las prácticas 1 y 2, completa el fichero fuente con la función `main()` para realizar un análisis empírico de la complejidad temporal de las tres funciones implementadas. Puedes hacerlo cronometrando tiempo de ejecución, contando pasos de programa o de cualquier otra forma que consideres apropiada. El programa resultante debe llamarse `pow2` y su ejecución debe mostrar por pantalla una tabla con los resultados de la medición para cada uno de los valores de n escogidos para el análisis y cada algoritmo.
4. A partir de la mencionada tabla y mediante la herramienta *Gnuplot* obtén una única gráfica que contenga las tres funciones de coste obtenidas. La gráfica debe llamarse `pow2.png` y se creará al ejecutar la orden `make` sin necesidad de añadir nada más. El archivo de órdenes de *gnuplot* debe llamarse `pow2.gnuplot`.

Normas para la entrega.

ATENCIÓN: Estas normas son de obligado cumplimiento para que esta práctica sea evaluada.

1. Se debe entregar los ficheros `pow2.cc`, `pow2.gnuplot` y `makefile`. **No hay que entregar nada más.** La gráfica que se pide deberá obtenerse escribiendo `make` en la terminal sin añadir nada más a la orden (`make` hará uso del fichero `makefile` entregado). **El tiempo total de proceso no debe ser superior a 15 segundos.**
2. Es imprescindible que no presente errores ni de compilación ni de interpretación (según corresponda), en los ordenadores del laboratorio asignado y en el sistema operativo *Linux*.² Se tratará de evitar también cualquier tipo de *warning*.
3. Todos los ficheros que se entregan deben contener el nombre del autor y su DNI (o NIE) en su primera línea (entre comentarios apropiados según el tipo de archivo).
4. Se comprimirán en un archivo `.tar.gz` cuyo nombre será el DNI del alumno, compuesto de 8 dígitos y una letra (o NIE, compuesto de una letra seguida de 7 dígitos y otra letra). Por ejemplo: `12345678A.tar.gz` o `X1234567A.tar.gz`. **Solo se admite este formato de compresión y solo es válida esta forma de nombrar el archivo.**
5. En el archivo comprimido **no debe existir subcarpetas**, es decir, al extraer sus archivos estos deben quedar guardados en la misma carpeta donde está el archivo que los contiene.
6. La práctica hay que subirla a *Moodle* respetando las fechas expuestas en el encabezado de este enunciado.

²Si trabajas con tu propio ordenador o con otro sistema operativo asegúrate de que este requisito se cumple (puedes comprobarlo haciendo uso del compilador *online* de <https://godbolt.org>).