comparación algoritmo recursivo con algoritmo iterativo

Programación de Sistemas

7 de abril de 2017

Sebastián Barbas 14025

Telmo García-Verdugo 14172

Cristina Escribano 14130

**Introducción**

En este proyecto vamos a estudiar las ventajas y desventajas en clave de tiempo de ejecución, memoria utilizada o numero de movimientos entre el mismo algoritmo implementado en forma iterativa o en forma recursiva.

En primer lugar vamos a definir las diferencias entre estas dos formas de trabajo; funciones iterativas son aquellas que se implementan por medio de algoritmos que se caracterizan por ejecutarse mediante ciclos. Y, por otro lado, funciones recursivas son aquellas que se invocan a si mismas en algún momento de su ejecución.

Para nuestro proyecto vamos a basarnos en el algoritmo de las Torres de Hanoi. Este rompecabezas tiene por objetivo cambiar todos los discos de la primera estaca a la tercera siguiendo unas reglas:

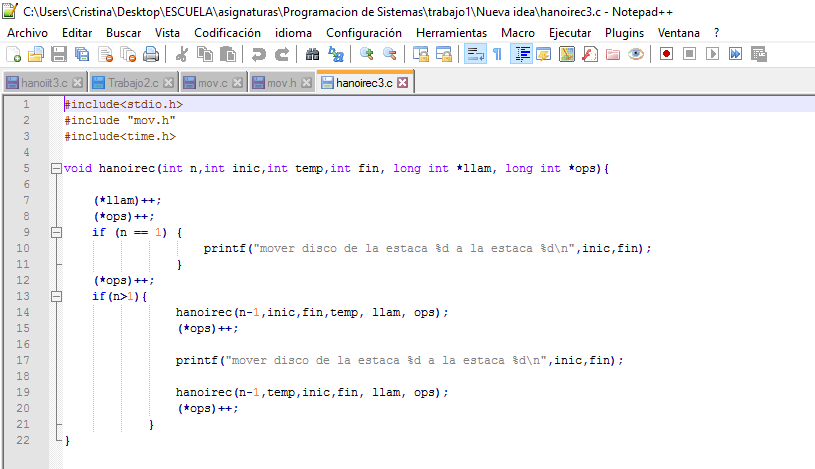
* Sólo se puede mover un disco cada vez.
* Un disco de mayor tamaño no puede descansar sobre uno más pequeño que él mismo.
* Sólo puedes desplazar el disco que se encuentre arriba en cada varilla



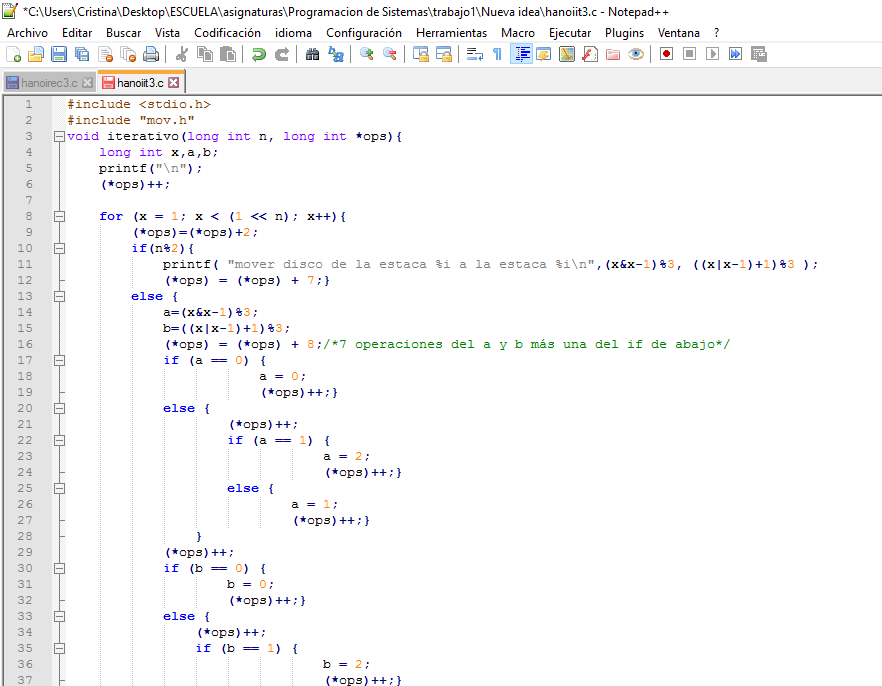
**Funciones empleadas**

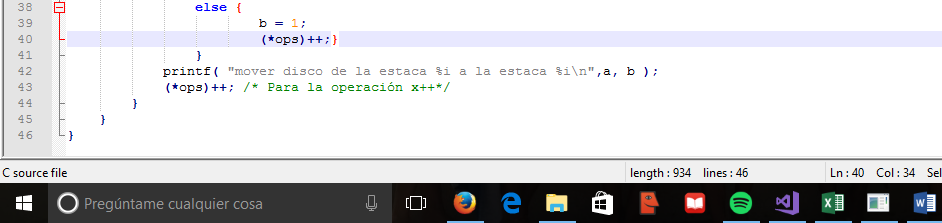
Para elaborar nuestro programa hemos implementado el algoritmo tanto en iterativo como en recursivo mediante funciones que, posteriormente, utilizaremos en el programa principal. Se han añadido instrucciones en el código de los ficheros, no están puestos en las fotos para una mayor legibilidad.

Función del algoritmo recursivo



Función del algoritmo iterativo

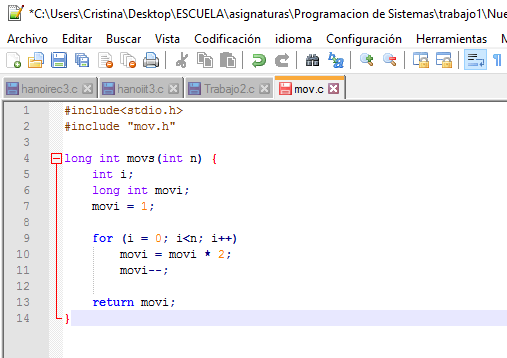




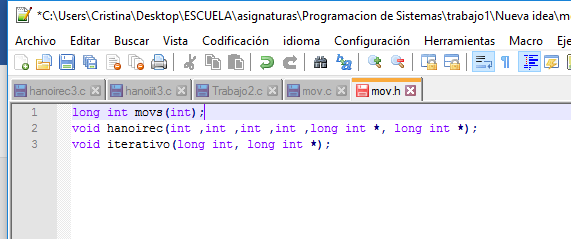
**Variables de medida de parámetros**

Para comparar los dos experimentos hay que definir variables de medida de las magnitudes a estudiar. En nuestro caso, para comprobar la eficiencia de las dos formas de trabajo hemos definido las siguientes:

* El número de movimientos *‘movs’*: para ello hemos creado un fichero llamada con el siguiente código.



Y su respectivo fichero *.h* donde se implementan los prototipos de las funciones.



* Contador de tiempo *‘tiempo’*: mediante una variable que almacena los milisegundos que tarda en ejercutarse la función, utilizamos un casting a *double* para obtener mayor precisión ya que las variables utilizadas eran de tipo *clock*.

*tiempo = (double)(t2 - t1) / CLOCKS\_PER\_SEC;*

* El número de llamadas a la función *‘llam’:* contabiliza las veces que entramos en la función recursiva. En el caso de utilizar la función iterativa siempre será la unidad.
* Espacio en memoria *‘tamn’*: Se hace un tratamiento diferente para recursivo y para iterativo. En el caso recursivo implementamos:

*tamn=llam\*(4\*(sizeof(int))+sizeof(int \*));*

donde multiplicamos variable *llam* (número de veces que entramos en la función recursiva) por el espacio que ocupa en memoria cada vez que se llama a dicha función.

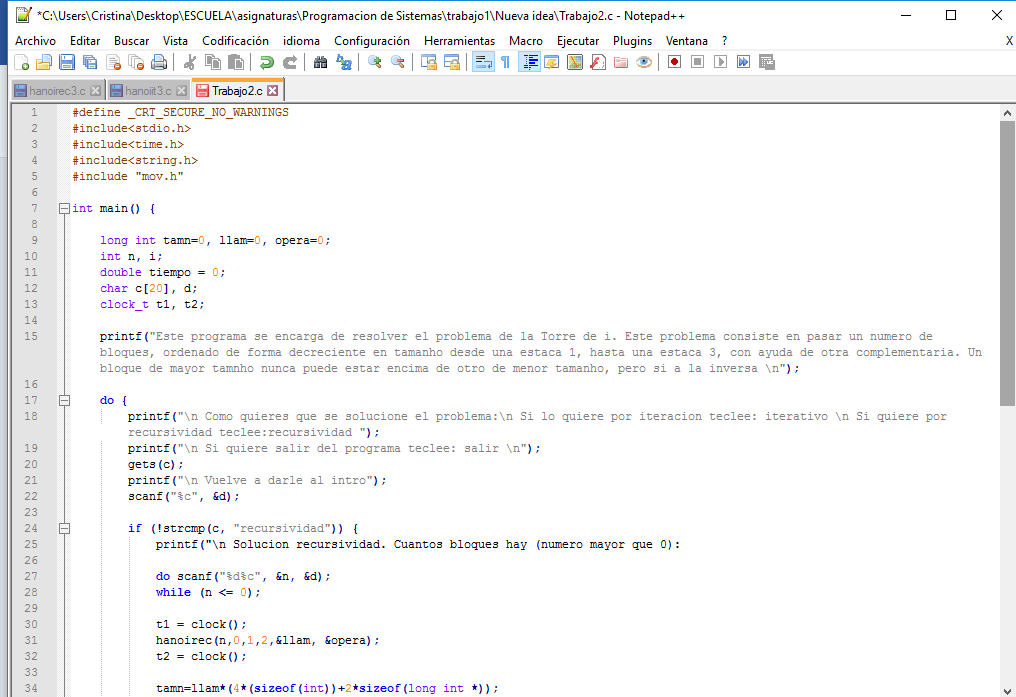
En el caso iterativo al no crearse nuevas variables solo se necesita:

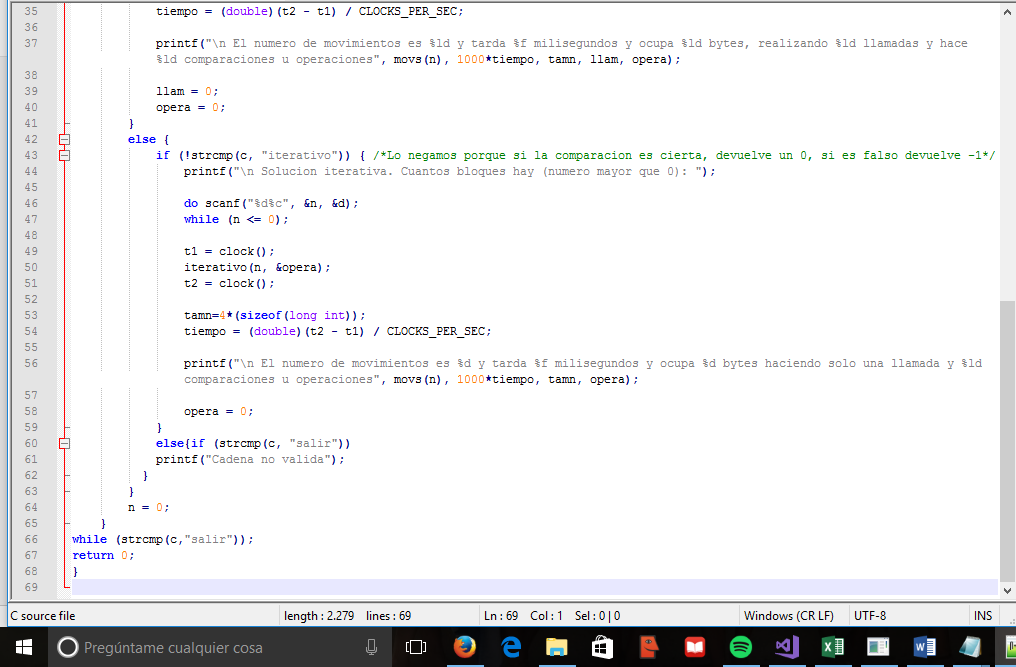
*tamn=2\*(sizeof(long int));*

* Número de comparaciones u operaciones que realiza cada función *‘opera’*: dentro de las funciones recursiva e iterativa hemos implementado un contador que incrementa el número exacto de operaciones o comparaciones que se realizan en cada expresión.

**Programa principal**

Tras crear las funciones explicadas creamos en programa principal.



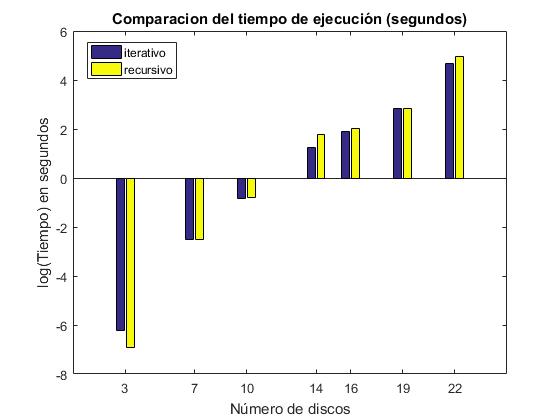
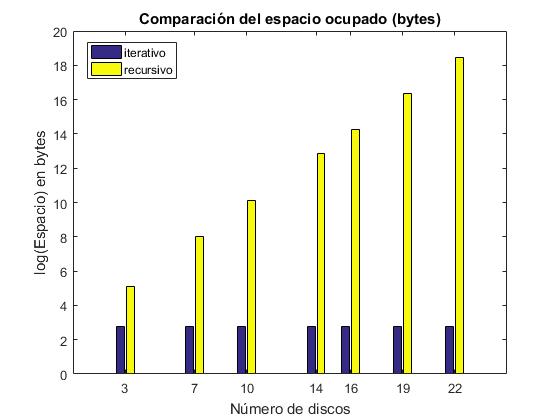


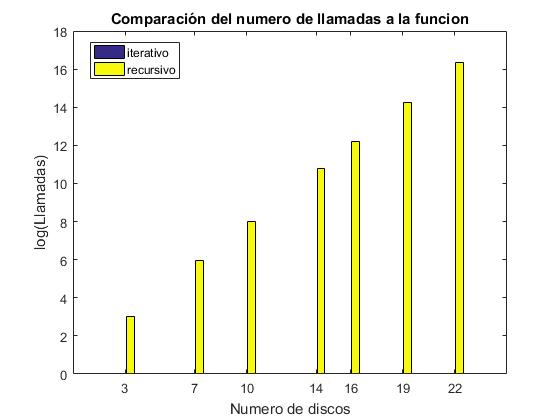
Resaltamos que el *strcmp* devuelve un 0 cuando la comparación es cierta y devuelve un -1 en caso contrario. En el caso de la línea 43 de código queremos que entre en el *if* si es cierto, por lo tanto, implementamos el *strcmp* negado.

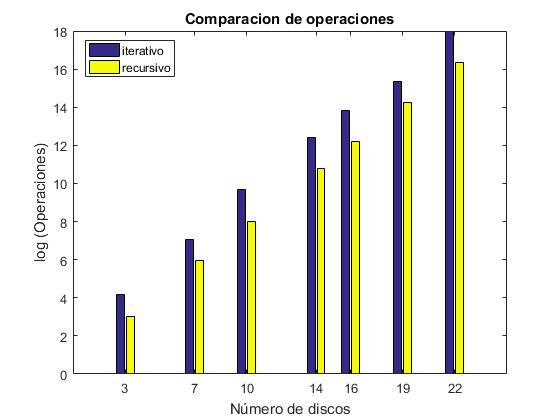
**Pruebas realizadas**

Tras compilar todos los ficheros indicados anteriormente realizamos las siguientes pruebas, y comparamos gráficamente.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nuemro de discos | iterativo | | | | | recursivo | | | | |
| Nº movimientos | Tiempo (min) | Espacio (bytes) | Llamadas | Operaciones | Nº movimientos | Tiempo (min) | Espacio (bytes) | Llamadas | Operaciones |
| 3 | 7 | 2 | 16 | 1 | 64 | 7 | 1 | 168 | 7 | 20 |
| 7 | 127 | 82 | 16 | 1 | 1 e3 | 127 | 80 | 3048 | 127 | 380 |
| 10 | 1 e3 | 428 | 16 | 1 | 15 e3 | 1 e3 | 444 | 24 e3 | 1 e3 | 3 e3 |
| 14 | 16 e3 | 3 e3 | 16 | 1 | 251 e3 | 16 e3 | 442 | 393 e3 | 16 e3 | 49 e3 |
| 16 | 65 e3 | 6 e3 | 16 | 1 | 1 e6 | 65 e3 | 6 e3 | 1 e6 | 65 e3 | 196 e3 |
| 19 | 524 e3 | 17 e3 | 16 | 1 | 4 e6 | 534 e3 | 17 e3 | 12 e6 | 52 e4 | 1 e6 |
| 22 | 4 e6 | 107 e3 | 16 | 1 | 64 e6 | 4 e6 | 14 e4 | 100 e6 | 4 e6 | 12 e6 |







A medida que se aumenta el número de discos el tiempo que tarda en ejecutarse el programa es mayor, como cabe esperar. Al principio nos llamó la atención que, aunque el programa recursivo realiza muchas llamadas y el iterativo solo una la diferencia de tiempo no es significativamente diferente. Tras observar el número de operaciones que se realizan vemos que el programa iterativo lleva a cabo muchas más comparaciones u operaciones que el recursivo, por lo tanto, entendemos que la diferencia de tiempo sea pequeña.

Vemos que en lo referido al espacio ocupado en memoria el iterativo es mucho más eficiente, no solo por el hecho de que solo ocupe 16 bytes frente a 168 bytes, 3048 bytes, etc; sino porque podemos saber a priori el espacio en memoria que vamos a utilizar, en cambio en el recursivo no podemos estimar previamente cuanto espacio necesitaremos y además es mucho mayor.

A pesar de que tras las comparaciones realizadas el programa iterativo parece bastante mejor que el recursivo, es cierto que es mucho menos intuitivo de programar.

**Reparto de los roles de trabajo**

En un primer momento trabajamos los tres juntos en las horas de prácticas de la asignatura y definimos las líneas generales del proyecto. Empezamos a implementar código básico del funcionamiento de las Torres de Hanoi y comenzamos a solventar los diferentes problemas que nos iban surgiendo.

Cuando estaba todo más o menos estructurado dividimos las partes, Telmo se encargó en profundidad de la función de iterativo y de investigar como contabilizar el tiempo, Sebastián implemento la función *main* e investigó como contabilizar la memoria utilizada y las operaciones realizadas en cada caso. Cristina implementó la función recursiva, llevo a cabo las diferentes pruebas y elaboro el documento.

Aunque es verdad que hemos dividido el trabajo, no ha habido necesidad de crear roles definidos, hemos trabajado eficientemente los tres sin que nadie haya tenido que organizar los pasos que debía seguir cada uno. Además, aunque cada uno elaborara una parte hemos ido compartiendo todos los archivos para que los tres supiéramos como se iba realizando el proyecto, aportando ideas y corrigiendo posibles errores.