

# COMPARACIÓN DE UN ALGORITMO RECURSIVO E ITERATIVO MEDIANTE EL CÁLCULO DE NÚMEROS PRIMOS EN UN INTERVALO DADO

Autores:

Adrián Rieker González 18300

Celia Ibáñez Martín 18172

Jorge Martínez de la Mata Gaitán 18211

ETSII UPM

Madrid, 30/04/2021

## INDICE

Introducción	pág. 2
Código fuente en C	pág. 3
Estudio de las condiciones	pág. 5
Pruebas experimentales	pág. 6
Resultados	pág. 7
Conclusiones	pág. 11
Reparto de roles	pág. 13

## Introducción

Este primer experimento tratará de observar, y de alguna forma medir, las diferencias en ejecución de un proceso según esté programado de forma iterativa o de forma recursiva. El algoritmo que hemos seleccionado tiene como objetivo calcular los números primos en un intervalo  $[a,b]$ . Para lograr esto se ha usado una función `esprimo()`, implementada de forma iterativa y recursiva. Esta función comprueba si un número introducido como parámetro es primo o no.

Implementaremos dentro del código variantes de `esprimo()` y funciones de experimentos que nos permitan extraer por pantalla datos relevantes de ejecución como pueden ser el tiempo que tarda en ejecutarse, el espacio consumido en memoria, o el número de comparaciones realizadas.

Para crear la función `EsPrimo` utilizaremos la teoría que sabemos para determinar si un número es primo o no, es decir, si sólo puede dividirse por sí mismo y por la unidad. Por tanto, en términos generales, ambos métodos versarán sobre comparar si el resto de la división de ese número por cualquier número anterior a éste hasta llegar a 2 da 0. Si se diera el caso, podríamos asegurar que ese número ya no es primo.

La función `EsPrimo` NO está optimizada, ya que, a pesar de que empezar a comparar por números bajos reduciría drásticamente todos los parámetros de ejecución del proceso, el método recursivo se complicaría de sobremanera, no siendo ese el objetivo del experimento.

```
| Que quieres hacer? |
A.- Info
B.- Experimentos
C.- Mostrar primos
D.- Introducir intervalo [a,b]
E.- Modo Iterativo/nRecursivo
F.- Salir

Has seleccionado: Experimentos
| Que quieres hacer? |
A.- Mostrar tiempo total transcurrido
B.- Mostrar numero de comparaciones
C.- Mostrar numero de llamadas
D.- Atras
```

## Código fuente en C

El código se divide en tres grupos:

- Ficheros para la interfaz: main.c main\_submenu.c y main\_submenu.h  
*En la interfaz se encarga de pedir al usuario que determine el intervalo, si se quiere calcular de forma recursiva o iterativa y los experimentos que se pueden realizar. Además, contiene información sobre su uso.*
- Ficheros para el cálculo de números primos: esprimo\_iterativo.c, esprimo\_recursivo.c esprimo\_iterativo.h, esprimo\_recursivo.h.
- Ficheros para la realización de los experimentos: experimentos.c y experimentos.h.
- Fichero de tipos: tipos.h

### Explicación general de cada fichero:

- **Esprimo\_iterativo.c:** Contiene el código en c de esprimo\_iterativo(), la función esprimo implementada de forma iterativa. Además, contiene el código en c de sus variantes para los experimentos: esprimo\_iterativo\_comparaciones (devuelve por dirección número de veces que se ha hecho una operación de modulo para saber si un número es divisor).
  - o **Esprimo\_iterativo.h:** contiene esencialmente los prototipos de las funciones contenidas en esprimo\_iterativo.c
- **Esprimo\_recursivo.c:** Contiene el código en c de esprimo\_recursivo(), la función esprimo implementada de forma recursiva. Además, contiene el código en c de sus variantes para los experimentos: esprimo\_recursivo\_comparaciones (devuelve por dirección número de veces que se ha hecho una operación de modulo para saber si un número es divisor) y esprimo\_recursivo\_llamadas (devuelve número de veces que la función recursiva se llama a si misma).
  - o **Esprimo\_recursivo.h:** contiene esencialmente los prototipos de las funciones contenidas en esprimo\_recursivo.c. Se ha usado compilación condicional para asegurar que no se incluya más de una vez este fichero.
- **Experimentos.c:** Contiene todas las funciones utilizadas en la recopilación de datos de ejecución de los experimentos. (experimento tiempo, experimento\_comparaciones y experimento\_llamadas).
  - o **Experimentos.h:** contiene esencialmente los prototipos de las funciones contenidas en experimentos.c. Se ha usado compilación condicional para asegurar que no se incluya más de una vez este fichero.
- **Main.c:** Contiene la interfaz del programa.
- **Main\_submenu.c:** Contiene las funciones de trabajo del menú. En concreto la función del submenú de experimentos, la función que pide por teclado el intervalo y una función del submenú info. Esta última sirve para mostrar información sobre cómo funciona el programa, pero no está totalmente terminada.
  - o **Main\_submenu.h:** contiene los prototipos de las funciones.
- **Mostrar\_primos.c:** Contiene el código en c de la función mostrar\_primos que muestra por pantalla todos los primos del intervalo [a,b] calculados ya sea iterativamente o recursivamente (opción elegida previamente por el usuario).
  - o **Mostrar\_primos.h:** Principalmente contiene los prototipos de las funciones

**Tipos.h:** Contiene la declaración de los tipos para las mediciones de tiempo, comparaciones, llamadas y para los números primos. Esto lo hemos hecho así para poder hacer pruebas con distintos tipos de forma eficiente. Se ha usado compilación condicional para asegurar que no se incluya más de una vez este fichero.

La medición del tiempo se realiza mediante la función `gettimeofday` contenida en la librería `<sys/time.h>`. Recibe por parámetro un `struct timeval` y `timezone`, que no se usa. La función almacena en segundos y microsegundos el valor del reloj del sistema. Para este programa se ha definido un `struct timeval` de inicio y otro de fin y de esa forma se ha calculado el tiempo de ejecución.

El fichero `tipos.h` se ha hecho para simplificar el cambio de los tipos con los que se definían las funciones y parámetros. Al principio, se planteó el problema declarando como `int` la mayoría de las variables, pero pronto se vio que era escaso y acabó utilizándose un `long unsigned int` para las llamadas, el tiempo y las comparaciones y `unsigned int` para los primos.

Las explicaciones más concretas se han hecho en comentarios en los ficheros.

## Estudio de las condiciones

Analizando el código podemos identificar diferentes condiciones que harán variar los parámetros de ejecución a evaluar:

El procesador en el que ejecutemos el código provocará que varíe en mayor o menor medida el tiempo de ejecución. Dentro del mismo procesador, un mismo programa con mismos datos puede tardar distintos tiempos de ejecución, por lo que habrá que hacer una media de tiempo.

Los datos que insertemos dentro del programa provocan que haya un mayor o menor número de comparaciones en el código, tanto por encontrar divisores del número, o por la longitud del número a analizar, lo que también aumentará tanto el espacio usado en memoria, como el tiempo de ejecución.

Los parámetros (metricas) elegidos para comparar los dos métodos son:

- **Tiempo total.** Tiempo que tarda desde que se llama a la función para calcular números primos hasta que termina de calcularlos en todo el intervalo.
- **Número de comparaciones.**
  - De forma iterativa la función calcula para cada número si es primo o no dividiéndolo por los inferiores a él hasta que encuentra uno divisible, en cuyo caso no es primo, o llega a uno. Cada división por uno de sus números inferiores es lo que hemos considerado una comparación.
  - De forma recursiva la función calcula si es divisible por su inmediatamente inferior y si lo es determina que no es primo, si no lo es se llama a la misma función para el número inmediatamente inferior al que estaba calculando y así sucesivamente. Esta división es lo que hemos considerado una comparación.
- **Número de llamadas.**
  - De forma iterativa consideramos el número de llamadas como el número de veces que se llama a la función que calcula números primos en el intervalo pedido de forma iterativa (`esprimo_iterativo_llamadas`) que coincide con la dimensión del intervalo a calcular.
  - De forma recursiva consideramos el número de llamadas como el número de veces que se llama a sí misma la función que calcula números primos en el intervalo pedido de forma recursiva (`esprimo_recursivo_llamadas`) más la dimensión del intervalo.

## Pruebas experimentales

El objetivo de estos experimentos es analizar las ventajas e inconvenientes de calcular números primos en un intervalo dado con un algoritmo iterativo y uno recursivo.

Las condiciones que hemos escogido se pueden dividir en dos grupos:

- Análisis de los experimentos en dos bloques, uno realizando los cálculos con un ordenador portátil genérico (bloque A) y otro con un ordenador de sobremesa con un procesador más potente (bloque B). Como el objetivo era comparar los dos algoritmos en las mismas condiciones los datos analizados son solo los del bloque A. No obstante, cabe destacar que para el bloque B se obtienen tiempos mucho más pequeños y para facilitar la comprensión se han utilizado los del bloque A.
- Análisis de los experimentos en tres intervalos distintos:
  - 1.- Intervalo de [1,100]
  - 2.- Intervalo de [1,12000]
  - 3.- Intervalo de [1, 1000]
  - 4.- Intervalo de [12000, 13000]
  - 5.- Intervalo de [15000, 16000]Estos tres últimos son interesantes para analizar solo las diferencias debidas a elegir números más altos ya que la longitud del intervalo es la misma.
  - 6.- También es interesante analizar comparaciones y número de llamadas para el intervalo [1,5] por su sencillez, pero para el análisis del tiempo no tiene sentido.

La razón de la elección de estos intervalos es que esperamos que para intervalos pequeños de números pequeños la diferencia entre los algoritmos no sea tan marcada mientras que para números muy grandes la forma iterativa será claramente más rápida y realizará menos comparaciones.

### **Acotación**

Para no complicar demasiado los cálculos, ya que el objetivo de este trabajo no es calcular números primos en el máximo intervalo posible, en ningún momento llegaremos a pedir un intervalo de un millón de números.

### **Decisión del número de repeticiones del experimento**

- **Tiempo total.** Analizando los primeros resultados obtenidos aleatorizando si se calcula de forma iterativa o recursiva (*figura 3*) concluimos que con ese tamaño de muestra se obtienen una varianza y desviación típica aceptables.
- **Número de comparaciones.** Repetimos este experimento solo una vez ya que el resultado siempre es el mismo.
- **Número de llamadas a la función.** Repetimos este experimento solo una vez ya que el resultado siempre es el mismo.

## Resultados:

### Bloque A

Intervalo [1,100]

Número de primos 25.

- **Tiempo total.** Este experimento es útil para analizar el número de comparaciones y de llamadas a la función con números más manejables, pero es un intervalo demasiado pequeño para estudiar el tiempo (*Figura 1*).

```
| Que quieres hacer? |
|-----|
A.- Mostrar tiempo total transcurrido
B.- Mostrar numero de comparaciones
C.- Mostrar numero de llamadas
D.- Atras
|-----|
a
Has seleccionado: Mostrar tiempo total transcurrido
Tiempo transcurrido = 0 microsegundos.
El numero de primos en el intervalo es 25:
```

Figura 1: Resultado del tiempo transcurrido para el intervalo [1,100]

- **Número de comparaciones.** Para el intervalo [1,100] el número de comparaciones que realiza la función iterativa es 3336 frente a las 3434 de la función recursiva (*Figura2*).
- **Número de llamadas a la función.** Para el intervalo [1,100] el número de llamadas que realiza a la función iterativa es 100 frente a las 3458 de la función recursiva (*Figura2*).

	Iterativo	Recursivo	Entre 1-100
numero de comparaciones	3336	3434	
	Iterativo	Recursivo	Entre 1-100
numero de llamadas	100	3458	

Figura 2: Numero de comparaciones y de llamadas para el intervalo [1,100]

Intervalo [1,12000]

Número de primos 1439.

- **Tiempo total.** Después de realizar varias veces el experimento se obtiene una media para el algoritmo iterativo de 164 milisegundos y 318 milisegundos para el recursivo. La diferencia de tiempo entre un método y otro es de 154 milisegundos (*Figura 3*).

	Iterativo	Recursivo	Entre 1-12000	Realizadas al azar si son iterativos o recursivos
1				
2 tiempo transcurrido	160,62	299,711		
3	198,543	288,263		
4	161,567	297,206		
5	154,662	393,945		
6	155,586	300,22		
7	153,587	329,118	milisegundos	
8 Media	164,0941667	318,077167	-153,983	
9 Varianza	0,295386749	1,57170255	milisegundos	
10 Desviación típica	0,017186819	0,0396447		

Figura 3: Resultado del tiempo transcurrido para el intervalo [1,12000]



- **Número de comparaciones.** Para el intervalo [1,12000] el número de comparaciones que realiza la función iterativa es 48235772 frente a las 48247770 de la función recursiva (Figura4).
- **Número de llamadas a la función.** Para el intervalo [1,12000] el número de llamadas que realiza a la función iterativa es 11999 frente a las 48249207 de la función recursiva (Figura4).

12		Iterativo	Recursivo	Entre 1-12000
13	numero de comparaciones	48235772	48247770	-11998
14		Iterativo	Recursivo	Entre 1-12000
15	numero de llamadas	11999	48249207	-48237208

Figura 4: Numero de comparaciones y de llamadas para el intervalo [1,12000]

Intervalo [1,1000]

Número de primos 169.

- **Tiempo total.** Después de realizar varias veces el experimento se obtiene una media para el algoritmo iterativo de 1,17 milisegundos y 2,31 milisegundos para el recursivo. La diferencia de tiempo entre un método y otro es de 1,14 milisegundos (Figura 5).

54		Iterativo	Recursivo	Entre 1-1000
55	tiempo transcurrido	1,029	2,029	
56		1,994	2,585	
57		0,999	1,994	
58		0,998	3,021	
59		0,996	1,957	
60		0,999	2,295	milisegundos
61	Media	1,169166667	2,3135	-1,144333333
62	Varianza	0,000163439	0,00017694	milisegundos
63	Desviación típica	0,000404276	0,00042065	

Figura 5: Resultado del tiempo transcurrido para el intervalo [1,1000]

- **Número de comparaciones.** Para el intervalo [1,1000] el número de comparaciones que realiza la función iterativa es 335054 frente a las 336052 de la función recursiva (Figura 6).
- **Número de llamadas a la función.** Para el intervalo [1,1000] el número de llamadas que realiza a la función iterativa es 1000 frente a las 3458 de la función recursiva (Figura 6).

	Iterativo	Recursivo	Entre 1-1000
numero de comparaciones	335054	336052	-998
	Iterativo	Recursivo	Entre 1-1000
numero de llamadas	1000	336219	-335219

Figura 6: Numero de comparaciones y de llamadas para el intervalo [1,1000]

Intervalo [12000,13000]

Número de primos 109.

- **Tiempo total.** Después de realizar varias veces el experimento se obtiene una media para el algoritmo iterativo de 29 milisegundos y 65 milisegundos para el recursivo. La diferencia de tiempo entre un método y otro es de 36 milisegundos (*Figura 7*).

	Iterativo	Recursivo	Entre 12000-13000	Realizadas al azar si son iterativos o recursivos
19				
20	tiempo transcurrido	28,889	56,853	
21		28,918	69,324	
22		26,928	94,747	
23		28,926	55,853	
24		30,946	54,857	
25		28,955	58,913 milisegundos	
26	Media	28,927	65,0911667	-36,16416667
27	Varianza	0,001614934	0,23870087 milisegundos	
28	Desviación típica	0,001270801	0,01544995	

Figura 7: Resultado del tiempo transcurrido para el intervalo [12000,13000]

- **Número de comparaciones.** Para el intervalo [12000,13000] el número de comparaciones que realiza la función iterativa es 8379105 frente a las 8380106 de la función recursiva (*Figura 8*).
- **Número de llamadas a la función.** Para el intervalo [12000,13000] el número de llamadas que realiza a la función iterativa es 1000 frente a las 48249207 de la función recursiva (*Figura 8*).

	Iterativo	Recursivo	Entre 12000-13000
30			
31	numero de comparaciones	8379105	8380106
32			-1001
33	numero de llamadas	1000	48249207
			-48248207

Figura 8: Numero de comparaciones y de llamadas para el intervalo [12000,13000]

Intervalo [15000,16000]

Número de primos 108.

- **Tiempo total.** Después de realizar varias veces el experimento se obtiene una media para el algoritmo iterativo de 37 milisegundos y 72 milisegundos para el recursivo. La diferencia de tiempo entre un método y otro es de 36 milisegundos (*Figura 9*).

	Iterativo	Recursivo	Entre 15000-16000	Realizadas al azar si son iterativos o recursivos
37				
38	tiempo transcurrido	33,94	68,86	
39		39,89	63,828	
40		33,944	83,776	
41		34,906	71,811	
42		37,884	64,818	
43		38,931	80,783 milisegundos	
44	Media	36,5825	72,3126667	-35,73016667
45	Varianza	0,00698081	0,06869735 milisegundos	
46	Desviación típica	0,002642122	0,00828839	

Figura 9: Resultado del tiempo transcurrido para el intervalo [15000,16000]

- **Número de comparaciones.** Para el intervalo [15000,16000] el número de comparaciones que realiza la función iterativa es 10390282 frente a las 10391283 de la función recursiva (*Figura 10*).
- **Número de llamadas a la función.** Para el intervalo [15000,16000] el número de llamadas que realiza a la función iterativa es 1000 frente a las 10391390 de la función recursiva (*Figura 10*).

65		Iterativo	Recursivo	Entre 15000-16000	
66	numero de comparaciones	10390282	10391283		-1001
67		Iterativo	Recursivo	Entre 15000-16000	
68	numero de llamadas	1000	10391390		-10390390

*Figura 10:* Numero de comparaciones y de llamadas para el intervalo [12000,13000]

Intervalo [1,5]

Número de primos 4.

Hemos considerado el 1 como número primo. Actualmente, no se suele considerar el 1 primo, pero para las comparaciones que se desean hacer en este trabajo carece de importancia si es primo o no.

En este caso el número de comparaciones de forma iterativa es 8 y el de llamadas es 4. Para el caso recursivo el número de comparaciones 11 es y el de llamadas es 13.

Este intervalo sirve más para comprender el funcionamiento del algoritmo que para analizar sus datos.

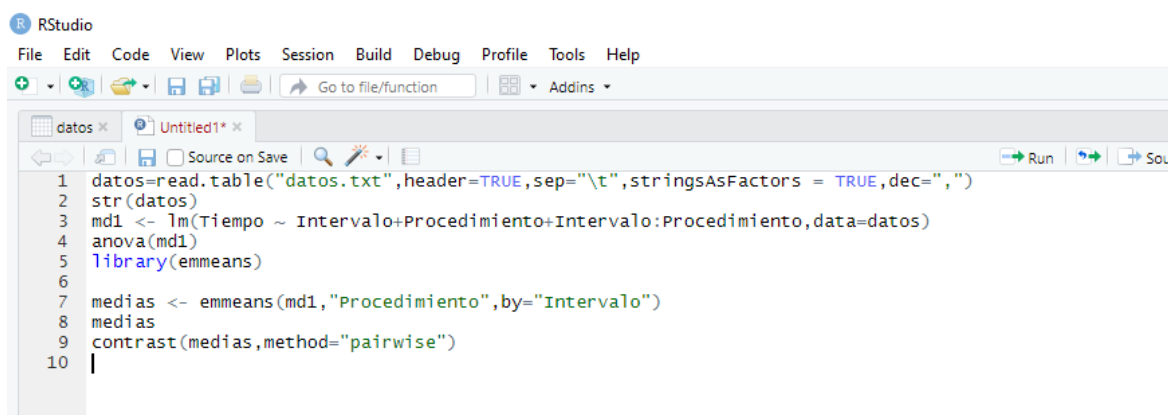
## Conclusiones

Al analizar los datos obtenido se ve en todos los intervalos que el número de comparaciones es siempre similar al número de llamadas para la función recursiva por cómo hemos definido las comparaciones para este algoritmo.

Por lo tanto, el número de llamadas de la función iterativa coincide siempre con el intervalo mientras que para la función recursiva es similar al número de comparaciones.

En el primer intervalo medido [1,100] observamos que no hay casi diferencia entre el número de comparaciones calculado de forma iterativa que recursiva. Sin embargo, para intervalos más altos como [1,1000], aunque siguen siendo números similares de comparaciones, la diferencia aumenta. En este caso, la función recursiva realiza 1000 comparaciones más que la iterativa.

Para comparar si las diferencias de tiempo entre iterativo y recursivo en los tres intervalos de misma longitud [1,1000], [12000,13000] y [15000,16000] utilizamos un análisis estadístico en R.



```
1 datos=read.table("datos.txt",header=TRUE,sep="\t",stringsAsFactors = TRUE,dec=",")
2 str(datos)
3 md1 <- lm(Tiempo ~ Intervalo+Procedimiento+Intervalo:Procedimiento,data=datos)
4 anova(md1)
5 library(emmeans)
6
7 medias <- emmeans(md1,"Procedimiento",by="Intervalo")
8 medias
9 contrast(medias,method="pairwise")
10 |
```

Figura 10: Programa utilizado en R para el análisis estadístico

Obtenemos las siguientes medias de tiempo y el error estándar (SE) según el procedimiento (Iterativo o Recursivo) y agrupados en los tres intervalos.

```
> medias
Intervalo = 1-1000:
Procedimiento emmean SE df lower.CL upper.CL
Ite           1.17 2.96 30  -4.88    7.22
Rec           2.31 2.96 30  -3.74    8.37

Intervalo = 12000-13000:
Procedimiento emmean SE df lower.CL upper.CL
Ite          28.93 2.96 30   22.87   34.98
Rec          65.09 2.96 30   59.04   71.15

Intervalo = 15000-16000:
Procedimiento emmean SE df lower.CL upper.CL
Ite          36.58 2.96 30   30.53   42.64
Rec          72.31 2.96 30   66.26   78.37
```

Figura 11: Tabla de medias y errores estándar del tiempo según el procedimiento y los intervalos escogidos

En la tabla ANOVA observamos que la interacción es significativa. Lo que significa que las diferencias entre los dos procedimientos dependen del intervalo.

```

Response: Tiempo
      Df Sum Sq Mean Sq F value    Pr(>F)
Intervalo 2 19529.8  9764.9 185.214 < 2.2e-16 ***
Procedimiento 1  5334.6   5334.6 101.184 3.989e-11 ***
Intervalo:Procedimiento 2  2422.8   1211.4  22.977 8.885e-07 ***
Residuals 30  1581.7    52.7
---

```

Figura 12: Tabla ANOVA para el análisis de la interacción entre intervalo y procedimiento

A continuación, se muestran las diferencias entre las medias de los procedimientos iterativo y recursivo. Para el primer intervalo, la diferencia es muy pequeña y el p.value es mayor que 0.05 por lo que no podemos decir que haya diferencia entre las medias. Sin embargo, para los intervalos [12000,13000] y [15000,16000] el p.value es menor que 0.05 por lo que la diferencia de medias es significativa. Podemos asegurar, con un nivel de significación del 5%, por ser la estima negativa, que el procedimiento recursivo necesita un tiempo medio mayor que el iterativo.

```

Confidence level used: 0.95
> contrast(medias,method="pairwise")
Intervalo = 1-1000:
contrast estimate SE df t.ratio p.value
Ite - Rec    -1.14 4.19 30 -0.273  0.7867

Intervalo = 12000-13000:
contrast estimate SE df t.ratio p.value
Ite - Rec   -36.16 4.19 30 -8.627 <.0001

Intervalo = 15000-16000:
contrast estimate SE df t.ratio p.value
Ite - Rec   -35.73 4.19 30 -8.523 <.0001

> |

```

Tabla 13: Análisis de la diferencia de medias de tiempo según el procedimiento y agrupados en los intervalos [1,1000], [12000,13000] y [15000,16000].

Por lo tanto, concluimos que el procedimiento recursivo siempre tarda más que el iterativo y que la diferencia por el intervalo no depende tanto de la longitud de este sino del tamaño sus números.

## Reparto de roles

Las tareas se han dividido de la siguiente forma:

Adrián Rieker:

- 1.- Preparar y probar el código básico de la interfaz y las funciones de cálculo de números primos.
- 2.- Probar los experimentos y recopilar los datos.

Celia Ibáñez:

- 1.- Preparar y probar el código básico de los experimentos.
- 2.- Probar los experimentos, recopilar los datos y analizar los resultados.
- 3.- Documentación.

Jorge Martínez de la Mata Gaitán

- 1.- Documentación.
- 2.- Probar los experimentos y recopilar los datos.