ESTRUCTURAS DE DATOS

RETO 1

Jose Manuel Martínez de la Insua

1) Usando la notación O , determinar la eficiencia de los siguientes segmentos de código:

a)

```
1   int n,j; int i=1; int x=0;                                  → O(1)
2   do{
3       j=1;                                        → O(1)
4       while (j <= n){
5           j=j*2;      → O(1)
6           x++;        → O(1)      → O(n x 1)      → O(n)      → O(n x n)      → O(n^2)
7       }
8       i++;                                        → O(1)
9   }while (i<=n);
```

b)

```
1   int n,j; int i=2; int x=0;                                  → O(1)
2   do{
3       j=1;                                    → O(1)
4       while (j <= i){
5           j=j*2;      → O(1)
6           x++;        → O(1)      → O(log2(i))    → O(log2(i))
7       }
8       i++;                                    → O(1)
9   }while (i<=n);
```

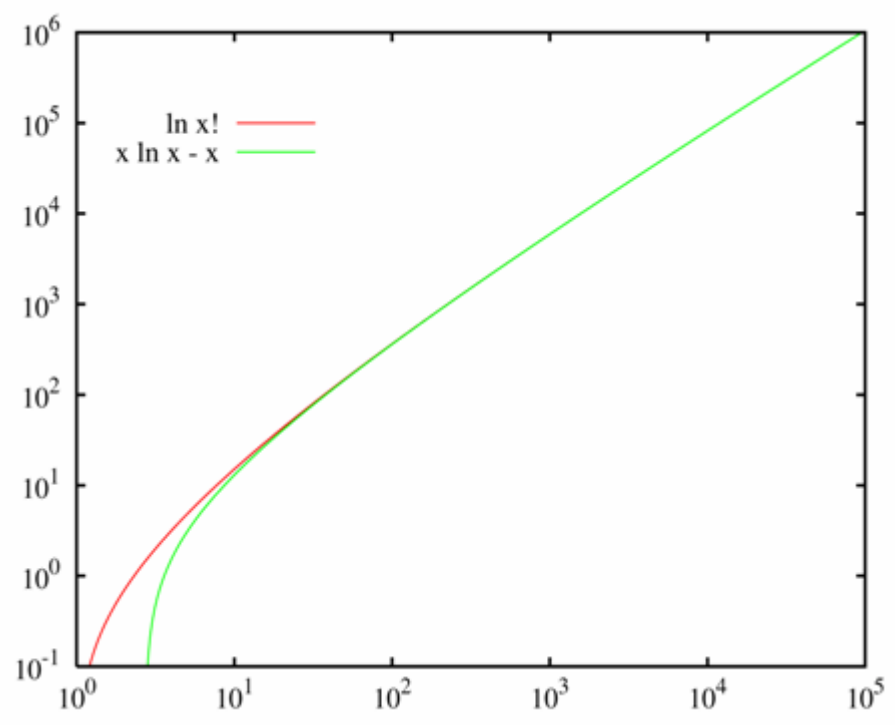
$$\sum_{i=2}^{n} \log_2(i) \;=\; \log_2(2) + \log_2(3) + \ldots + \log_2(n) \;=\; \log_2(\,2 \times 3 \times \ldots \times n\,) \;=\; \mathbf{O(\log_2(n!))}$$

En matemáticas la ***fórmula de Stirling*** es una aproximación para factoriales muy grandes y dice :

$$\ln(x!) - (x * \ln(x) - x) \;\rightarrow\; 0 \text{ al crecer } x$$

En otras palabras, podríamos aceptar $\log_2(n!) = n * \log(n) - n$, luego obtenemos **O(n*log(n))**

Anexo ejercicio 1.



Gráfica tomada de Wikipedia. Aproximación fórmula Stirling.
La diferencia tiende a 0 conforme aumenta x.

2) Para cada función f (n) y cada tiempo t de la tabla siguiente, determinar el mayor tamaño de un problema que puede ser resuelto en un tiempo t (suponiendo que el algoritmo para resolver el problema tarda f (n) micro-segundos, es decir, f (n) × 10 −6 sg.)

f(n)	***t***				
	1 sg.	1 h.	1 semana	1 año	1000 años
$\log_2 n$	10^{300000}	-	-	-	-
n	10^6	$3{,}6 \times 10^9$	$6{,}05 \times 10^{11}$	$3{,}15 \times 10^{15}$	$3{,}15 \times 10^{18}$
$n \log_2 n$	62746	$1{,}33 \times 10^8$	$1{,}77 \times 10^{10}$	8×10^{11}	$6{,}41 \times 10^{14}$
n^3	100	1532	8456	31593	315938
2^n	19	31	39	44	54
n!	9	12	14	16	18

Anexo ejercicio 2.

Herramienta implementada en c++ para comprobar los resultados obtenidos, solamente la he podido usar para las 3 últimas funciones ya que el tamaño del problema no es muy grande.

```
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <math.h>

using namespace std;

// Functions

long int exponente(int n){
	return  pow ( (double) 2,(double) n ) ;
}

long int cubo(int n){
	return  pow( (double) n,(double) 3) ;
}

long int factorial(int n){
	long int result = 1;
	for (int i = 2 ; i <= n ; i++)
		result *= i;
	return result;
}

// Función que admite como parámetro un puntero a función.

int tam_problema (long int ini,long int fin,long int (* fp)(int),long int limit) {
	bool encontrado = false;
	int result;
	for(int i = ini; i < fin && !encontrado; i++){
		result = i;
		if(fp(i) * (pow( (double) 10, (double) -6) ) > limit ) encontrado = true;
	}
	return result -1 ;
}


int main(int argc, char * argv[]){
	long int times[5] = {1,3600,604800,31536000,31536000000};
	for (int k = 0; k < 5; k++)
		cout << tam_problema(0,1000000,cubo,times[k]) << endl;
	for (int k = 0; k < 5; k++)
		cout << tam_problema(0,1000000,exponente,times[k]) << endl;
	for (int k = 0; k < 5; k++)
		cout << tam_problema(0,1000000,factorial,times[k]) << endl;
}
```