Ejercicio 3

```
unsigned ejercicio3 (unsigned n) {
unsigned  i=n, k=0;
while (i>0){
   unsigned j=i;
   do{
      j = j * 2;
      k = k + 1;
   } while (j<=n);
   i = i / 2;
}
return k;
}
```

Primero de todo definimos la complejidad del problema por el tamaño de "n", una vez visto eso, buscamos el mejor y peor caso, en este caso no hay mejor caso debido a que si o si hay que ir incrementando y decrementando "j" e "i" para que se cumplan las condiciones de los bucles.

En el peor caso (O), en primer lugar, estudiamos el bucle "grande" (línea 3-10), este controla la "i", esta, vale al en la primera iteración "n", y se va dividiendo entre 2 por cada iteración. En el bucle pequeño (línea 5-8) controla la "j", esta la duplica por cada iteración, al principio "j" tiene el valor de "i", y el bucle acaba cuando "j" supera a "n", el tiempo de ejecución de este bucle depende del valor que tenga el contador del anterior bucle.

iteración	i	J
1	n/2	1
2	n/4	2
3	n/8	3
.	.	.
n-1	0	n-1

La complejidad de "i" seria $\log(n)$, y la combinada de las dos seria estaría definida por:

$$\sum_{i}^{\log(n)} i \in O(log^2(n)$$

$\boldsymbol{O(log^2(n)}$ sería el orden.