Algoritmos
y
Estructuras de Datos

Cursada 2011

Ejercicio 3h

$2^{n+1} esO\,(2^n)\,?$

2^{n+1} crece a una velocidad menor o igual que 2^n ?

En otras palabras, 2^n es cota superior de 2^{n+1}?

Usando la definición de big Oh :
Podemos encontrar constantes positivas c y n_0 tal que 2^{n+1} $<= c2^n \forall n >= n_0$?

Sabemos que 2^{n+1} = $2(2^n)$ (por propiedad de las exponenciales)
Dado que encontramos el c y dado que aplica para $\forall n_0$ demostramos que 2^{n+1} es O(2^n)

Ejercicio 3i

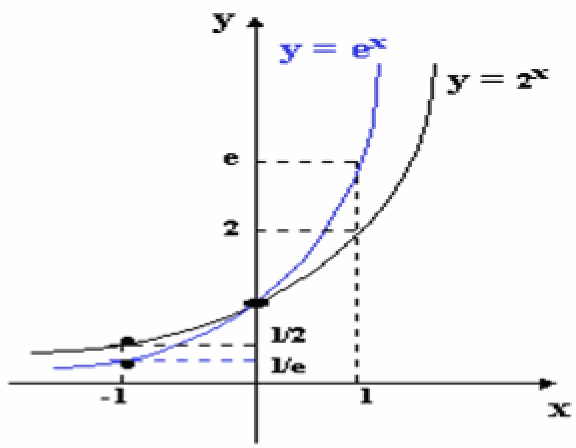
$2^{2n} es O(2^n)?$

Supongamos que es verdadero, por lo cual existen constantes positivas c y n_0 de forma tal que: $2^{2n} <= c2^n \forall n >= n_0$, luego $2^{2n} = 2^n 2^n <= c2^n \forall n >= n_0$

Esta igualdad es por la misma propiedad vista anteriormente.

Sin embargo, es falso que una constante sea superior a una función exponencial, por lo cual es absurdo y el planteo inicial es FALSO!

Ejercicio 3i continuación



Dibujar la función 2^n para valores positivos de n. Así visualmente se ve mejor que la misma es siempre creciente, por lo tanto NO se puede acotar por una cte, cualquiera sea esta!

Tener presente:

- Que se utilizo la definición de Big O
- Que se pueden utilizar todos los conceptos matemáticos conocidos (obviamente verdaderos!) SIN NECESIDAD de demostrarlos para encontrar las constantes. Remarcar esto. En este caso se utilizo que $a^n a^m = a^{n+m}$
- Se puede demostrar por el absurdo, suponer que la función es del orden y llegar a una contradicción evidente.

Ejercicio 6.1

```
public static void uno (int n)  {
   int  i, j, k ;
   int [] [] a, b, c;
   a = new int [n] [n];
   b = new int [n] [n];
   c = new int [n] [n];
   for ( i=1; i<=n-1; i++) {
    for ( j=i+1; j<=n; j++) {
      for ( k=1; k<=j; k++)  {
       c[i][j] = c[i][j]+ a[i][j]*b[i][j];
      }
    }
   }
  }
```

Igualdades que utilizaremos en el cálculo:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Ejercicio 6.1 continuación

d (por las instanciaciones que están afuera) $+\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{j} c =$

$$d+\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} cj =$$

$$d+c\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} j =$$

$$d+c\sum_{i=1}^{n-1} \left(\sum_{j=1}^{n} j - \sum_{j=1}^{i} j\right) =$$

$$d+c\sum_{i=1}^{n-1}\left(\frac{n(n+1)}{2}-\frac{i(i+1)}{2}\right)=$$

$$d+c\sum_{i=1}^{n-1}\left(\frac{n^2+n}{2}-\frac{i^2+i}{2}\right)=$$

$$d+\frac{c}{2}\sum_{i=1}^{n-1}n^2+\frac{c}{2}\sum_{i=1}^{n-1}n-\frac{c}{2}\sum_{i=1}^{n-1}i^2-\frac{c}{2}\sum_{i=1}^{n-1}i=$$

$$d+\frac{c}{2}\left(\sum_{i=1}^{n-1}n^2+\sum_{i=1}^{n-1}n-\sum_{i=1}^{n-1}i^2-\sum_{i=1}^{n-1}i\right)=$$

Ejercicio 6i (cont.)

$$d+\frac{c}{2}\left(n^2(n-1)+n(n-1)-\frac{(n-1)((n-1)+1)(2(n-1)+1)}{6}-\frac{(n-1)((n-1)+1)}{2}\right)=$$

$$d+\frac{c}{2}\left(n^3-n^2+n^2-n-\frac{(n-1)n(2n-1)}{6}-\frac{(n-1)n}{2}\right)=$$

$$d+\frac{c}{2}\left(n^3-n-\frac{(n^2-n)(2n-1)}{6}-\frac{n^2-n}{2}\right)=$$

$$d+\frac{c}{2}\left(n^3-n-\frac{2n^3-n^2-2n^2+n}{6}-\frac{n^2-n}{2}\right)=$$

$$d+\frac{c}{2}\left(n^3-n-\frac{2n^3-3n^2+n}{6}-\frac{n^2-n}{2}\right)=$$

$$d+\frac{c}{2}\left(n^3-n-\frac{n^3}{3}+\frac{n^2}{2}-\frac{n}{6}-\frac{n^2}{2}+\frac{n}{2}\right)=$$

$$d+\frac{c}{2}\left(\frac{2n^3}{3}-\frac{2n}{3}\right)=$$

Ejercicio 6i (cont.)

T(n) = $d + \frac{c}{2}\left(\frac{2n^3}{3} - \frac{2n}{3}\right) =$

Por aplicación de la regla de la suma la cual es el max(O(f(n)),O(g(n)))

Entonces como el término cúbico crece con mayor velocidad que el término lineal, nos quedamos con el mismo para el calculo del big - Oh

si existen constantes $c > 0$ y n_0 tales que:

$T_1(n) \leq c\, f(n)$ para todo $n \geq n_0$

$T_1(n) = \frac{2cn^3}{6} \leq c_1 n^3 \Rightarrow \frac{2c}{6} \leq c_1$ con $c_1 = c$ se cumple la condición.

$\therefore$ $O(n) = n^3$