Universidad Nacional de Ingeniería
Ciencias de la Computación
Análisis Asintótico

Yuri Nuñez Medrano *
ynunezm@gmail.com

Resumen

Se evaluará el tiempo de ejecucion (complejidad del algorítmo) en los diferentes casos mejor, peor y promedio. Se analizara el tipo de notación asíntotica " La Gran O".

1. Introducción

El analisis del algoritmo lo aproximaremos a una función en sus diferentes casos, que pudiera tener el algorímo mediante la aproximación a una función conocida, con el análisis asintótico. En una primera instancia con el limite superior con la "Gran O"

2. Analisis Asintótico

"Asintótico dicho de una curva, que se acerca de continuo a una recta o a otra curva sin llegar a nunca encontrarla " [DRA, 2001].

2.1. Notación Asintótica

Tiene como objeto simplificar el análisis de tiempo de ejecución mediante la eliminación de detalles que pueden verse afectados por especificaciones en la implementación y el Hw.
Ejm. similud de redondeo
$1000001 = 1000000$
$3n^2 = n^2$
Capturar la escencia: como el tiempo de ejecución aumenta, con el input en el limite.
Asitóticamente los algoritmos más eficientes son mejores para todos los inputs.

2.2. Caso Mejor, Peor y Promedio

Cada caso tiene su complejidad con su tiempo de ejecución de un algoritmo.

*Escuela de Ciencias de la Computación, 27-08-15

Ejm: Evaluar el tiempo de ejecución del algoritmo 1 y luego el caso mejor, peor y promedio.

Algorithm 1: INSERTION_SORT(A,n)

Input: Array A de n elementos de números
Output: Array A ordenado

```
1 for j = 2 to n do
2     key = A[j]
3     i = j − 1
4     while i > 0 and A[i] > key do
5         A[i + 1] = A[i]
6         i = i − 1
7     A[i + 1] = key
```

Al evaluar la complejidad linea por línea encontraremos los costos c_i donde $i = 1, 2, 3, ...$ y t_j es para evaluar el caso peor, mejor e intermedio.
1: $c_1 n$
2: $c_2(n-1)$
3: $c_3(n-1)$
4: $c_4 \sum_{j=2}^{n} t_j$
5: $c_5 \sum_{j=2}^{n} (t_j - 1)$
6: $c_6 \sum_{j=2}^{n} (t_j - 1)$
7: $c_7(n-1)$

El tiempo de ejecucion $T(n)$ es la suma de cada tiempo del algoritmo.

$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^{n} t_j + c_5 \sum_{j=2}^{n} (t_j - 1) + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7(n-1)$
Reemplazando las sumatorias
$T(n) = c_1 n + c_2 n - c_2 + c_3 n - c_3 + c_4(\frac{n^2}{2} + \frac{n}{2} - 1) + c_5(\frac{n^2}{2} - \frac{n}{2}) + c_6(\frac{n^2}{2} - \frac{n}{2}) + c_7 n - c_7$
$T(n) = -(c_2 + c_3 + c_4 + c_7) + n(c_1 + c_2 + c_3 + c_7 + \frac{c_4}{2} - \frac{c_5}{2} - \frac{c_6}{2}) + n^2(\frac{c_4}{2} + \frac{c_5}{2} + \frac{c_6}{2})$
La solución tiene la siguiente forma.
$-k_0 + nk_1 + n^2 k_2$
Evaluando los diferentes casos y reemplazando:

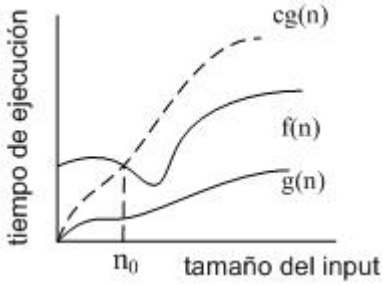

Figura 1: gran O

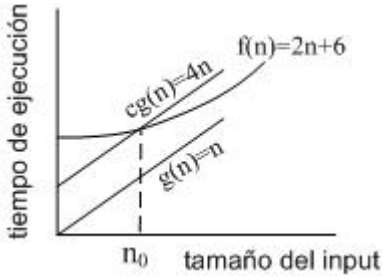

Figura 2: $2n + 6$

- Caso mejor: Cuando los elementos ya estan ordenados $t_j = 1$ tiempo de ejecución $= f(n)$, tiempo lineal.
- Caso peor: Cuando los elementos estan ordenado de manera inversa $t_j = j$ tiempo de ejecución $= f(n^2)$, tiempo cuadratico.
- Caso promedio: Cuando aproximadamente $t_j = j/2$ tiempo de ejecución $= f(n^2)$, tiempo cuadratico.

2.3. Notación "Gran O"

Asintóticamente el límite superior
$f(n) = O(g(n))$, Si existe c ctes y n_0, de tal manera que $f(n) \leq cg(n)$, $\forall n \geq n_0$.
$f(n)$ y $g(n)$ son funciones sobre numeros no negativos.
Es usada para el peor de los casos, en la figura 1.
Ejm: Para funciones $f(n)$ y $g(n)$ que tiene c constantes positivos y n_0, de tal manera que $f(n) \leq cg(n)$ para $n \geq n_0$, se define la pregunta. ¿$2n + 6$ es $O(n)$?
en la figura 2.
$f(n) = O(g(n))$
$f(n) = cg(n)$
si probamos con $c = 4$.
$4n = 2n + 6$
$n_0 = \frac{6}{2}$
Ejm: Se define la pregunta ¿n^2 es $O(n)$?
en la figura 3, observamos que n^2 no es $O(n)$ porque no hay c y $n \geq n_0$ de tal manera que no cumple $f(n) \leq cg(n)$ para $n \geq n_0$

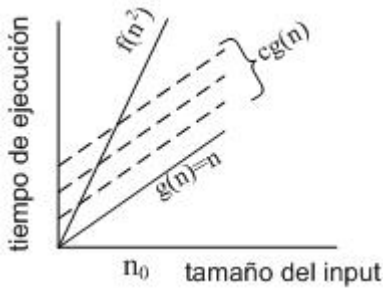

Figura 3: n^2

En la figura 3 se ilustra que no importa que tan grande sea elegido una "c" existe un n suficientemente grade $n^2 > cn$.
$f(n) \leq g(n)$
$f(n) = n^2$
$f(n) \nleq cg(n)$
$n^2 \nleq O(n)$
$n^2 > cn$

Regla Simple
Podemos definir una regla simple.
$50nlogn$ es $O(nlogn)$
$7n - 3$ es $O(n)$
$8n^2lgn + 5n^2 + n$ es $O(n^2lgn)$
$50nlogn$ es $O(n^5)$
Aunque $50nlgn$ es $O(n^5)$, se espera que la aproximación sea la menor posible.

Analisis Asintótico del tiempo de ejecución

- Uso de la notación-O para expresar el número de operaciones primitivas ejecutadas en función del tamaño del input.
- Comparación el tiempo de ejecución asintótico.
  - Un algorítmo ejecuta en un tiempo $O(n)$ es mejor que otro que ejecuta a un tiempo $O(n^2)$.
  - Jerarquia de funciones: $logn < n < n^2 < n^3 < 2^n$.

  Advertencia tener cuidado con los factores de costantes muy grandes.
- Un algoritmo ejecuta en un tiempo $1000000n$ es todavia $O(n)$ pero puede ser menos eficiente que una ejecución en tiempo $2n^2$, que es $O(n^2)$.

Ejr: Evaluar el tiempo de ejecución del algoritmo 2 y luego el caso mejor, peor y promedio. Luego diseñar un nuevo algoritmo PROM_PREFIJO2(X,n), la que sera mas optimo que el anterior, para que se evalue su tiempo de ejecución y su caso mejor, peor y promedio.

Algorithm 2: PROM_PREFIJO(X,n)

Input: Array X de n elementos de números

Output: Array A de n elementos, de talmanera que A[i] es el promedio de X[0]...X[i]

```
1 for i = 0 to n − 1 do
2 |   a = 0
3 |   for j = 0 to i do
4 |   |   a = a + X[j]
5 |   A[i] = a/(i + 1)
6 return A
```

Referencias

[DRA, 2001] DRA, R. A. E. (2001). *Diccionario de la Real Academia Española*. Espasa Calpe.

[H.Cormen et al., 2009] H.Cormen, T., Leiserson, C., and Riverson, R. L. (2009). *Algorithms*. The MIT Press.