Clase 5

Análisis de algoritmos
Introducción a matemáticas discretas
(Comp. 420)

José Joaquín Zubieta Rico

Abstract

Análisis de recursión y reducción.

Reducción Reducir un problema X a otro problema Y significa escribir un algoritmo para X que use un algoritmo Y como caja negra a subrutina.

Para probar que X es correcto, supongo que Y correcto.

Recursión Es un tipo de reducción muy poderoso que se describe:

- Si una instancia dada del problema se puede resolver directamente, resolverla directamente.
- Si no, reducirla a una o más instancias *más simples* del mismo problema (Hada de la recursión = Hipótesis inductiva).

No debe de haber una secuencia infinita de reducciones a instancias más simples Eventualmente las reducciones recursivas deben llegar a un *caso base* que se puede resolver con otro método.

Como ejemplo podemos definir el producto de dos número enteros de forma recursiva como

$$x \cdot y = \begin{cases} 0 & \text{if} x = 0 \\ \left\lfloor \frac{x}{2} \right\rfloor \cdot (y + y) & \text{if } 2 \mid x \\ \left\lfloor \frac{x}{2} \right\rfloor \cdot (y + y) + y & \text{if } 2 \nmid x \end{cases}$$

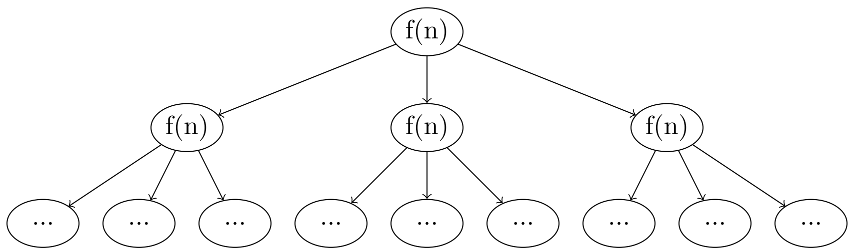

Figure 1: Árbol recursivo.

Y podemos escribir su algoritmo como

```
Multiply(x,y):
    prod = 0
    if x=0:
        prod = 0
    else
        x' = $\lfloor$ x/2$\rfloor$
        y' = y+y
        prod = Multiply(x', y')
        if x if odd
            prod = prod+y
    return prod
```

Divide and Conquer

1. Dividir la instancia del problema en varias instancias más pequeñas *INDEPENDIENTES*.
2. Delegar cada instancia más pequeña al hada de la recursión.
3. Combinar las soluciones para resolver la instancia general.

Árboles recursivos

Considerar un algoritmo de tipo *D&C* que toma $O(f(n))$ tiempo en su trabajo no recursivo y hace r llamadas recursivas, cada una a una instancia de tamaño $\frac{n}{c}$.

$$T(n) = rT\left(\frac{n}{c}\right) + f(n)$$