Técnicas de diseño de algoritmos
Introducción
Ejercicios (Bloque 1): con soluciones

Luis Javier Rodríguez Fuentes
Amparo Varona Fernández

Departamento de Electricidad y Electrónica
Facultad de Ciencia y Tecnología, UPV/EHU
luisjavier.rodriguez@ehu.es
amparo.varona@ehu.es

OpenCourseWare 2015
Campus Virtual UPV/EHU

Introducción – Ejercicios (Bloque 1)

(B1.1) A partir de la definición de $O()$, demostrar que $c_1 n + c_2 \in O(n)$.

(B1.2) A partir de las definiciones de $O()$ y $\Omega()$, demostrar que $f(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$.

(B1.3) Escribir en lenguaje Python una función que calcule la suma de los elementos de un vector. Calcular la complejidad temporal de dicha función y expresarla en notación asintótica.

(B1.4) Escribir en lenguaje Python una función que devuelva el número de valores que aparecen dos o más veces en un vector. Calcular la complejidad temporal de dicha función y expresarla en notación asintótica.

Introducción – Ejercicios (Bloque 1)

(B1.5) Calcular la complejidad temporal de la función que se muestra a continuación y expresarla en notación asintótica:

```
def neg(v):
    i=0
    while i<len(v) and v[i]>=0:
        i=i+1
    if i<len(v):
        return i
    else:
        return None
```

(B1.6) Calcular la complejidad temporal de la función que se muestra a continuación y expresarla en notación asintótica:

```
def heapify(h):
    for k in range(2,len(h)):
        j=k
        i=k//2
        item=h[k]
        while i>0 and h[i]<item:
            h[j]=h[i]
            j=i
            i=i//2
        h[j]=item
```


