Técnicas de diseño de algoritmos
Introducción
Examen (Autoevaluación): 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 – Examen

1. A partir de las definiciones de $O()$ y $\Theta()$, demostrar la propiedad de absorción:

$$f_1(n) \in O(g(n)) \wedge f_2(n) \in \Theta(g(n)) \Rightarrow f_1(n) + f_2(n) \in \Theta(g(n))$$

2. Calcular la complejidad temporal de la siguiente función, y expresarla en notación asintótica:

```
def sneg(v):
    for i in range(len(v)):
        s=0
        for j in range(i+1):
            s+=v[j]
        if s<0:
            return i
    return None
```

3. Escribir en lenguaje Python una función que, tomando como entradas un vector de enteros v de longitud n y un entero s, devuelva la lista de combinaciones de elementos de v que sumen exactamente s (cada combinación de índices se representará, a su vez, en forma de lista). La función devolverá una lista vacía si ninguna combinación de elementos de v suma exactamente s. Por ejemplo, dado $v = [3, 12, -1, 34, 7, -5]$, si la suma buscada es 4, la función retornará $[[0, 2, 4, 5]]$ (la única combinación de elementos que suma 4). Sin embargo, si la suma buscada es 8, la función devolverá una lista vacía, ya que no hay combinación de elementos que sume 8. Calcular la complejidad temporal de la función desarrollada, y expresarla en notación asintótica.