PECL3 - Divide y Vencerás

Miguel Ángel Guerrero y Jorge Guillamón

21 de junio de 2020

Resumen

Memoria de los ejercicios 3, 4 y 6. El código fuente se puede encontrar en este mismo directorio. Todos los ejercicios vienen con probador.

Ejercicios 3 y 6 - Búsqueda Binaria

Hemos planteado la resolución de estos ejercicios mediante una búsqueda binaria. La solución presentada hace uso de dos funciones:

```python
def funcion(x):
    inicio, fin, minimo = [0, 0], [100, 0], [30, -100]
    m1 = (minimo[1] - inicio[1]) / (minimo[0] - inicio[0])
    m2 = (minimo[1] - fin[1]) / (minimo[0] - fin[0])

    if inicio[0] <= x <= minimo[0]:
        return (x - inicio[0]) * m1 + inicio[1]
    elif minimo[0] < x <= fin[0]:
        return (x - minimo[0]) * m2 + minimo[1]
```

La primera de ellas es una función del tipo que se requiere en ambos problemas: una función definida en un intervalo [a,b], estrictamente decreciente en [a,c] con $a < c < b$ y estrictamente creciente en [c,b].
Como se puede apreciar en el código fuente, hemos escogido una función lineal de 2 tramos. Cambiando las variables de la primera línea se puede alterar el resultado al gusto.
Finalmente, la función principal del programa, tiene como argumentos: dos enteros, la x inicial y el tamaño del dominio; una función como la de antes, para calcular valores y finalmente el máximo error admisible.

```python
def busqueda(inicio, rango, funcion, error):
    if rango <= error:
        return inicio

    f1 = funcion(inicio + rango / 4)
    f2 = funcion(inicio + rango * 3 / 4)

    if (f1 <= f2):  # Mitad inferior
        return busqueda(inicio, rango / 2, funcion, error)
    else:  # Mitad superior
        return busqueda(inicio + rango / 2, rango / 2, funcion, error)
```

Figura 1:

Como se puede apreciar es una búsqueda binaria al uso. En este caso hemos tomado valores al 1/4 del dominio y 3/4 del dominio para saber a qué mitad dirigirnos en la siguiente iteración. Si $f2 \geq f1$ significa que el mínimo se encuentra en la mitad inferior y viceversa.

Ejercicio 4 - El robot embotellador

Para afrontar este problema se ha emparejado el primer tapón con su botella. A continuación, hemos usado esta información para dividir los tapones y botellas restantes en dos grupos según sean mayores o menores que la pareja. Tras esto nos quedamos con 2 grupos que se pueden resolver recursivamente.

```python
def robot(botellas: list, tapones: list):
    izdaT, izdaB, dchaT, dchaB = [], [], [], []
    tapon = tapones[0]
    botella = 0
    for i in range(len(tapones)):
        if botellas[i] == tapon:
            botella = botellas[i]
            print(tapon, "con", botella)
            break
    tapones.remove(tapon)
    botellas.remove(botella)

    for i in range(len(tapones)):
        if tapones[i] < botella:  # Dividimos tapones
            izdaT.append(tapones[i])
        else:
            dchaT.append(tapones[i])

        if botellas[i] < tapon:  # Dividimos botellas
            izdaB.append(botellas[i])
        else:
            dchaB.append(botellas[i])

    if len(izdaB) > 0:
        robot(izdaB, izdaT)
    if len(dchaB) > 0:
        robot(dchaB, dchaT)
```

Figura 2:

La salida de la función se hace directamente por pantalla. El programa sacará por pantalla un mensaje cada vez que el robot haya conseguido hacer una pareja.

Los probadores generan una lista aleatoria para cada ejecución del programa.

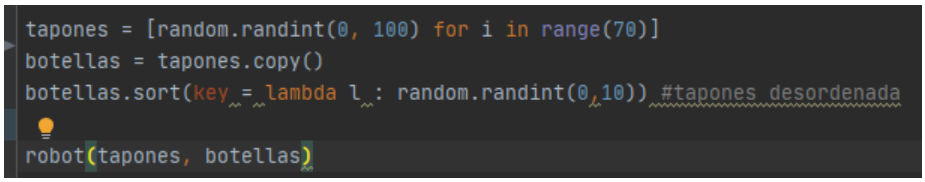

Figura 3:

Las listas se desordenan antes de ejecutarse.