Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Laboratorio de Algoritmos y Estructuras I
Trimestre Abril-Julio 2015

Prelaboratorio 9

El objetivo de este laboratorio es dar una introducción a la técnica de recursión.

1. Recursión

Para resolver un problema complicado podemos hacer un análisis descendente y dividirlo en varios subproblemas. Cuando en análisis descendente de un problema incluye la resolución de otra instancia más pequeña de ese mismo problema, diremos que el análisis propone una solución recursiva. Hay muchos problemas en los cuales es conveniente aplicar una solución recursiva. En particular, si un problema puede dividirse en una o más instancias del mismo problema, pero con una entrada más pequeña, diremos que la solución propuesta utiliza la técnica: divide y conquistarás.

2. Soluciones Recursivas

La recursión no sólo puede utilizarse para calcular el valor de funciones recursivas, sino también para resolver problemas algorítmicos. En este caso, la resolución de un problema depende de la resolución de instancias más pequeñas del mismo problema.

Por ejemplo, considere el problema de decidir si un elemento e se encuentra o no, en un arreglo de enteros, tal que los elementos están ordenados ascendentemente. Una posible solución sería recorrer secuencialmente el arreglo y comparar cada elemento con e. Otra solución más eficiente se puede formular de la siguiente manera:

- Se revisa un elemento aleatorio x del arreglo y se compara con e.
- Si $x = e$ entonces se indica que el elemento se encuentra en el arreglo.
- De lo contrario puede ocurrir dos casos:
  - Si $x < e$ podemos continuar la búsqueda solamente a los elementos que estén después de x en el arreglo.
  - Si $x > e$, de forma análoga se continúa la búsqueda sobre los elementos que están antes de x en el arreglo.

Note que el arreglo que se pasa como entrada a los subproblemas es siempre más pequeño que el original. Eventualmente, el arreglo será de tamaño cero. En tal caso, ya no tiene sentido dividir el problema, por lo que podemos reportar que el elemento no se encuentra en el arreglo. La figura 1

muestra la versión iterativa de la búsqueda binaria. La versión recursiva de la búsqueda binara se presenta en la figura 2. La ejecución de la funciones que implementan la búsqueda binaria se muestra en la figura 3.

```
def busqueda_binaria_iter(A, e, i, j):
        encontrado = False
        while(i <= j) and not encontrado:
            medio = (i + j)//2
            if e == A[medio] :
                encontrado =  True
            elif e < A[medio] :
                j = medio - 1
            else :
                i = medio + 1
        return encontrado
```

Figura 1: Código en Python de la búsqueda binaria iterativa

```
def busqueda_binaria_rec(A, e, i, j):
    if i <= j:
        medio = (i + j)//2
        if e == A[medio]:
            return True
        elif e < A[medio]:
            return busqueda_binaria_rec(A, e, i, medio-1)
        else :
            return busqueda_binaria_rec(A, e, medio+1, j)
    else:
        return False
```

Figura 2: Código en Python de la búsqueda binaria recursiva

```
A = [1,2,3,4,5,7,8,9,10]
e = 8
print( busqueda_binaria_iter(A, e, 0, len(A)-1) )
print( busqueda_binaria_rec(A, e, 0, len(A)-1) )
```

Figura 3: Ejemplo en Python de una llamada a búsqueda binaria iterativa y recursiva

3. Iteración vs Recursión

Escoger entre implementar una solución iterativa o recursiva dependerá de muchos factores, como la eficiencia, la facilidad de codificación, la legibilidad, entre otros. Generalmente, una solución iterativa será más eficiente que una recursiva (la razón de esto será mucho más clara adelante

en la carrera). Una solución iterativa no debe invocarse a sí misma, lo cual tendría un costo en tiempo y espacio. Sin embargo, diseñar soluciones iterativas no siempre es sencillo. Muchos problemas tienen soluciones que son naturalmente recursivas. Un buen ejemplo es el cálculo de la serie de Fibonacci. La solución recursiva es inmediata de la definición, sin embargo es sumamente ineficiente. La versión iterativa, aunque mucho más eficiente, es más difícil de diseñar.

4. Actividades a realizar

Debe resolver los siguientes problemas con programas en Python por medio de funciones recursivas y tiene que mostrar los resultados por la salida estándar.

1. **PreLab9Ejercicio1.py:** Dado por el usuario dos números enteros, determine el máximo común divisor de ambos.
2. **PreLab9Ejercicio2.py:** Dado una palabra por el usuario, determine si la misma es palíndromo.
3. **PreLab9Ejercicio3.py:** Dado por el usuario dos números enteros A y B con $B \geq 0$ compute A^B.

5. Condiciones de entrega

Tiene que entregar en la página del curso en el aula virtual, un archivo comprimido llamado `PreLab9-X.tar.gz`, donde X es su número de carné, que debe contener los archivos fuentes del código de su solución. La entrega debe hacerse antes de las 9:30 am del día martes de la semana 11.