Preparación PEP 2

Autores: Cristian Espinoza S., Shalini Ramchandani M, y otros.

RECURSIÓN E ITERACIÓN:

Ejercicio 1:

El caracol Turbo se encuentra participando en una de las carreras más grandes del mundo llamada *SnailRace*. La carrera consiste en una pista simple y recta de 30 cm, donde hay 9 caracoles más compitiendo. La velocidad de Turbo es de 3cm/seg, sin embargo, justo el gran día de la carrera corre un viento muy fuerte, por lo que por cada segundo que recorre 3 cm de la pista, al segundo siguiente retrocede 2 cm. Implemente un algoritmo recursivo y uno iterativo del problema descrito, con el objetivo que se pueda calcular en cuantos segundos recorre los 30 cm de la pista de la carrera.

Ejercicio 2:

Se tiene una lista que contiene todo lo nombres de un curso de Métodos de Programación, lo cual se pide formar todas las parejas posibles para cada alumno del curso, donde se debe realizar de forma recursiva y iterativa.

- Lista [“Juanito”, “Pedrito” , “Marcelo” , …… , “ ”]

DIVISIÓN Y CONQUISTA:

Ejercicio 3:

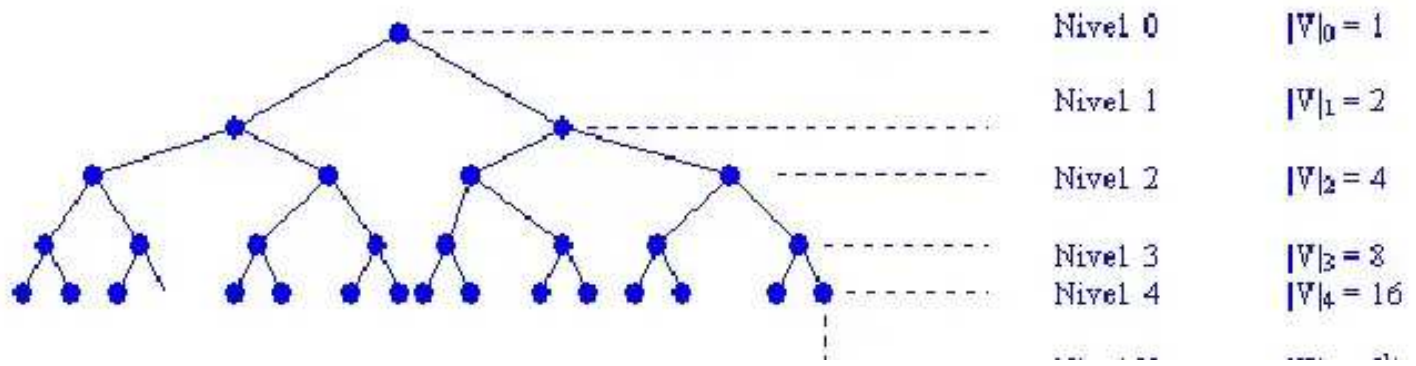
Se tiene una lista de números aleatorios que pueden ir desde 1 a 100 y se encuentran posicionados de manera desordenada. Además, la lista puede contener números repetidos. Se le solicita, aplicando la técnica de Divide y Conquista, implementar un algoritmo que permita obtener el número mínimo de la lista.

Consideraciones:

- Parámetros de entrada una lista *L* y el tamaño de ella *N*.
- Asuma *min(valor1, valor2)* como una función que calcula el mínimo entre dos números.

Ejercicio 4:

Tenemos un sistema de Grafos que representa una villa en la comuna de San Bernardo, cada uno de estos grafos identifica una casa y se conoce por cada una de ellas la cantidad de automóviles que contiene la familia. Se pide calcular la cantidad total de automóviles que hay por cada una de las casa respetando cada nivel que se tiene, es decir, que se debe calcular la cantidad de automóviles que existen en cada uno de los niveles y luego entregar el número total de automóviles que contiene la villa de San Bernardo.



- Grafo 1 {
  - CantidadAutomóviles = 2

  }

- Grafo 2 {
  - CantidadAutomóviles = 4

  }
- Estos son los datos que entrega cada uno de los nodos y la forma que cada casa entrega la información.

DIVISIÓN EN SUB-PROBLEMAS:

Ejercicio 5:

Se le solicita a implementar una calculadora utilizando la técnica de división en sub-problemas. Debe ingresar como parámetro los dos números y la operación a realizar, retornando el valor resultante. Se les exige mínimo de 5 sub-problemas y debe anotarlos todos junto a su algoritmo correspondiente.

Ejercicio 6:

Se pide realizar un aseo profundo dentro de un edificio, donde debemos considerar cada uno de los pisos y sus respectivos departamentos que contiene cada uno de ellos. Tener en consideración que cada uno de los departamentos contiene 3 dormitorios, 2 baños (Tener en consideración que un baño se encuentra en el primero piso y el segundo baño se encuentra en el segundo piso), 1 living comedor y por último, 1 cocina. Se solicita plantear cómo resolvería el problema utilizando división en sub-problemas.

Observación: Dejar planteado cada una de los supuestos utilizados para resolver el problema.

SOLUCIONES

RECURSIÓN E ITERACIÓN:

Ejercicio 1:

Algoritmo Recursivo:

```
funcion calcularSegundos(segundos, distancia):
        Si distancia = 0:  //caso base
                retornar segundos;
        Sino:
                Si segundos = impar:
                        retornar calcularSegundos(segundos+1, distancia+2);  //recursión
                Sino:
                        retornar calcularSegundos(segundos+1, distancia-3);  //recursión
```

Algoritmo Iterativo:

```
funcion calcularSegundos(distancia):
        segundos = 0;
        Mientras( distancia > 0 ):  //Iteración
                Si segundos = impar:
                        segundos = segundos + 1;
                        distancia = distancia + 2;
                Sino:
                        segundos = segundos + 1;
                        distancia = distancia - 3;
        retornar segundos;
```

Ejercicio 2:

Algoritmo Recursivo:

```
def funcion(lista, aux,aux1):


    if(aux1 == len(lista)):
        print("Cambio de sujeto")
        return funcion(lista,aux+1,0)

    if(aux == len(lista)):
        print("Largo completado")
        return "Terminado"

    else:
        if (aux == aux1):
            print("Mismo sujeto")
            return funcion(lista,aux,aux1+1)
        elif (aux1 <= len(lista)):
            print(lista[aux] + lista[aux1])
```

```
        return funcion(lista,aux,aux1+1)
```

Algoritmo Iterativo

```
def funcinoIte (lista):
    aux = 0
    aux1 = 0

    while (aux < len(lista)):
        while ( aux1 < len(lista) ):
            if (aux == aux1):
                print("Mismo sujeto")
                aux1 = aux1 + 1
            else:
                print(lista[aux] + lista[aux1])
                aux1 = aux1 + 1
        aux = aux + 1
        aux1 = 0

    print("Terminado")
    return
```

DIVISIÓN Y CONQUISTA:
Ejercicio 3:

```
funcion obtenerMinimo(lista L, tamaño N):
        Si N = 1:
                retornar L[0];
        Si N = 2:
                retornar min(L[0], L[1]);
        Sino:
                Si N es par:
                        lista L1 = L[0…(N/2) -1];
                        lista L2 = L[(N/2) … N-1];
                        minimo1 = obtenerMinimo(L1, N/2);
                        minimo2 = obtenerMinimo(L2, N/2);
                        retornar  min(minimo1,minimo2);
                Sino:
                        lista L1 = L[0 … ((N-1)/2)];
                        lista L2 = L[((N-1)/2)+1 … N-1];
```

```
minimo1 = obtenerMinimo(L1, ((N-1)/2)+1);
minimo2 = obtenerMinimo(L2, ((N-1)/2));
retornar min(minimo1,minimo2);
```

Ejercicio 4:

```
# Contendrá la cantidad de automoviles de cada una de las
# casas.
ListNivel = [ ..............  ]

def sumaNiveles(ListNivel):

    suma = 0
    contador = 0

    while (contador < len(ListNivel)):
        suma = suma + ListNivel[contador].CantidadAutomóviles
        contador = contador + 1

    print("El valor es: " + str(suma))
    # Esto se tiene que agregar a la lista que contendrá la suma
    # total de automóviles de cada nivel.
    return

# Finalmente se entrega la cantidad total de automóviles que se
# tiene en el nivel que se está analizando con el fin de agregar
# este valor a la lista de los totales de automóviles de cada
# uno de los niveles.


# Aca se tendrá la cantidad total de automóviles que contiene
# cada nivel, con el fin que se pueda entregar el valor total
# de automóviles, lo cual esta lista se va llenando a medida
# que se va calculando cada nivel.

ListTotal = [ ......................  ]

def valorTotal (ListTotal):

    suma = 0
    contador = 0
```

```
    while (contador < len(ListTotal)):
        suma = suma + ListTotal[contador]
        contador = contador + 1

    print("El valor es: " + str(suma))
    return

# Finalmente, se entrega la cantidad total de automóviles que existen
# en la villa
```

DIVISIÓN EN SUB-PROBLEMAS:

Ejercicio 5:

El problema principal corresponde a la calculadora, y se sub-divide este problema en las distintas operaciones que existen.

```
funcion calculadora(numero1, numero2, operacion):
        Si operacion = “suma”:
                retornar suma(numero1, numero2);
        Sino Si operacion = “resta”:
                retornar resta(numero1, numero2);
        Sino Si operacion = “multiplicación”:
                retornar multiplicacion(numero1, numero2);
        Sino SI operacion = “division”:
                retornar division(numero1, numero2);
        Sino Si operacion = “potencia”:
                retornar potencia(numero1, numero2);

funcion suma(numero1, numero2):
        retornar numero1 + numero2;

funcion resta(numero1, numero2):
        retornar numero1 -  numero2;

funcion multiplicacion(numero1, numero2):
        retornar numero1*numero2;

funcion division(numero1, numero2):
        Si numero2 = 0:
                retornar “Indefinido”;
        Sino:
                retornar numero1 / numero2;

funcion potencia(numero1, numero2): //numero1 = base, numero2 = exponente
```

```
Si numero2 = 0:
        retornar 1;
Sino:
        Si numero2 = 1:
                retonar numero1;
        Sino:
                retonar  numero1*potencia(numero1,numero2-1);
```

Ejercicio 6:

- Tener en cuenta que a simple vista podemos identificar que el problema se puede dividir en distintas funcionalidades para poder atacar el problema por partes, dentro de las cuales tenemos:
    - Separar el problema en cada uno de los niveles | Pisos:
        - Separar por cada uno de los departamentos:
            - 3 Dormitorios:
                - Dormitorio 1.
                - Dormitorio 2.
                - Dormitorio 3.
            - 2 Baños:
                - Baño primer piso.
                - Baño segundo piso.
            - LIving comedor.
            - Cocina.
    - Luego de tener todo estos sub-problemas separados, podemos empezar a resolverlos uno por uno y comenzar la limpieza en cada uno de los espacios que contienen los departamento.

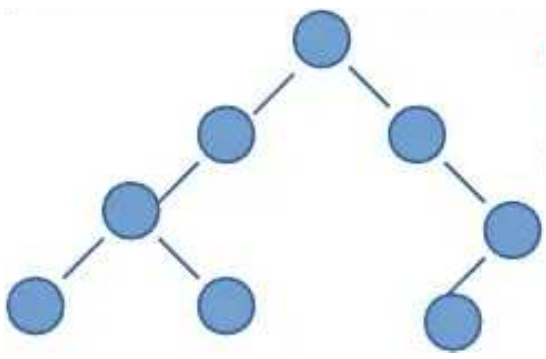
EJERCICIOS PROPUESTOS

1. Elemento mas repetidos en un arreglo

   Usando división en subproblemas resuelva como obtener el elemento más repetido en un arreglo X de n elementos X = [X0, . . . , Xn−1]. Indique cuales son son subproblemas planteados e implemente las funciones. Por ejemplo considere la función **mas_repetido**(X) que retorna el valor del elemento más repetido. Como retornaría ud además el índice del elemento más repetido?

2. Calcular la altura de un árbol binario.

   Usando división y conquista implente una función que retorne la altura de un árbol binario. Un árbol binario corresponde a grafo donde cada nodo puede tener dos nodos hijos(izquierdo y derecho). Un nodo puede no tener hijos, tener unos solo, o ambos. El arbol completo se representa por la raiz que es aquel del cual se originan todas las ramificaciones. La altura de un arbol es la cantidad maxima de relaciones “hijo” entre nodos. Como ejemplo el arbol siguiente tiene altura igual a 3.

   Para ello utilice las siguientes funciones:
   - izq(nodo) : entrega el nodo a la izquierda del nodo recibido, sino tiene retorna 0.
   - der(nodo) : entrega el nodo a la derecha del nodo recibido, sino tiene retorna 0.

3. Contar la cantidad de nodos de un árbol binario.

   Usando división y conquista implente una función que retorne la cantidad de nodos de un árbol binario. Para ello utilice las funciones definidas en la pregunta anterior. El ejemplo anterior tiene 8 nodos.

4. Contar la cantidad de hojas de un árbol binario.

   Usando división y conquista implente una función que retorne la cantidad de hojas de un árbol binario. Una hoja corresponde a un nodo con ningún hijo. Para ello utilice las funciones definidas en la pregunta anterior. En el ejemplo anterior el arbol tiene 3 hojas.

5. Contar la cantidad de nodos que tienen un solo hijo en un árbol binario.

   Usando división y conquista implente una función que retorne la cantidad de nodos con un solo hijo en un árbol binario. Para ello utilice las funciones definidas en la pregunta anterior. En el ejemplo anterior se tiene 3 nodos con un solo hijo.

6. Factoriales que terminan exactamente en n ceros

   Usando división y conquista implemente una función para buscar los enteros entre 0 y M cuyo factorial termine exactamente en n ceros. Utilice la función **nzeros**() para contar el número de ceros usando la operación x%y que indica cual es el resto de la división de x por y. Notar que siempre van a haber ceros después del 5! ya que hay factores que dan $10 = 5 * 2$. Divida los números a buscar en 2.

7. Separar palabras en un texto sin espacios

   Separar palabras en un texto sin espacios Implemente usando el método de división y conquista la función segmentar(T,D) que retorna las palabras separadas en el texto T suponiendo que dichas palabras se encuentran todas en el arreglo D. Por ejemplo el diccionario D puede ser:

   - D = [dolar,enrollado,enrolla] y
   - El texto S=’enrolladolar’

   El texto S debiese separarse en las palabras “enrrolla” y “dolar”. Notar que no se puede separar en ’enrollado’ y ’lar’, ya que “lar” no está en D. Lo anterior debiera retornar el arreglo [enrolla,dolar]. Utilice las funciones **buscar**(p,D) que retorna verdadero si la palabra p está en el arreglo D, y **substring**(s,i,j) que retorna la cadena de caracteres desde la posición i a j del texto s. Se retorna un arreglo vacío si no se encuentra una separación.

8. Pintores que pintan paredes contiguas

Se dispone de n paredes donde P = [P1, . . . , Pn−1] son los tiempos que demora pintarse una pared. Hay k pintores pero cada uno de ellos solo pinta paredes contiguas. Utilizando división y conquista implemente una función tiempo(P,n,k) que retorne el tiempo mínimo para pintar todas las paredes usando k pintores. Solo un pintor pinta una pared y cuando un pintor pinta sus paredes contiguas ya no trabaja mas.
Divida el problema en un pintor que pinta las paredes del final (de j a n-1) y el resto de los pintores (k-1) que pinta las primeras paredes (de 0 a j-1). Entonces al conocer ambos tiempos, el tiempo que demora en pintarse es el máximo de ambos números. Esto es para una elección de dicha división. Hay que buscar el j que entregue el tiempo mínimo.

9. Subsecuencia de suma máxima

   Se dispone de un arreglo de números A = [A0, . . . , An−1] que pueden ser positivos o negativos. Utilizando división y conquista implemente una función **sumaMax**(A) que entregue el valor máximo que puede alcanzar la suma de una sub-secuencia contigua de los números en A. Describa adicionalmente la division en subproblemas que realizó.

10. Subsecuencia creciente máxima

    Se dispone de un arreglo de números A = [A0, . . . , An−1] que pueden ser positivos o negativos. Utilizando división y conquista implemente una función **crecienteMax**(A) que entregue el largo máximo que puede alcanzar la sub-secuencia contigua de A que sea creciente. Describa adicionalmente la division en subproblemas que realizó.

11. Usted y un amigo se encuentra en el primer vagón de un tren y ha quedado de encontrarse con una amiga de la universidad en el vagón central, al avanzar se han percatado que no saben cuántos vagones posee el tren, pero este parece ser muy largo y solo recuerdan que en el folleto decía que el tren posee una cantidad impar de vagones. Además, los vagones al parecer no tienen ningún tipo de identificación para saber su número y todos los vagones lucen iguales por lo que cuesta distinguirlos. Al buscar ayuda notan que son los únicos en el tren. Saben que su amiga subirá al tren en la próxima

estación y ustedes quieren haber apartado un lugar para ella en ese vagón antes que suba gente en la estación. Es por esto que utiliza sus habilidades de programación para realizar un programa que le ayude a encontrar el vagón central del tren. Considere que ambos se comunican por celular. Para esto su amigo le ha ayudado con algunas funcionalidades

- Función avanzarYo(): Lo hace avanzar a usted al vagón siguiente.
- Función avanzarAmigo():Hace avanzar a su amigo al vagón siguiente.
- Función esVagonFinalYo(): Retorna verdadero si usted se encuentra en el último vagón.
- Función esVagonFinalAmigo(): Retorna verdadero si su amigo se encuentra en el último vagón.
- Función BuscarAmigo(): Hace que usted vaya al vagón donde se encuentra actualmente su amigo.
- Función LlamarAmigo(): Hace que su amigo vaya al vagón en el que se encuentra actualmente usted.

Ya que su amigo le ha proporcionado estas funciones que usted puede utilizar, lo reta a ver si puede crear un algoritmo recursivo y otro iterativo que utilicen estas instrucciones que permita ubicar a usted y su amigo en el vagón central del tren, partiendo ambos en el primer vagón de pasajeros. Además, le apuesta 30 puntos en la PEP 2 de métodos de programación, si le indica de forma correcta cuál tipo de recursión aplicó e implementa el algoritmo recursivo e iterativo donde ambos amigos recorren una sola vez el tren hasta el momento de encontrar el centro.

12. Se tiene una lista de números ordenados ascendentemente la cual contiene números enteros entre 1 y N incluido ambos extremos. La lista posee N-1 elementos y esto se debe a que a la lista le falta un número. Se le solicita

crear un algoritmo que utilice el método de resolución Divide y Conquista para determinar cuál es el número faltante en el listado, dada la lista L y el número máximo del listado N.

13. Divida el problema propuesto en la pregunta anterior en al menos 3 subproblemas. Considere además subdividir uno se esos subproblemas en al menos 2 partes. Entonces defina todos los subproblemas anteriores como funciones (no las implemente), explique que hacen, y utilicelas para escribir un pseudocódigo que resuelva el problema y el subproblema subdividido.

14. Usando recursion de pila, de cola, y iteración resuelva la suma de los numeros de Fibonacci. Es decir si los números de Fibonacci son F(n), entonces la suma de ellos es S(n)=S(n-1)+F(n).

15. Calcule la suma de los máximos de todos los subarreglos contiguos de A usando división y conquista. Por ejemplo si A=[1,3,1,2], los sub-arreglos son [[1], [1,3], [1,3,1], [1,3,1,2], [3], [3,1],[3,1,2],[1],[1,2]], y los máximos de dichos sub-arreglos son respectivamente [1,3,3,3,3,,3,3,1,2] entonces su suma es 22. Describa adicionalmente la division en subproblemas que realizó.

16. Usando división y conquista implemente una función que indique si un string que contiene solo parentesis “[]{}()” se encuentra balanceado o no. Por ejemplo “[[{}{}(()())]]” esta bien balanceado pero “[((()))” no lo esta.

17. Resuelva los siguientes ejercicios utilizando el método de resolución de problemas división y conquista:

    a) Escribir una función **pow**(a,b) en pseudocódigo que retorne la potencia de ab . Siendo a y b dos números enteros.

    b) Escribir una función **fib**(n) en pseudocódigo que retorne el enésimo número de Fibonacci.

    Realice la traza, acorde a los algoritmos implementados anteriormente para los casos:

    a) **pow**(2,10)
    b) **fib**(7)

18. Se desea implementar la multiplicación de matrices cuadradas de m x m por vectores de tamaño m. Se restringe m a ser una potencia de dos (m=2n) Por ejemplo, la matriz M por el vector v representados a continuación:

$$M = \begin{pmatrix} 1 & 2 & 1 & 2 \\ 3 & 4 & 3 & 4 \\ 1 & 2 & 1 & 2 \\ 3 & 4 & 3 & 4 \end{pmatrix}, \quad v = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

$$M \, x \, v = \begin{pmatrix} 1*1+2*2+1*3+2*4 \\ 3*1+4*2+3*3+4*4 \\ 1*1+2*2+1*3+2*4 \\ 3*1+4*2+3*3+4*4 \end{pmatrix} = \begin{pmatrix} 16 \\ 36 \\ 16 \\ 36 \end{pmatrix}$$

Utilice división y conquista para resolver el problema planteado.

a) Realizar un algoritmo acorde al método mencionado anteriormente. Este algoritmo debe poder resolver cualquier problema que posea la estructura indicada en el enunciado y no solamente el ejemplo.

b) Justificar e indicar el por qué el algoritmo desarrollado representa el método de resolución mencionado.

19. Usando recursión imprima todas las posibles permitaciones de n digitos. Por ejemplo n=3 genera la impresión de : 123, 132, 213, 231, 312, 321. Para ello considere que para “n” digitos (puede considerar letras si es mayor que 10) el primer dígito de cada sequencia puede variar en los n casos recibidos, y el resto de la secuencia puede variar con “n-1” digitos distintos. Establesca la recurencia necesaria y despues implemente la version iterativa.

20. Usando recursión imprima todas las combinaciones de “n” elementos en conjuntos de tamaño “k”. Por ejemplo si n=3 y k=2 se tiene : 12,13,23. Notar que por ejemplo 12 y 21 es la misma combinacion de elementos. Para ello considere que se puede separar el conjunto de combinaciones en aquellas que contienen al elemento 1 y aquellos que no lo contienen. El primero de ellos son todas las combinaciones de n-1 elementos en conjuntos de k-1 (agregando despues el elemento “1” faltante). El segundo son todas las combinaciones de n-1 elementos en conjuntos de k elementos. Note que lo anterior genera una recursion.
21. Se dispone de un arreglo A de numeros que comienza en forma creciente y termina en forma decreciente. Por ejemplo A=[1,2,3,4,5,3,2,-8]. Unsando división y conquista encuentre el elemento maximo, trate de basarse en el algoritmo de la búsqueda binaria.