El futuro digital es de todos

MinTIC

Ordenamiento y búsqueda recursiva

Misión TIC 2022

unab
VIGILADA MINEDUCACIÓN

Misión TIC 2022

Fundamentos de Programación

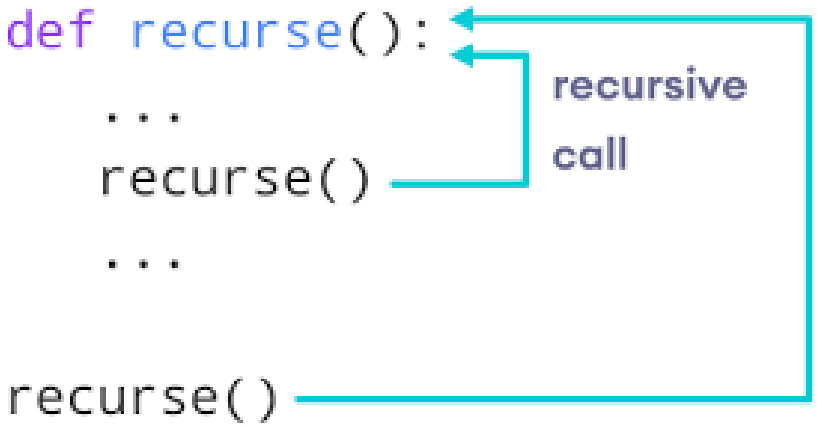
Ordenamiento y Búsqueda

- Búsqueda de información recursiva
- Ordenamiento de información recursivo

Ordenamiento y Búsqueda Recursiva

Recursividad

Se llama **recursividad** a un proceso mediante el cual una función **se** llama a sí misma de forma iterativa, hasta que se satisface alguna determinada condición.



Recursividad

Dado un número entero, calcular su factorial utilizando recursividad

Factorial NUM=NUM*(NUM-1)*(NUM-2)…….*1

Ejemplo : Factorial de 3 o 3! = 3*2*1=6

Casos especiales: Factorial (0) = 1 Factorial(1)=1

$$n! = n \times (n-1) \times (n-2) \times \ldots \times 1$$

$$1! = 1$$
$$2! = 2 \times 1 = 2$$
$$3! = 3 \times 2 \times 1 = 6$$
$$4! = 4 \times 3 \times 2 \times 1 = 24$$
$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

Recursividad

Análisis –> Ejercicio Factorial (Sin recursividad)

Entrada LEER

Proceso

Salida
IMPRIMIR

1 numero

3 Ciclos=> Recorrer los antecesores (FOR) inicia: numero finaliza: 1 incremento:-1

2 Condicionales=> Casos especiales . Si el numero es 0 o 1

4 Operaciones=> calculo factorial (multiplicación)

fact 5

Recursividad

Modularidad

Parámetros de entrada

numero

MODULO (FUNCION)
Factorial
- Condicional
- Ciclo
- Operaciones

2 3 4

Parámetros de salida

fact

FUNCIÓN retorna o regresa un solo valor

Recursividad


```
#Programa para el calculo del factorial sin recursividad
#Autor: Sergio Medina
#Fecha: 10/06/2021

#Funciones
def factorial(numero):
    if numero==0 or numero==1:
        return 1
    else:
        fact=1
        for i in range(numero,1,-1):
            fact=fact*i
        return fact
def valida_entero(etiqueta):
    while True:
        try:
            dato=int(input(etiqueta))
            break
        except ValueError:
            print(etiqueta," debe ser dato ENTERO")
    return dato
#Programa principal
numero=valida_entero("Ingrese Número: ")
fact=factorial(numero)
print("Factorial: ",fact)
```

Construcción –> Programa SIN recursividad

Recursividad


```
#Programa para el calculo del factorial con recursividad
#Autor: Sergio Medina
#Fecha: 10/06/2021

#Funciones
def factorial_recursivo(numero):
    if numero==0 or numero==1:
        return 1
    else:
        return numero*factorial_recursivo(numero-1)
def valida_entero(etiqueta):
    while True:
        try:
            dato=int(input(etiqueta))
            break
        except ValueError:
            print(etiqueta," debe ser dato ENTERO")
    return dato
#Programa principal
numero=valida_entero("Ingrese Número: ")
fact=factorial_recursivo(numero)
print("Factorial: ",fact)

#numero=3
# 3*factorial_recursivo(2)
# 3*2*factorial_recursivo(1)
# 3*2*1
```

Condición de salida de la recursividad

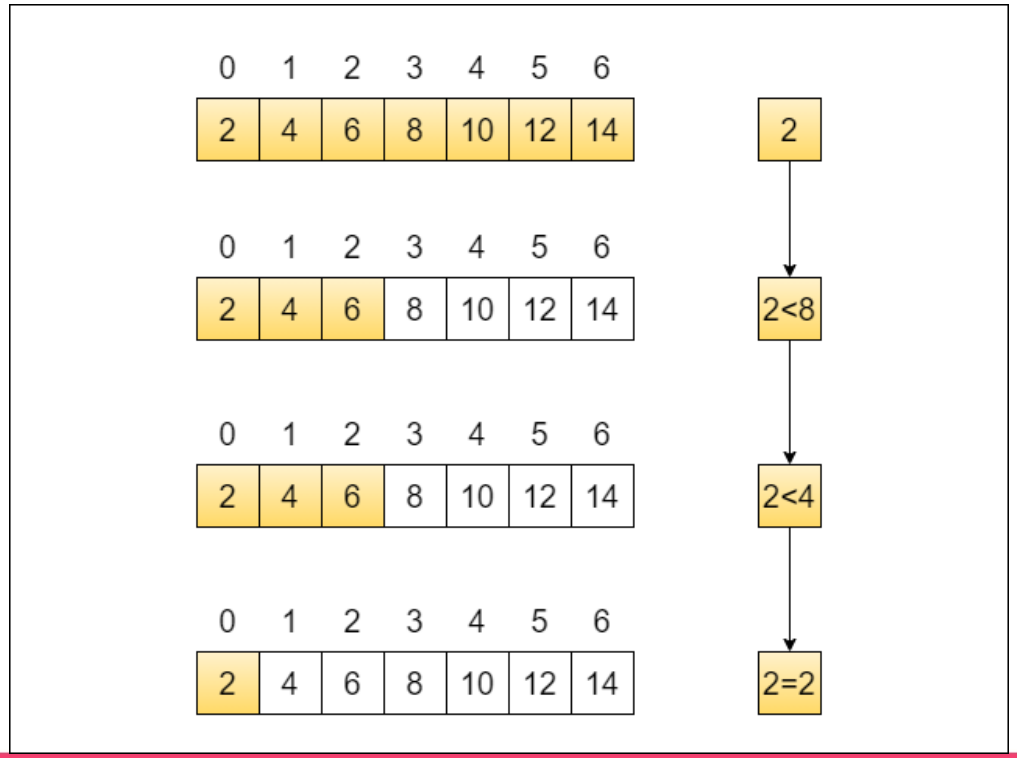
Llanada a la función dentro de la función (Recursividad)

Construcción –> Programa CON recursividad

Ordenamiento y Búsqueda de información

Búsqueda de información – Búsqueda binaria recursiva

La **búsqueda binaria es** un algoritmo eficiente para encontrar un elemento en una lista ordenada de elementos. Funciona al dividir repetidamente a la mitad la porción de la lista que podría contener al elemento, hasta reducir las ubicaciones posibles a solo una.



Búsqueda binaria recursiva

Construcción –> Programa

```
#Programa busqueda binaria recursivo
def busqueda_binaria_recursiva(vector,elemento,izq,der):
    if izq>der:
        return -1
    med=(izq+der)//2
    if vector[med]==elemento:
        return med
    elif vector[med]>elemento:
        return busqueda_binaria_recursiva(vector,elemento,izq,med-1)
    else:
        return busqueda_binaria_recursiva(vector,elemento,med+1,der)
def valida_entero(etiqueta):
    while True:
        try:
            dato=int(input(etiqueta))
            break
        except ValueError:
            print(etiqueta," debe ser dato ENTERO")
    return dato
```

Condición de salida de la recursividad

Llanada a la función dentro de la función (Recursividad)

Búsqueda binaria recursiva

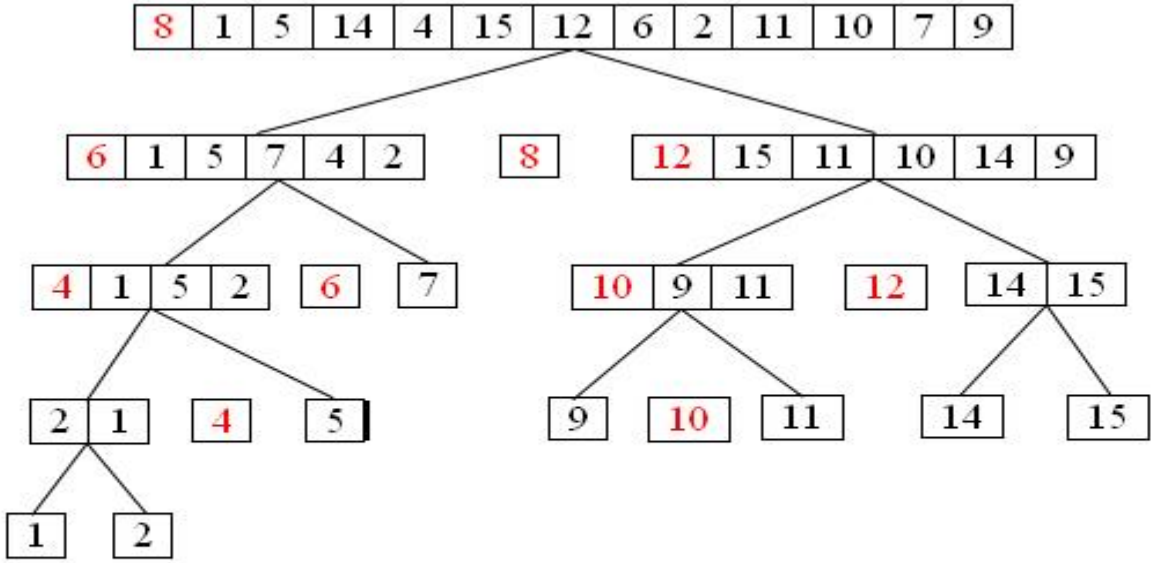
Construcción –> Programa

```
vector=(1,2,3,4,5,6,7,8,9,10)
elemento=valida_entero("Ingrese elemento: ")
izq=0
der=len(vector)-1
if busqueda_binaria_recursiva(vector,elemento,izq,der)==-1:
    print("Elemento no encontrado en lista")
else:
    print ("Posición elemento: ",busqueda_binaria_recursiva(vector,elemento,izq,der))
```

Ordenamiento recursivo

Ordenamiento Recursivo – Método QuickSort

QuickSort (en inglés, ordenamiento rápido). Es un algoritmo basado en la técnica de divide y vencerás, que permite, en promedio, ordenar n elementos en un tiempo proporcional a n log n.



Juntando los elementos, el arreglo quedaría ordenado

1	2	4	5	6	7	8	9	10	11	12	14	15

Ordenamiento recursivo

Ordenamiento Recursivo – Método QuickSort

Elementos importantes del método QuickSort:

- **PIVOTE**: Cualquier elemento de la lista (Se toma como referencia)=>Primero, último o mitad
- Proceso **PARTICION**: Recorre la lista y genera dos sublistas, una con los menores y otra con los mayores respecto al pivote.
- Proceso **RECURSIVO**: Aplica recursivamente la partición a las sublistas izquierda y derecha con la condición de salida (caso base) cuando la sublistas estén vacías o tengan un elemento

Ordenamiento recursivo

Método QuickSort – Primera parte: Partición

```
#Programa ordenamiento Quick Sort (Primera parte=>partición)
#Autor: Sergio Medina
#Fecha: 18/06/2021

#Funciones
def particion(lista):
    pivote=lista[0]
    menores=[]
    mayores=[]
    for i in range(1,len(lista),1):
        if lista[i]<pivote:
            menores.append(lista[i])
        else:
            mayores.append(lista[i])
    return menores,pivote,mayores

#Programa principal
lista=[8,1,5,14,4,15,12,6,2,11,10,7,9]
print("Listainicial: ",lista)

print("Partición: ",particion(lista))
```

Ordenamiento recursivo

Método QuickSort – Completo

```
#Programa ordenamiento Quick Sort
#Autor: Sergio Medina
#Fecha: 18/06/2021

#Funciones
def particion(lista):
    pivote=lista[0]
    menores=[]
    mayores=[]
    for i in range(1,len(lista),1):
        if lista[i]<pivote:
            menores.append(lista[i])
        else:
            mayores.append(lista[i])
    return menores,pivote,mayores

def quicksort(lista):
    if len(lista)<2:
        return lista
    else:
        menores,pivote,mayores=particion(lista)
        return quicksort(menores)+[pivote]+quicksort(mayores)
```

Ordenamiento recursivo

Método Quicksort - Completo

```
#Programa principal
lista=[8,1,5,14,4,15,12,6,2,11,10,7,9]
print("Listainicial: ",lista)
lista=quicksort(lista)
print("Lista ordenada: ",lista)
```

El futuro digital es de todos

MinTIC

Ordenamiento y búsqueda recursiva

Misión TIC 2022

unab
VIGILADA MINEDUCACIÓN

Misión TIC 2022