Laboratorio Nro. 1
Recursión

Luis Fernando Vargas Agudelo
Universidad Eafit
Medellín, Colombia
lvarga12@eafit.edu.co

Tomás Bedoya Henao
Universidad Eafit
Medellín, Colombia
Tbedoyah@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1

```
def rectangulo(N):
  if (N<=2):                                  # 1
    return N                                  # 1
  return rectangulo(N-1) + rectangulo(N-2)    # 1 + T(n-1) + T(n-2)
```

$$T(n) = \begin{cases} 1 & if\ n \leq 2 \\ 1 + T(n-1) + T(n-2)\ if\ n > 2 \end{cases}$$

Usando Wolfram para resolverlo, el resultado es:

T(n) = 2^n + c1 (donde C1 es un parámetro arbitrario)

3.2

a)

Para la realización de la gráfica se tomaron 20 valores diferentes para N con los cuales se analizó la complejidad del algoritmo, los valores para N aumentaron de forma lineal en base 10 mientras que los resultados como se esperaban se comportaron de forma exponencial.

La gráfica 1 está compuesta por los valores exponenciales, como se puede observar, el valor de tiempo que tarda para responder al problema con N = 1 es de 3 milisegundos mientras que para un N = 190 es de 1,81 E+49 días, lo que equivale a 4,97 E+44 siglos, estos datos se realizaron teniendo en cuenta que las operaciones básicas tardan un milisegundo.

La segunda gráfica tiene la ayuda de escala logarítmica en base 10, esto permite tener una mejor visualización de los valores mínimos y máximos.

b)

El algoritmo se demoraría 13031248,92 días, aproximadamente 365.542 siglos para resolver el problema con N = 50, es importante resaltar que estos valores pueden cambiar según la unidad de tiempo que demoren en realizar las operaciones básicas, ya que si para este mismo ejemplo, la unidad de respuesta es 1 nanosegundo la cantidad de tiempo que tardaría en responder para N = 1 sería de 312,74 horas

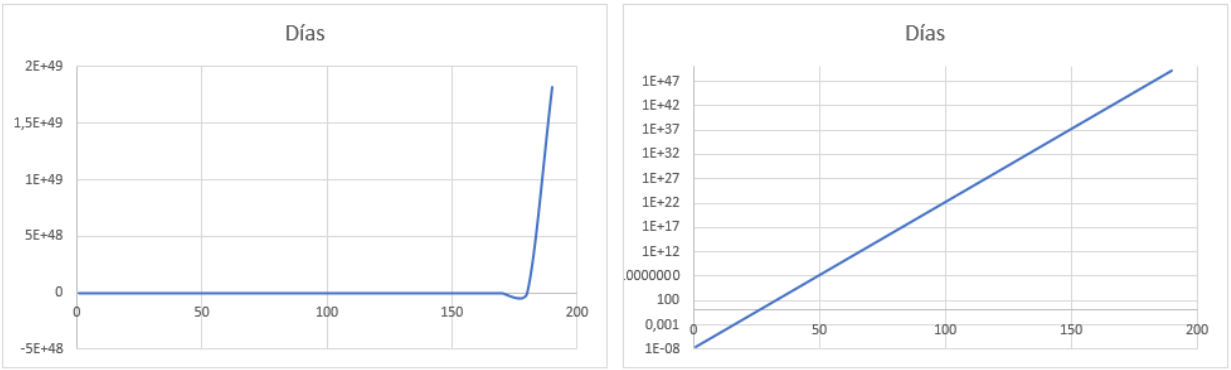

Ilustración 1 Tiempos de ejecución

3.3

Debido a su complejidad sería un algoritmo inviable para solucionar este problema ya que las dimensiones del contenedor expresadas en centímetros son demasiado grandes, por ende, la talla del problema sería tan grande que nunca se podría obtener una respuesta. Una aproximación para que pudiese funcionar sería tener las medidas del contenedor en metros, de igual forma evitando tener números muy grandes porque como se sabe el comportamiento es exponencial.

3.4 El ejercicio GroupSum5 resuelto es el siguiente:

```
def groupSum5(index, nums, target):
  if(index >= len(nums)):
    return target==0
  if (nums[index]%5==0):
   return groupSum5(index+1 ,nums, target -nums[index] )
  elif (nums[index-1]%5==0) and (nums[index]==1):
    return groupSum5(index+1 ,nums, target)
  else:
    return groupSum5(index+1 ,nums, target - nums[index]) or groupSum5(index+1
,nums, target)
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Se ingresan los siguientes parámetros a la función:un indexador o variable contadora (Que aumenta por cada llamado recursivo), la lista o arreglo de números enteros y el target u objetivo.
El algoritmo primero se asegura de que no se haya recorrido toda la lista (Y no se recorre infinitamente tras los llamados recursivos) comparando la variable index contra la longitud de la lista; si es así (Ya se recorrió la lista o se intentó con todos los valores una combinación posible), se contrasta si el objetivo es igual a cero, indicando que hubo al menos una combinación que cumplía el requisito (Un subgrupo que enteros que sume Target). Posteriormente, testea si la posición de la lista en la que se encuentra es múltiplo de 5 y lo toma, aumentando entonces el index en 1 unidad y restando ese valor (La posición de la lista en la que se encuentra evaluando) al target, para poder seguir restando, es decir, ya se tomó uno de los números y por tanto lo que falta para llegar al objetivo (Target) es target – ese número. Si no es múltiplo de 5, entonces testea que el anterior número evaluado sea múltiplo de 5 y el actual (Donde se encuentra evaluando) sea igual a 1, para proceder a no tomarlo, es decir, aumentar el index en una unidad (Para que continúe evaluando hasta que index sea mayor o igual a la longitud de la lista) y el target se deja tal cual (Pues no se tomó, y por tanto lo que falta para llegar al target es el mismo target). Si no se cumplen ninguna de las dos condiciones especiales (Que puede ser visto como una sola), el algoritmo ensaya poniendo y no poninendo ese valor, es decir, restándolo al target y no restándolo al target, pero siempre avanzando (Index + 1). El algoritmo entonces podemos decir que toma un elemento de la lista y lo combina o no lo combina con los demás, es decir, lo resta del target o no lo resta, luego procede a combinar o no combinar el número siguiente y así sucesivamente, siempre y cuando no sea un múltiplo de 5, porque lo toma inmediatamente y siempre y cuando no haya un 1 después de un múltiplo de 5, porque no lo toma (Aunque fuera necesario para cumplir el objetivo), hasta que se llega al punto en el que I es mayor o igual a la longitud y se prueba si se cumplió o no el objetivo (Target = 0), al final, después de llegar al caso base (Análogamente a un árbol), el algoritmo se empieza a devolver' o a retornar los valores y contrasta si al menos una de las dos 'ramas' o lo_tomo y no_lo_tomo es verdadera, y va subiendo hasta retornar verdadero o falso.

3.5 Para comprender con qué base se desarrollan los algoritmos planteados, para cada uno se dispondrá del enunciado propuesto en Codingbat.

3.5 ->2.1.1 "Defina un método recursivo de fibonacci (n) que devuelva el enésimo número de fibonacci, con n = 0 que representa el comienzo de la secuencia."

```
def fibo(n):
  if(n==0):                    #1
    return (0)                 #1
  elif(n==1):                  #1
    return (1)                 #1
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
  else:
    return fibo(n-1) + fibo(n-2)          # 1 +T(n-1) + T(n-2)
```

$$T(n) = \begin{cases} 0 & if\ n = 0 \\ 1 & if\ n = 1 \\ T(n-1) + T(n-2) & if\ n \geq 2 \end{cases}$$

Usando Wolfram para resolverlo, el resultado es:
T(n) = 2^n + c1
(donde C1 es un parámetro arbitrario)

3.5 -> 2.1.2 "Dado un n de 1 o más, devuelva la factorial de n, que es n * (n-1) * (n-2) ...1. Calcule el resultado de forma recursiva (sin bucles)"

```
def fact(n):
  if (n==0):                  # 1
    return 1                  # 1
  else:
    return(fact(n-1) * n)     # 1 + T(n-1) * T(n)
```

$$T(n) = \begin{cases} 1 & if\ n = 0 \\ 1 + T(n-1) * T(n) & if\ n \geq 1 \end{cases}$$

Usando Wolfram para resolverlo, el resultado es:
T(n) = c1 + (1)n
(donde c1 es un parámetro arbitrario)

3.5 -> 2.1.3"Tenemos conejitos parados en una línea, numerados 1, 2, ... Los conejitos impares (1, 3, ...) tienen las 2 orejas normales. Los conejos pares (2, 4, ...) diremos que tienen 3 orejas, porque cada uno tiene un pie elevado. Recursivamente devuelve el número de "orejas" en la línea de conejito 1, 2, ... n (sin bucles ni multiplicación) "

```
def ears(n):
  if (n==0):                  # 1
    return(0)                 # 1
  elif (n%2 != 0):            # 2
    return(ears(n-1))+ 3      # 1 + T(n-1)
  elif (n%2 ==0):             # 2
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
  return(ears(n-1)+2)      # 1 + T(n-1)
```

$$T(n) = \begin{cases} 1 & if\ n = 0 \\ 1 + T(n-1) + T(n-1)\ if\ n \geq 1 \end{cases}$$

Usando Wolfram para resolverlo, el resultado es:

$$T(n) = (c_1 + 2)\, 2^{n-1} - 1$$

(donde c1 es un parámetro arbitrario)

3.5 -> 2.1.4"Dada una cadena, calcule recursivamente (sin bucles) el número de caracteres 'x' en minúscula en la cadena".

```
def countx(string):
  if len(string)==0:                    # 1
    return 0                            # 1
  elif (string[len(string)-1])== "x":   # 2
    return 1 + countx(string[:len(string)-1])   # 3 + T(n-1)
  else:
    return countx(string[0:len(string)-1])      # 2 + T(n-1)
```

$$T(n) = \begin{cases} 1 & if\ n = 0 \\ 10 + T(n-1) + T(n-1)\ if\ n \geq 1 \end{cases}$$

Usando Wolfram para resolverlo, el resultado es:

$$T(n) = (c_1 + 20)\, 2^{n-1} - 10$$

(donde c1 es un parámetro arbitrario)

3.5 -> 2.1.5"Dada una matriz de entradas, calcule recursivamente el número de veces que el valor 11 aparece en la matriz. Usaremos la convención de considerar solo la parte de la matriz que comienza en el índice dado. De esta manera, una llamada recursiva puede pasar el índice + 1 para moverse hacia abajo en la matriz. La llamada inicial pasará en el índice como 0. "

```
def array11(array, i=0):
  if len(array)==0:                     # 1
    return 0                            # 1
  elif (array[0]== 11):                 # 1
    return 1 +array11(array[i+1:])      # 2 +T(n-1)
  else:
    return array11(array[i+1:])         # 1 +T(n-1)
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

$$T(n) = \begin{cases} 1 & if\ n = 0 \\ 4 + T(n-1) + T(n-1)\ if\ n \geq 1 \end{cases}$$

Usando Wolfram para resolverlo, el resultado es:

$T(n) = (c_1 + 8)\, 2^{n-1} - 4$

(donde c1 es un parámetro arbitrario)

3.5 Para comprender con qué base se desarrollan los algoritmos planteados, para cada uno se dispondrá del enunciado propuesto en Codingbat

3.5 -> 2.2.1"Dado un conjunto de entradas, ¿es posible elegir un grupo de algunas entradas, comenzando en el índice de inicio, de modo que el grupo sume al objetivo dado? Sin embargo, con la restricción adicional de que todos los 6 deben ser elegidos. (No se necesitan bucles).

```
def Sum_six(index, nums, target):
  if(index >= len(nums)):                                  # 1
    return target==0                                       # 1
  if (nums[index]==6):                                     # 1
   return Sum_six(index+1 ,nums, target - 6)               # 1 + T(n-1)*m
  SeToma = Sum_six(index+1 ,nums, target - nums[index])    # 2 + T(n-1)*m
  NoSeToma = Sum_six(index+1 ,nums, target)                # 1 + T(n-1)*m
  if SeToma or NoSeToma:                                   # 2
    return True                                            # 1
  else:
    return False                                           # 1
```

$$T(n) = \begin{cases} 1 & if\ n \geq len(m) \\ 5 + 3T(n-1) * m & if\ n < len(m) \end{cases}$$

Usando Wolfram para resolverlo, el resultado es:

$T(n) = c_1\, 3^{n-1}\, m^{n-1} + \frac{5 - 5 \times 3^n\, m^n}{1 - 3\, m}$

(donde c1 es un parámetro arbitrario)

3.5 -> 2.2.2 "Dada una matriz de entradas, ¿es posible elegir un grupo de algunas entradas, de modo que el grupo sume al objetivo dado con esta restricción adicional: si un valor en la matriz se elige para estar en el grupo, el valor no debe elegirse inmediatamente después de la matriz. (No se necesitan bucles).

```
def groupNoAdj(index, nums, target):
 if (index >= len(nums)):                                  # 1
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
    return target == 0                                          # 1

    return (groupNoAdj(index + 2, nums, target - nums[index]) # 1 + T(n-2)*m +T(n-1)*m
            or  groupNoAdj(index + 1, nums, target))
```

$$T(n) = \begin{cases} 1 & if\ n \geq len(m) \\ 2 + T(n-1) * m + T(n-2) * m & if\ n < len(m) \end{cases}$$

Usando Wolfram para resolverlo, el resultado es:

$T(n) = c_1\ 2^{-n}\left(m - \sqrt{m}\ \sqrt{m+4}\right)^n + c_2\ 2^{-n}\left(m + \sqrt{m+4}\ \sqrt{m}\right)^n + \frac{2}{1-2m}$

(donde c1 y c2 son unos parámetros arbitrarios)

3.5 -> 2.2.3 "Dada una matriz de entradas, ¿es posible elegir un grupo de algunas de las entradas, de modo que el grupo sume al objetivo dado con estas restricciones adicionales: todos los múltiplos de 5 en la matriz deben incluirse en el grupo. Si el valor que sigue inmediatamente a un múltiplo de 5 es 1, no debe elegirse. (No se necesitan bucles).

```
def groupSum5(index, nums, target):
  if(index >= len(nums)):                                   # 1
    return target==0                                        # 1

  if (nums[index]%5==0):                                    # 2
   return groupSum5(index+1 ,nums, target -nums[index] )    # 1 + T(n-1)*m
  elif (nums[index-1]%5==0) and (nums[index]==1):           # 4
    return groupSum5(index+1 ,nums, target)                 # 1 + T(n-1)*m
  else:
    return groupSum5(index+1 ,nums, target - nums[index])   # 1 + T(n-1)*m
          or groupSum5(index+1 ,nums, target)               # 1 + T(n-1)*m
```

$$T(n) = \begin{cases} 1 & if\ n \geq len(m) \\ 7 + 4T(n-1) * m & if\ n < len(m) \end{cases}$$

Usando Wolfram para resolverlo, el resultado es:

$T(n) = c_1\ 4^{n-1}\ m^{n-1} + \frac{7 - 7 \times 4^n\ m^n}{1-4m}$

(donde c1 es uno parámetro arbitrario)

3.5 -> 2.2.4 "Dada una matriz de entradas, ¿es posible elegir un grupo de algunas entradas, de modo que el grupo sume al objetivo dado, con esta restricción adicional: si hay números en la matriz que son adyacentes y tienen el mismo valor, deben ser todos elegidos o ninguno de ellos elegido. Por ejemplo, con la matriz {1, 2, 2, 2, 5, 2},

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

los tres 2 en el medio deben elegirse o no, todos como grupo. (se puede usar un bucle para encontrar la extensión de los valores idénticos) ".

Algoritmo tomado de: https://repl.it/@ebrahim_akh95/Recursion3

```
def groupSumClump(start, nums, target):
  if (start >= len(nums)):                                    # 1
   return target ==0                                          # 1
  i = start + 1                                               # 2
  while (i < len(nums) and nums[start] == nums[i]):           # 3
    i += 1                                                    # 1
  if (groupSumClump(i, nums, target - ((i - start) * nums[start]))):   # 1 + T(n-1)*m
    return True                                               # 1
  return groupSumClump(i, nums, target)                       # n*m
```

$$T(n) = \begin{cases} 1 & if\ n \geq len(m) \\ 9 + T(n-1) * n * m & if\ n < len(m) \end{cases}$$

Usando Wolfram para resolverlo, el resultado es:

$$T(n) = (1)_n \left(\left(c_1 + 9\sqrt[m]{e} - 9 \right) m^n - \frac{9\sqrt[m]{e} \left(\frac{1}{m}\right)^{-n} \left(\Gamma(n+1) - \Gamma\left(n+1, \frac{1}{m}\right) \right)}{\Gamma(n+1)} \right)$$

(donde c1 es uno parámetro arbitrario)

3.6 Las variables n y m son ambas requerimientos de la función que se desea evaluar, esta función como se desarrolla de manera recursiva debe tener un caso base que permita parar y obtener los resultados, en los casos anteriormente analizados, la variable que estaba en el caso base n y no m, aunque cada vez que se llame nuevamente la función se debe tener en cuenta la variable m, ya que con esta se realiza una operación de comparación o asignación generando así una nueva tarea, la cual va a ser tan grande como lo sea m, de la misma forma funciona para n, pero esta variable como es quien se encuentra directamente relacionada con el caso base, debe disminuir su valor cada vez que se llame nuevamente la función, creando así la tarea de complejidad, la cual se compone por como reduce n y la comparación o asignación de m.

4) Simulacro de Parcial

Nota: para la mayoría de los ejercicios, dado que se ha permitido cursar la materia usando Java y Python, se realizó un código análogo al planteado por los ejercicios y se completó, pero en PYTHON.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4.1) *Las líneas o palabras subrayadas son las que se propone completar del código para línea 3 y línea 4, respectivamente.*

```
def pal(string):
      if len(string)==0 or len(string)==1:
            return True
      if string[0]== string[len(string)-1]:
            return pal(string[1:len(string)-1])
      else:
            return False
```

4.2) *opción c) T(n)=2.T(n/2)+Cn*

4.4) *opción e) La suma de los elementos del arreglo a y es O(n)*

4.5)

4.5.1) *Las líneas o palabras subrayadas son las que se propone completar del código para línea 3, línea 4 y línea 8, respectivamente.*

```
def formas(T):
      if T<0:return 0
      elif (T==0):return 1

f1 = formas(T-3)
f2 = formas(T-5)
f3 = formas(T-7)

return f1 + f2 + f3
```

4.5.2) *opción b) T(n)=T(n−1)+T(n−2)+C*

4.6) *Las líneas o palabras subrayadas son las que se propone completar del código para línea 10 y línea 12, respectivamente.*

```
def Suma(string):
  return SumaAux(string, 0)

def SumaAux(string, i):
  print(i)
  if i>=len(string):
    return 0

  if (i+1) < len(string) and string[i] == string[i+1]:
    return SumaAux(string, i+2)
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
  return  int(string[i]) + SumaAux(string, i+1)
```

4.7) *Las líneas o palabras subrayadas son las que se propone completar del código para línea 9 y línea 10, respectivamente.*

```
def comb(s, i, t):
  if i>= len(s):
    return t == 0;
  if s[i]%2 == 0:
    return comb(s,i+2,t-s[i]) or comb(s,i+1,t)
  return comb(s,i+1,t-s[i]) or comb(s, i+1, t)
```

4.8) *Las líneas o palabras subrayadas son las que se propone completar del código para línea 9 y línea 13, respectivamente.*

```
def cuantas(k,v,n):
  imposible = False
  if k== 0:
    return 1
  if n<=0 and k>=1:
    imposible = True
  elif k <=0:
    imposible = True
  if imposible == True:
    return 0
  ni = cuantas(k,v,n-1)
  nj = cuantas(k-v[n-1],v,n)
  suma = ni + nj
  return suma
```

4.9) *opción c) 22*

4.10) *opción b) 6*

4.11)

4.11.1) *Las líneas o palabras subrayadas son las que se propone completar del código para línea 9 y línea 13, respectivamente*

```
def lucas(n):
if(n==0):
      return (2)
elif(n==1):
      return(1)
else:
      return lucas(n-1)+lucas(n-2)
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4.11.2) *opción c) $T(n)=T(n-1)+T(n-2)+c$, que es $O(2^n)$*

4.12)

4.12.1) *sat*
4.12.2) *math.max(fi,fj)*
4.12.3) *sat*

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473