

# Taller de Herramientas Computacionales

Brenda Paola García Rivas

24.01.19



## Jueves 24 de Enero. Clase 14

Hoy fue una clase muy interesante, se inició con el problema de fibonacci, el cuál es un problema de recursividad; ayer se vió como resolverlo, sin embargo, esta vez el profesor nos explicó una forma más fácil de hacer los llamados para cada número. Este método implica guardar los resultados en una lista, en este se modificará que: el programa observe que esté un número en la lista para después guardarlo y posteriormente usarlo, pero ¿cómo se sabe que el décimo, onceavo, o el número que sea está en la lista?, pues fácil, teniendo en cuenta la longitud de la lista.

Además de esto, también se resolvieron algunas dudas de la tarea, se inició con el problema de la creación de un laberinto, para realizar éste, debemos tener en mente varias cosas:

1. Primero, se identificó en el pizarrón cómo se iba a hacer el laberinto, se intentó hacerlo lo más fácil posible dado que sería únicamente un ejemplo. El laberinto tenía forma de 3x3, en dónde únicamente se podía pasar horizontalmente por la fila de enmedio.
2. Ahora bien, lo siguiente en cuestionarse es: ¿Puedo avanzar hacia enfrente?, si la respuesta es "sí", entonces, se debe mover una columna, y en la coordenada 'y' se le aumentaría un lugar.
3. Lo siguiente que hay que ver es que

```
# -*- coding: utf-8 -*-
L=[[True, True, True, True],      #Se define el laberinto
  [False, False, False, True],
  [True, True, False, True]]
def resolver(L,e):
    print (e)
    n=len(L[0]) #Columna 1
    x=e[0]
    y=e[1]
    if y==n-1: salida
    return e[0]+1,e[1]+1 #Ya llegué
    else: #Si no se ha llegado...
    if L[x][y+1] == False: #Se puede mover una columna más, por eso se aumenta a y
    e=[x,y+1]
    return resolver(L,e)
    else:
    print 'Ya no se puede avanzar'
```