***Clase # 2***

Recordatorio clase anterior

1. **Serie Fibonacci : 0 1 1 2 3 5 8 13…….fibo(k-2) fibo(k-1) fibo(k)**

**Desarrollo de la ecuación de Recurrencia**

**fibo(1) = 0 Para k = 1**

**fibo(2) = 1 Para k = 2**

**fibo(k) = fibo(k-2) + fibo(k-1) Para todo k>= 3**

**fibo(5) = fibo(3) + fibo(4)**

**fibo(5) = (fibo(1) + fibo(2)) + fibo(4)**

**fibo(1) = 0**

**fibo(2) = 1**

**fibo(3) = (0 +1) + fibo(4)**

**fibo(3) = 1 + fibo(4)**

**fibo(4) = (fibo(2) + fibo(3))**

**fibo(2) = 1 + fibo(3)**

**(fibo(1)+fibo(2))**

**0           +  1**

**fibo(4) = 2**

**fibo(3) = 1 + 2**

**= 1 + 2**

**Algoritmo Recursivo**

**int fibo(int k)**

|  |
| --- |
| **Ecuación de Recurrencia** |
| **F(1)=0**  **F(2)=1**  **F(k) = F(k-1)+F(k-2)** |

**{**

**if k < 2**

**return k - 1**

**else**

**return fibo(k-2) + fibo(k-1)**

**}**

**Algoritmo Iterativo**

**int fibo2(int n)**

{

k1 = 0; k2 = 1; i = 3

while i <= n

{

k3 = k1 + k2

k1 = k2; k2 = k3

i = i + 1

}

return k3

}

**Programación Dinámica**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  | 0 | 1 | 1 | 2 | 3 | 5 |
|  |  |  | i | i | i | i |

def f(fibo,n):

fibo[1] = 0

fibo[2]=1

n = 6

i = 3

While i < = n:

fibo[i] = fibo[i-2]+ fibo[i-1]

i=i+1

#end while

return fibo[i-1]

#end def

Suma recurrente dos números (multiplicación de a\*b)

Ej : 3 \*  4 ⇒ 3 se suma cuatro veces

Ec. de Recurrencia

S(a,b) = 0 ssi b = 0

S(a,b) = a + S(a,b-1)  ssi b > 0

Ej : S(3,4) = 3 + S(3,3) = 12

3 + S(3,2) = 9

3 + S(3,1) = 6

3 + S(3,0) = 3

0 = 0

Algoritmo recurrente

int S(int a; int b)

{

if b == 0

return 0

else

return a + S(a,b-1)

}

**Programación Dinámica**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |  |
| 3 | 6 | 9 | 12 |  |
|  | i | i | i | i |

def mult(S,a,b):

S.append(a)

n=b

i=1

while i<=n:

S.append(S[0]+S[i-1])

i=i+1

#end while

return S[i-2]

#Programa principal

S=[]

resultado = mult(S,3,4)

print("la multiplicación de 3 \* 4 es :",resultado)

Problema del LOTO : ¿Cuántas tarjetas hay que jugar para acertar el Loto?

Es coger 6 números entre 1 y 41 sin repetición

De 41 números debo tomar 6 ===> 41!/(6!\* (41-6)!)

Combinatoria de n sobre K è n!/(k! (n-k)!)

Solución:

F(0) = 1

F(n) = n \* F(n-1)

Ej :

n=7 y k=5

f :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 1 | 2 | 6 | 24 | 120 | 720 | 5040 |

Solución en Python:

def fact(n):

i = 1

f = 1

factorial=[]

factorial.append(f)

while i<=n:

f = factorial[i-1]\* i

factorial.append(f)

i = i + 1

#end while

return factorial

#end def

n = int(input("Ingrese valor n : "))

k = int(input("Ingrese valor k : "))

lista = fact(n)

print(lista)

targetas = lista[n]/(lista[k]\*lista[n-k])

print(targetas)

***Actividad de la semana***

1.- Los resultados obtenidos son correctos??

2.- Si la cantidad de números a elegir en muy grande. ¿Funciona el programa?

3.- Control de entrada(evaluación formativa):

Hacer un algoritmo recursivo que busque la notas de un estudiante, para esto se sabe que:

1. Hay 3 lista (rut, notas, código de la asignatura)
2. Las lista están ordenadas por rut de mayor a menor

Grafosq

Def : Grafo es un tupla (V,A),

donde :

A : Son línea que une dos puntos

V : Es punto en el espacio

EJ :

Triángulo V= 3 A = 3

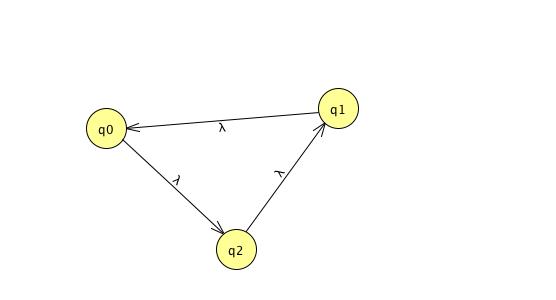
Cuadrado V = 4 A = 4

Árbol : Es un grafo sin ciclos.

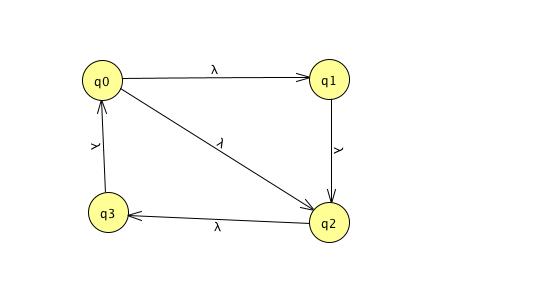
Grados o Valencia : Es cantidad de aristas que salen de un vértice.

Secuencia de grados : Representa a un grafo

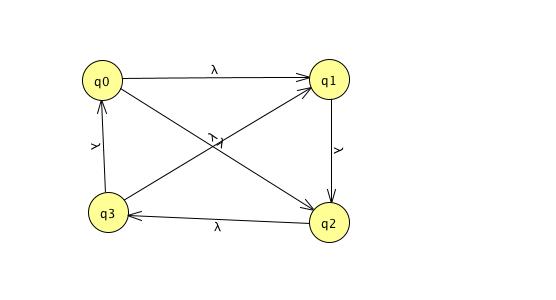
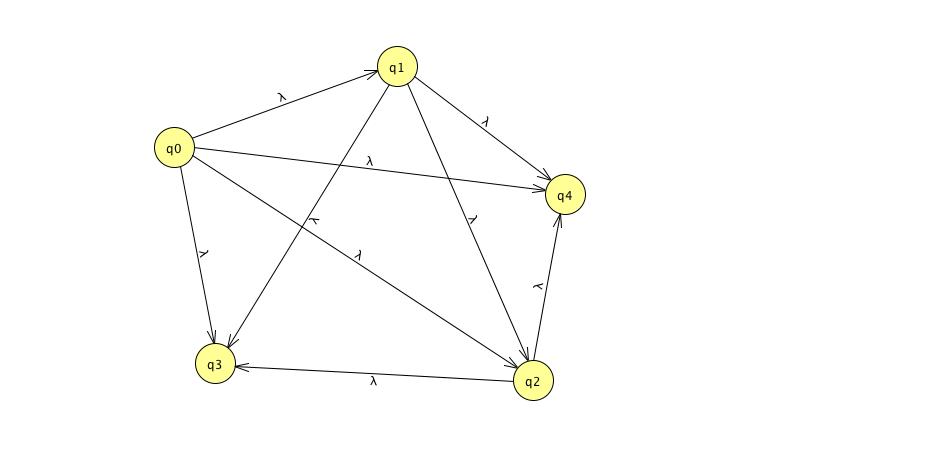
Ej : 2:2:2 Triángulo



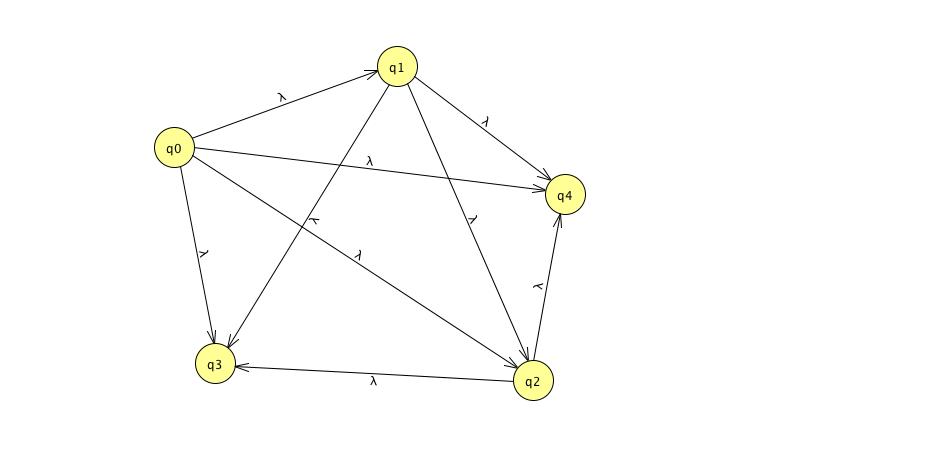
3:3:2:2 Cuadrado con línea cruzada



3:3:3:3 Cuadrado con líneas cruzadas



4:4:4:3:3 Dibujo



5:5:4:4:3:3:3:3

Algoritmo de las Valencias :

5:5:4:4:3:3:3:3

4:3:3:2:2:3:3 ordenar de mayor a menor

4:3:3:3:3:2:2

2:2:2:2:2:2 ordenar de mayor a menor

2:2:2:2:2:2

1:1:2:2:2 ordenar de mayor a menor

2:2:2:1:1

1:1:1:1 ordenar de mayor a menor

1:1:1:1

0:1:1 ordenar de mayor a menor

1:1:0

0:0 è La secuencia representa un grafo