Notación asintótica

Sandoval Huerta Carlos Arturo – Análisis de algoritmos

```
def imprimir_lista(lista):
    for i in lista:
        print(i)
    print("\n")

def main():
    for i in range(1,101):
        lista.append(random.randint(0, 50))

        start_time = time.time()
        imprimir_lista(lista)
        time.sleep(.2)
        end_time = time.time()

        elapsed_time = end_time-start_time
        elapsed_times.append(elapsed_time)
        print("\n")

    plt.plot(range(1,101),elapsed_times)
    plt.xlabel("Iteracion")
    plt.ylabel("Tiempo (s)")
    plt.title('Tiempo transcurrido')
    plt.show()
```

Big O = O(N*N)

```
lista=[]
elapsed_times=[]

def imprime_ultimo(lista):
    print(lista[len(lista)-1])

def main():
    for i in range(1,101):
        lista.append(random.randint(0, 50))
        start_time=time.time()
        imprime_ultimo(lista)
        time.sleep(.2)
        end_time=time.time()
        elapsed_time=end_time-start_time
        elapsed_times.append(elapsed_time)
        print("\n")

    plt.plot(range(1,101),elapsed_times)
    plt.xlabel("N")
    plt.ylabel("Tiempo")
    plt.title("Tiempo transcurrido.")
    plt.show()


if __name__=="__main__":
    main()
```

Big O = O(N)

```
elapsed_times=[]

def multiplicarMatriz(N):
    listaPrincipal = []
    listaPrincipalMultiplicada = [[0 for _ in range(N)] for _ in range(N)]
    for i in range(N):
        listaSecundaria = []
        for j in range(N):
            listaSecundaria.append(random.randint(0, 50))
        listaPrincipal.append(listaSecundaria)


    for i in range(len(listaPrincipal)):
        for j in range(len(listaPrincipalMultiplicada)):
            listaPrincipalMultiplicada[i][j]=(listaPrincipal[i][0]*
                                              listaPrincipal[i][j])


    print("Matriz Original: ",listaPrincipal, "\n")
    print("Matriz multiplicada",listaPrincipalMultiplicada,"\n")

def main():
    N=1
    for i in range(11):
        start_time=time.time()
        multiplicarMatriz(N)
        time.sleep(.4)
        end_time=time.time()
        elapsed_time=end_time-start_time
        elapsed_times.append(elapsed_time)
        N+=1

    plt.plot(range(11),elapsed_times)
    plt.xlabel("N")
    plt.ylabel("Tiempo")
    plt.title("Tiempo Transcurrido")
    plt.show()
```

Big O = O(N^2) + O(N^2)

= O(2N^2)

= O(11*N^2)

= O(N^2)

```python
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

def get_orientation(p, q, r):
    orientation = (q.y-p.y) * (r.x-q.x) - (q.x-p.x) * (r.y-q.y)
    if orientation > 0:
        return 1
    elif orientation <0:
        return -1
    return 0


def compare_orientation(a, b):
    if a != b:
        return 1
    else:
        return 0

def plot_points(p1, q1, p2, q2):
    plt.scatter([p1.x, q1.x, p2.x, q2.x], [p1.y, q1.y, p2.y, q2.y], c='red',
                marker='o', label='Points')
    plt.plot([p1.x, q1.x], [p1.y, q1.y], c='blue', label='Segment 1')
    plt.plot([p2.x, q2.x], [p2.y, q2.y], c='green', label='Segment 2')

def numericalAlgorithm():
    LIMITE_N=10
    for i in range(5,LIMITE_N,5):
        print("NUMERO EN i: " , i)
        for j in range(0,i):
            randP1X = random.randint(0,50)
            randP1Y = random.randint(0,50)
            randQ1X = random.randint(0,50)
            randQ1Y = random.randint(0,50)
            randP2X = random.randint(0,50)
            randP2Y = random.randint(0,50)
            randQ2X = random.randint(0,50)
            randQ2Y = random.randint(0,50)
```

```python
            p1 = Point(randP1X, randP1Y)
            q1 = Point(randQ1X, randQ1Y)
            p2 = Point(randP2X, randP2Y)
            q2 = Point(randQ2X, randQ2Y)

            a = get_orientation(p1, q1, p2)
            b = get_orientation(p1, q1, q2)
            c = get_orientation(p2, q2, p1)
            d = get_orientation(p2, q2, q1)
            print("NUMERO EN J: " , j)
            if compare_orientation(a, b) and compare_orientation(c, d):
                print("Hay interseccion")
            else:
                print("Las rectas no coinciden")


            plot_points(p1, q1, p2, q2)
            plt.legend()
            plt.xlabel('X-axis')
            plt.ylabel('Y-axis')
            plt.grid()
            plt.show()

def main():
    start_time = time.process_time()
    numericalAlgorithm()
    end_time = time.process_time()
    cpu_time=end_time-start_time
    print(f"Tiempo: {cpu_time} segundos")
```

Big O = O(n^2)