Taller 01

Isaac Gonzalez

Ejericio 1

La sumatoria $1 + 1/2 + 1/4 + 1/8 \ldots$ tal que el error absoluto $e_{abs} < 10^{-1}$.

In [1]:
```python
error = 10**-1
suma = 0
n = 0
while abs(2 - suma) > error:
    suma += 1/(2**n)
    n += 1
print(n)
```

```
5
```

Ejercicio 2 (Bubble sort)

XSDF

Modifique el Algoritmo

Determine el número de comparaciones realizadas al ordenar la serie 5, 4, 3, 2, 1

In [2]:
```python
def bubble_sort(a):
    n = len(a)
    print("i | Vector")
    print("--|----------------")
    cont = 0
    for i in range(n):
        swapped = False
        for j in range(1, n - i):
            if a[j] < a[j - 1]:

                a[j], a[j - 1] = a[j - 1], a[j]
                swapped = True
                cont += 1
        print(f"{i} | {a}")
        print(f"Intercambios en esta pasada: {cont}")

        if not swapped:
            break
    return a

#Prueba de escritorio
```

```python
v1 = [3, 2, 5, 8, 4, 1]
print("Vector original:", v1)
v_sorted = bubble_sort(v1.copy())
print("Resultado ordenado:", v_sorted)

#Caso de prueba
v2 = [ -1, 0, 4, 5, 6, 7]
v3 = [5, 4, 3, 2, 1]
print("\n Caso 2 :",  bubble_sort(v2.copy()))
print("\n Caso 3 :", bubble_sort(v3.copy()))
import random
#v3 = [random.randint(-200, 145) for _ in range(100000)]
#print("\n Caso 3 (random) :", bubble_sort(v3.copy()))
```

```
Vector original: [3, 2, 5, 8, 4, 1]
i | Vector
--|----------------
0 | [2, 3, 5, 4, 1, 8]
Intercambios en esta pasada: 3
1 | [2, 3, 4, 1, 5, 8]
Intercambios en esta pasada: 5
2 | [2, 3, 1, 4, 5, 8]
Intercambios en esta pasada: 6
3 | [2, 1, 3, 4, 5, 8]
Intercambios en esta pasada: 7
4 | [1, 2, 3, 4, 5, 8]
Intercambios en esta pasada: 8
5 | [1, 2, 3, 4, 5, 8]
Intercambios en esta pasada: 8
Resultado ordenado: [1, 2, 3, 4, 5, 8]
i | Vector
--|----------------
0 | [-1, 0, 4, 5, 6, 7]
Intercambios en esta pasada: 0

 Caso 2 : [-1, 0, 4, 5, 6, 7]
i | Vector
--|----------------
0 | [4, 3, 2, 1, 5]
Intercambios en esta pasada: 4
1 | [3, 2, 1, 4, 5]
Intercambios en esta pasada: 7
2 | [2, 1, 3, 4, 5]
Intercambios en esta pasada: 9
3 | [1, 2, 3, 4, 5]
Intercambios en esta pasada: 10
4 | [1, 2, 3, 4, 5]
Intercambios en esta pasada: 10

 Caso 3 : [1, 2, 3, 4, 5]
```

Algoritmo 3

XSDF

In [6]:

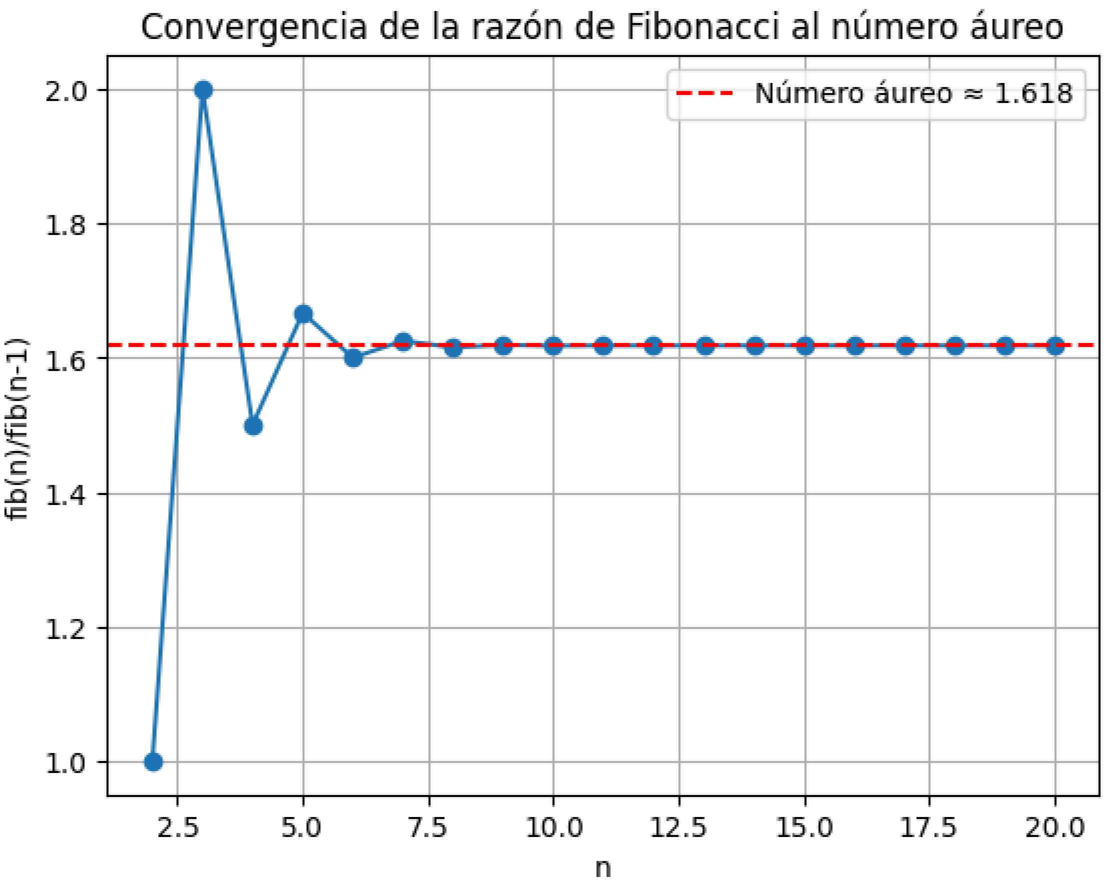
```
#fibonacci
def fibonacci(n):
    if n == 0:
        return 0
    x,y = 0,1
    for i in range(1, n):
        z = x + y
        x, y = y, z
    return y

#Prueba de escritorio

print("Fibonacci 11 : ", fibonacci(11))
print("Fibonacci 84 : ", fibonacci(84))
print("Fibonacci 1531 : ", fibonacci(1531))
```

```
Fibonacci 11 :  89
Fibonacci 84 :  160500643816367088
Fibonacci 1531 :  407936176052377669101778911015323059541693566794692519680122463207
854422013990100626081201338987968421592147014912276452966402513511180974144525129433
779239442408519013425119983218373176872312001814049893514987716130911286090664428422
730299315959724514396175573827117595933842787346948580100247676460231570134185935472
69
```

Gráfica del numero Aureo

In [7]:

```
#Grafica de la serie de Fibonacci
import matplotlib.pyplot as plt

def fib_iterative(n):
    if n == 0:
        return 0
    x, y = 0, 1
    for i in range(1, n):
        z = x + y
        x, y = y, z
    return y

# Calcular serie y cocientes
N = 20
fibs = [fib_iterative(i) for i in range(1, N + 1)]
ratios = [fibs[i] / fibs[i - 1] for i in range(1, len(fibs))]

# Graficar
plt.plot(range(2, N + 1), ratios, marker='o')
plt.axhline(y=(1 + 5**0.5)/2, color='r', linestyle='--', label='Número áureo ≈ 1.61
plt.xlabel('n')
plt.ylabel('fib(n)/fib(n-1)')
plt.title('Convergencia de la razón de Fibonacci al número áureo')
plt.legend()
plt.grid(True)
plt.show()
```



Extra:

Usando el Algoritmo 03 y la aritmética de redondeo con 3 cifras, determine la iteración desde la cual el error relativo de $\frac{y_{i+1}}{y_i}(i > 0)$ con respecto a $\frac{1+\sqrt{5}}{2}$ está dentro de 10^{-5}

In [6]:
```
def fibonacci_redondeado(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        x, y = 0, 1
        for i in range(2, n + 1):
            z = x + y  # sin redondeo dentro del cálculo
            x, y = y, z
        return y


phi = (1 + 5**0.5) / 2      # número áureo
tolerancia = 1e-5            # 10^-5
i = 1                        # empezamos desde i=1

while True:
    y_i = fibonacci_redondeado(i)
    y_next = fibonacci_redondeado(i + 1)
```

```
    if y_i == 0:
        i += 1
        continue

    razon = round(y_next / y_i, 3)
    razon_real = y_next / y_i
    error_rel = abs((razon_real - phi) / phi)

    if error_rel < tolerancia:
        break

    i += 1
print(f"Iteración donde el error relativo está dentro de la tolerancia ({tolerancia
```

```
Iteración donde el error relativo está dentro de la tolerancia (1e-05): 13
```

Algoritmo 04

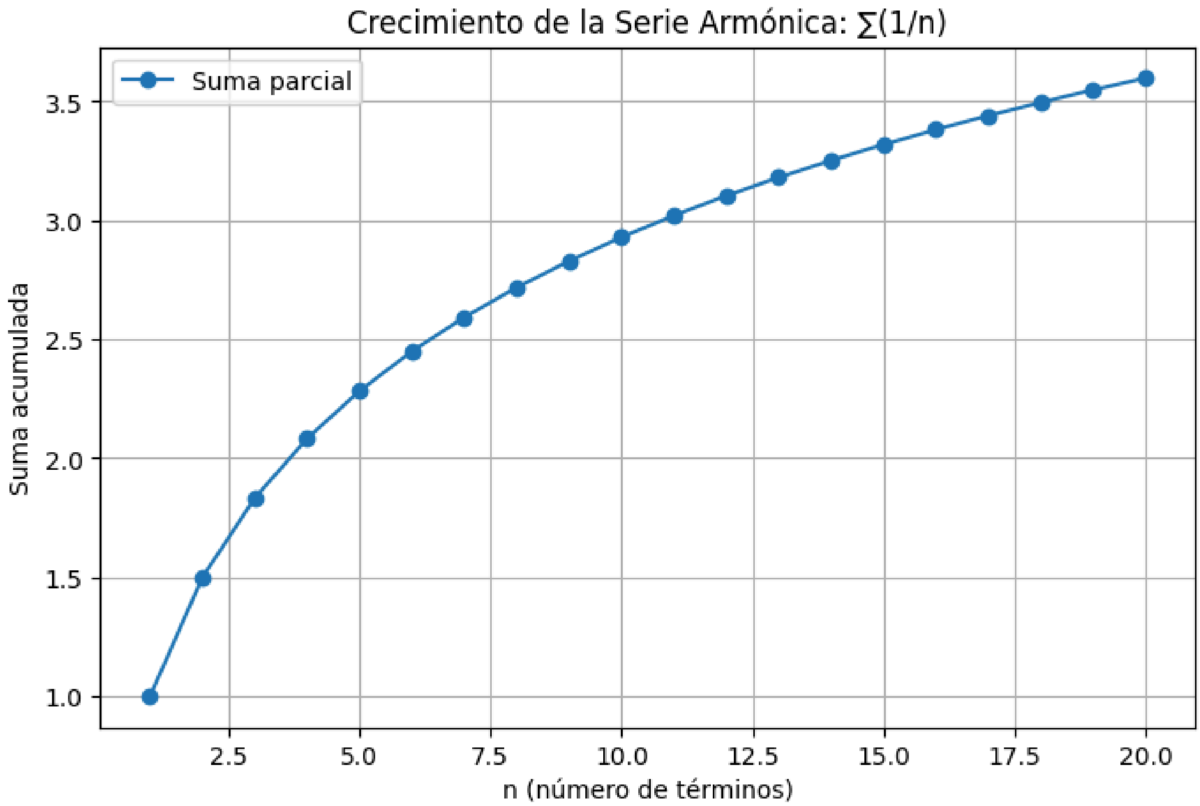
Implemente la serie geométrica

$\sum_{n=1}^{\infty} \frac{1}{n}$

A qué válor converge el algoritmo 04?

In [13]:
```
import matplotlib.pyplot as plt

N = 20
suma = 0
valores_n = []
sumas_parciales = []

for i in range(1, N + 1):
    suma += 1 / i
    valores_n.append(i)
    sumas_parciales.append(suma)

print(f"Suma parcial con {N} términos: {suma}")

plt.figure(figsize=(8, 5))
plt.plot(valores_n, sumas_parciales, marker='o', linestyle='-', label='Suma parcial
plt.xlabel('n (número de términos)')
plt.ylabel('Suma acumulada')
plt.title('Crecimiento de la Serie Armónica: ∑(1/n)')
plt.grid(True)
plt.legend()
plt.show()
```

```
Suma parcial con 20 términos: 3.597739657143682
```


Crecimiento de la Serie Armónica: ∑(1/n)
Suma parcial
Suma acumulada
n (número de términos)
1.0
1.5
2.0
2.5
3.0
3.5
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0