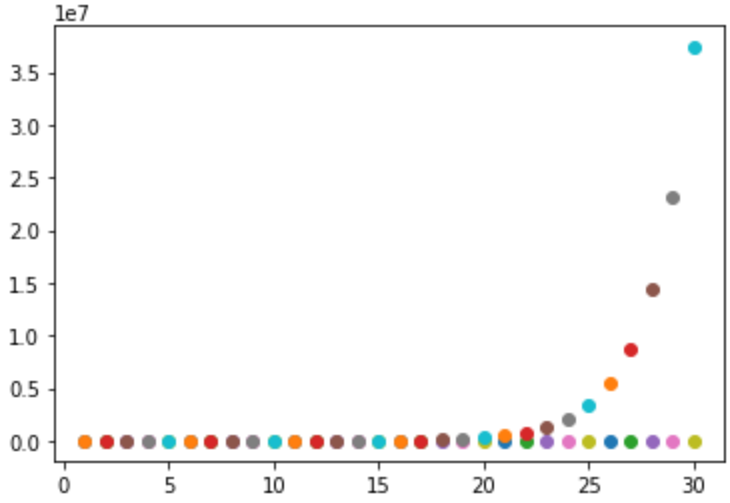
In [20]:
```
import ctypes

# Load the shared library
libpmu = ctypes.CDLL('./libpmu.so')

# Now you can call the functions from the shared library
libpmu.initialize_pmu_counters()
print(libpmu.get_cycle_count())
```

```
5395
```

In [1]:
```
import ctypes
import time
import psutil
import numpy as np
import matplotlib.pyplot as plt

# Load the shared library
libpmu = ctypes.CDLL('./libpmu.so')

def recur_fibo(n):
    if n <= 1:
        return n
    else:
        return(recur_fibo(n-1) + recur_fibo(n-2))

# Vary the number of terms from 1 to 30
for n in range(1, 31):
    times = []
    cycle_counts = []

    # Take multiple trials for each variation
    for _ in range(3):
        # Initialize the cyclecounter
        libpmu.initialize_pmu_counters()

        # Get the 'before' time and cycle count
        start_time = time.time()
        start_cycles = libpmu.get_cycle_count()

        # Run the recur_fibo function on a CPU 1
        psutil.Process().cpu_affinity([1])
        recur_fibo(n)

        # Get the 'after' cycle count and time
        end_cycles = libpmu.get_cycle_count()
        end_time = time.time()

        # Get the cycle count and the amount of time used
        cycle_counts.append(end_cycles - start_cycles)
        times.append(end_time - start_time)

    # Calculate the average and standard deviation
    avg_time = np.mean(times)
    avg_cycles = np.mean(cycle_counts)
    error_time = np.std(times) / np.sqrt(3)
    error_cycles = np.std(cycle_counts) / np.sqrt(3)
```

```
    # Plot the average results for varying 'n' along with error bars of your measuremen
    plt.errorbar(n, avg_time, yerr=error_time, fmt='o')
    plt.errorbar(n, avg_cycles, yerr=error_cycles, fmt='o')

plt.show()
```



In [ ]:

In [ ]: