Zadanie 4.

Zaimplementuj szybki algorytm wyznaczania dyskretnej transformaty Fouriera (FFT).\ • Zaimplementuj naiwną procedurę mnożenia wielomianów reprezentowanych przez współczynniki.

• Zaimplementuj szybką procedurę mnożenia wielomianów reprezentowanych przez współczynniki (wykorzystaj FFT)

• Eksperymentalnie zbadaj szybkość obu procedur dla rosnącego stopnia wielomianu. Wyniki przedstaw na wykresie czasu działania algorytmu od rozmiaru wielomianu.

In [23]:
```
import numpy as np
from timeit import default_timer as timer
from numpy.fft import rfft, irfft
from timeit import default_timer as timer
from matplotlib import pyplot as plt


def fft(x):
    '''Oblicza FFT dla sygnału x o długości będącej potęgą 2.'''
    N = len(x)
    if N == 2:
        return np.array([x[0] + x[1], x[0] - x[1]], dtype=complex)
    Xp = fft(x[::2])
    Xn = fft(x[1::2])
    w = np.exp(-2 * np.pi * 1j * np.arange(N // 2) / N)
    X = np.hstack((Xp + w * Xn, Xp - w * Xn))
    return X


def naive_multiply(A, B):
    ''' Naiwne mnożenie '''
    start = timer()
    m = len(A)
    n = len(B)
    # tworzymy tablicę o wymiarach n + m - 1
    prod = [0] * (m + n - 1)

    for i in range(m):
        for j in range(n):
            prod[i + j] += A[i] * B[j]
            end = timer()

    return prod, end - start


def fft_multiply(arr_a, arr_b):
    start = timer()
    L = len(arr_a) + len(arr_b)
    # wywołanie funkcji np.rfft - zwraca jednowymiarową dyskretną transformatę fouriera
    a_f = rfft(arr_a, L)
    b_f = rfft(arr_b, L)
    end = timer()
    return irfft(a_f * b_f), end - start


A = []
B = []
T1 = []
T2 = []

for _ in range(10):
    # losowanie tablic(wielomianów)
    A.append(np.random.randint(1, 10))
    B.append(np.random.randint(1, 10))

    # wywołanie obu sposobów mnożenia
    ret1, time1 = naive_multiply(A, B)
    ret2, time2 = fft_multiply(A, B)

    # dodanie czasu do tablic(wyników)
    T1.append(time1)
    T2.append(time2)

    print(re1)
    print(ret2)

plt.plot(T1, color='r', label='Naiwne mnożenie')
plt.plot(T2, color='g', label='Mnożenie FFT')
plt.title('Czas mnożenia wielomianów dwoma metodami', size=18)
plt.xlabel('Rozmiar')
plt.ylabel('Czas [s]')
plt.legend()
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_4256/1642271091.py in <module>
     62     T2.append(time2)
     63
---> 64     print(re1)
     65     print(ret2)
     66

NameError: name 're1' is not defined
```

In [ ]: