```python
import numpy as np
import math
import matplotlib.pyplot as plt

def combos(n, i):
  # This function evaluates the combinations.
  return math.factorial(n) / (math.factorial(n-i)*math.factorial(i))

def binom_eu(S0, K , T, r, sigma, N, option_type):
  # This function evaluates the price of a
  # European option, given option type.

  # constatns
  dt = T/N
  u = np.exp(sigma * np.sqrt(dt))
  d = np.exp(-sigma * np.sqrt(dt))
  p = (np.exp(r * dt) - d)  /  (u - d) #risk-neutral probability
  value = 0

  # for-loop to evaluate the option payoffs
  for i in range(N+1):
    node_prob = combos(N, i) * p**i * (1-p)**(N-i) # probability at
the node
    ST = S0 * u**i * d**(N-i)  # price of underlying asset

    # use appropriate payoff for the option type
    if option_type == 'call':
      value += max(ST-K,0) * node_prob
    elif option_type == 'put':
      value += max(K-ST, 0) * node_prob
    else:
      raise ValueError("Option type must be 'call' or 'put'" )

  # discount the option price
  discounted_value =  value * np.exp(-r*T)

  return discounted_value

# Initial constants for pricing
N = 4
S0  = 100
T = 0.5
sigma = 0.4
K = 105
r = 0.05

# increasing the number of tree and convergance of option price
Ns = [2, 4, 6, 8, 10, 20, 50, 100, 200, 300, 400,500, 600]
for n in Ns:
```

```python
    c = binom_eu(S0, K, T, r,sigma, n, 'call')
    print(f'Price is {n} steps is {round(c,2)}')
```

```
Price is 2 steps is 9.99
Price is 4 steps is 10.29
Price is 6 steps is 10.35
Price is 8 steps is 10.37
Price is 10 steps is 10.37
Price is 20 steps is 10.34
Price is 50 steps is 10.27
Price is 100 steps is 10.22
Price is 200 steps is 10.22
Price is 300 steps is 10.23
Price is 400 steps is 10.22
Price is 500 steps is 10.22
Price is 600 steps is 10.22
```