# Authors: Guhan Jeong, Hector Gonzalez, Brianna Taylor
# Date: 17 March 2025
# Description: Homework Assignment 3

1. **Implement the binomial coefficient using divide and conquer. Please use Algorithm 3.1 on page 96 to guide your implementation.**

   # Wil be done using VS Code
   # Everything will be commented explaining the process at which it all works
   https://github.com/hgonzalez542/Algorithm-Analysis

2. **Implement the binomial coefficient using dynamic programming. Please use Algorithm 3.2 on page 99 to guide your implementation. You may use a 2D list in Python to implement the array.**

   # WIll be done using VS Code
   # Everything will be commented explaining the process at which it all works
   https://github.com/hgonzalez542/Algorithm-Analysis

3. **Which of the above is more efficient? Why do you think that is the case?**

Between the two versions, the dynamic programming approach (bincoeff_dp.py) is more efficient than the divide and conquer method (bincoeff_ds.py). The recursive version solves the same problems over and over again, which takes a lot more time—especially as the numbers get bigger. It grows really fast in how many steps it needs, kind of like a tree that keeps splitting. On the other hand, the dynamic programming version saves the answers to smaller problems in a table and reuses them, so it only solves each smaller piece once. This makes it much faster overall.

The reason dynamic programming works better here is because it avoids doing extra work. Instead of re-solving the same problem again and again, it builds up the answer step by step using already-solved parts.

For anyone that goes into this field these practices are good to know, especially learning algorithms because it’s a good example of how thinking ahead and planning your steps can save a lot of time and make your code way more efficient.

-

4. **Establish Equality 3.1 given below:**

$$\binom{n}{k} = \begin{cases} \binom{n-1}{k-1} + \binom{n-1}{k} & 0 < k < n \\ 1 & k = 0 \quad \text{or} \quad k = n. \end{cases} \tag{3.1}$$