第一题

无

第二题

该函数使用了**记忆化**技术来优化重复计算。记忆化是一种通过存储先前计算结果的技术，能够在相同输入再次出现时复用结果，从而减少冗余的计算。如果不使用记忆化，递归调用过程中可能会出现重复计算同一个 x 的问题。通过将结果存储在 memo 字典中，函数可以避免重复计算，从而在大输入情况下显著提高效率。——参考：Python 搜索算法高级技巧：动态规划与记忆化搜索精讲 - CSDN 文库

F(x, memo) 函数，如果 x 之前已经计算过（即存在于 memo 字典中），则直接从 memo 中返回其结果，以避免重复计算；calculate_F_for_list(input_list) 函数，接收一个输入列表 input_list，并为列表中的每个元素计算 F(x)。

第三题

动态规划数组 dp：dp[i][j] 表示用 i 个骰子掷出和为 j 的方式数。

初始化：dp[0][0] = 1，表示用 0 个骰子得到和为 0 的方式有 1 种（即什么都不做）。

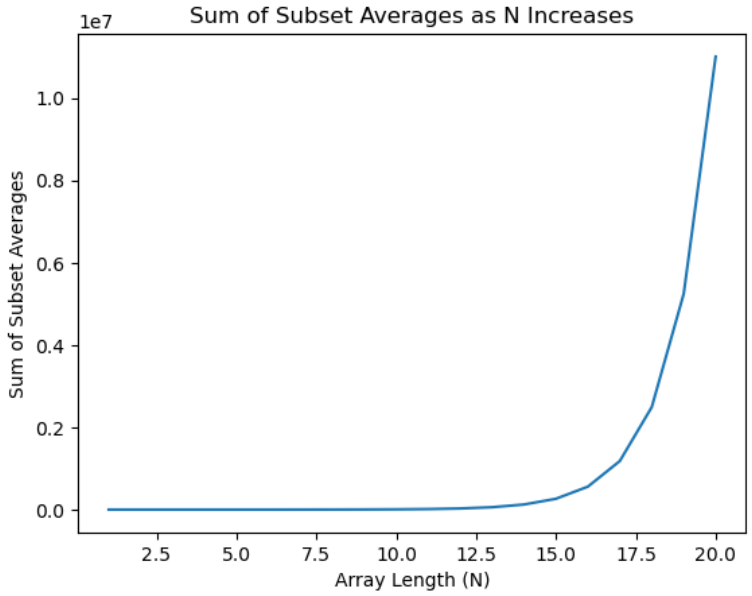
状态转移：对于每个骰子 i 和每个和 j，考虑骰子的每一面（即从 1 到 6），将前一个状态 dp[i-1][j-k] 累加到当前状态 dp[i][j] 中。只有当 j-k >= 0 时，才可以进行这种累加，确保不计算无效的和。——参考：Python 搜索算法高级技巧：动态规划与记忆化搜索精讲 - CSDN 文库

第四题

子集数量的指数增长：随着数组长度的增加，生成的子集数量呈现指数增长，这导致子集平均值之和也呈现相似的增长趋势。

动态变化可视化：通过可视化图形，我们能够清晰地看到这种变化趋势，表明随着数组规模的扩大，子集平均值总和的增长速度非常快。——参考：Python 搜索算法高级技巧：动态规划与记忆化搜索精讲 - CSDN 文库

由于 1-100 计算量太大，故在此只用 1-20 进行计算，下图即为所绘制的图。



第五题

create_random_matrix(N, M) 函数该函数生成一个 N 行 M 列的矩阵，矩阵中的元素随机填充为 0 或 1。Count_path(matrix, i, j, N, M) 函数中每个位置 (i, j)表示每个位置。