Assignment 01

1. Flowchart

Report your output when a = 10, b = 5, c = 1.

首先，a > b 成立，进入第一个条件块。进一步比较 b > c 成立，因此选择 a+b−10×c 作为计算公式。

计算 10+5−10×1=15−10=5。

输出：Result: 5 (a > b > c)。

2. Continuous ceiling function

解题思路如下：

（1）给定一个递归函数 F(x)，其中：

基础情况是：F(1) = 1。

递归情况是：F(x) = F(ceil(x / 3)) + 2 * x，其中 ceil 表示向上取整。

递归的计算涉及对输入值逐步递归调用，最终求出每个 x 的结果。

（2）递归函数 F(x)的设计：

通过 a = {1: 1} 确保当 x == 1 时，函数直接返回 1，不再递归。

对于任意大于 1 的 x， F(x) 被定义为 F(ceil(x / 3)) + 2 * x。这里 ceil(x / 3) 用于处理向上取整的递归。

我们使用 a 来保存已经计算过的 F(x) 的结果。

（3）主函数 compute_values(lst)：

接收一个列表 lst，然后对列表中的每个元素 x 调用 F(x) 递归函数，并将结果存入一个新列表。

通过列表推导式 [F(x) for x in lst] 实现对每个 x 的计算。

（4）给定输入列表 lst = [10, 15, 20]，函数会依次计算 F(10)、F(15) 和 F(20) 的值。每次递归计算 F(x)，若遇到之前计算过的值，则直接从 a 中返回，从而避免重复计算。

（5）以 F(15) 为例，递归过程如下：

首先 F(15) 需要计算 F(ceil(15 / 3))，即 F(5)，然后再加上 2 * 15。

继续计算 F(5)，依赖于 F(ceil(5 / 3))，即 F(2)，再加上 2 * 5。

计算 F(2)，依赖于 F(ceil(2 / 3))，即 F(1)，再加上 2 * 2。

最终 F(1) 已经在 a 中，返回 1。

通过这样的递归计算，最终得到 F(10) 的值，并保存在 a 中。

3. Dice rolling

解题思路如下：

（1）给定 10 个 6 面骰子（每个骰子的面值为 1 至 6），我们需要计算从骰子中得到和为 x 的不同方式数目。

（2）可以通过构建二维 DP 表 dp[i][j]来计算每个骰子数目下对应的和的可能性，其中：

i 表示使用 i 个骰子， j 表示骰子总和为 j 的情况。

dp[i][j]存储的是使用 i 个骰子得到总和 j 的不同方式数。

（3）初始条件：

dp[0][0] = 1：使用 0 个骰子得到和为 0 的方式有 1 种。

（4）当我们有 i 个骰子时，如果当前骰子的面值为 k ，我们可以通过从前一个骰子的和 j – k 状态转移过来。因此：

$$dp[i][j] = \sum_{k=1}^{6} dp[i-1][j-k] \quad (\text{前提是 } j-k \geq 0)$$

这意味着我们可以通过前一个骰子的和为 j-k 的方式数来推导出当前和为 j 的方式数。

（5） Find_number_of_ways 函数：

该函数接收两个参数 n 和 x，分别表示骰子的数量和目标和。它使用上述动态规划表来计算出得到和为 x 的方式数。

对于每个骰子数 i ， 和 j ，根据前面讨论的状态转移公式填充 DP 表。

最后，返回 dp[n][x]，即使用 n 个骰子得到和为 x 的方式数。

（6） Find_max_number_of_ways 函数：

该函数通过调用 Find_number_of_ways 来计算当 n = 10 时，从和 x = 10 到 x = 60 的不同方式数，并将结果存储在 Number_of_ways 列表中。

找出最大方式数：

使用 max() 函数找到 Number_of_ways 列表中的最大值，并返回对应的和 x 。

最终，函数返回计算的所有方式数列表，以及产生最大方式数的和 x 和对应的方式数。

（7） Find_max_number_of_ways 函数会计算骰子数 n = 10 时，目标和 x 从 10 到 60 的所有情况，并输出能够产生最多方式数的和 x 及其对应的方式数。

（8）运行该代码，输出如下：

当和为 35 时，存在最多的路径数，路径数为 45685。

4. Dynamic programming

解题思路如下：

（1） Random_integer 函数：

该函数生成一个包含 N 个随机整数的数组，数组中的每个元素都是从 0 到 10 范围内随机选择的。

通过 random.randint(0, 10) 实现随机数生成，并返回长度为 N 的列表。

（2） Sum_averages 函数：

计算给定数组所有非空子集的平均值之和。

Sum of averages of all subsets = 1/N * total sum of array elements * $2^{(N-1)}$

式中：

total_sum 是数组元素的总和。

$2^{(N-1)}$ 是每个元素在所有非空子集中出现的次数。这个结论可以通过组合数学来推导：

总子集数量：给定一个包含 N 个元素的集合，它的所有子集的总数为 2^N。这个是因为每个元素都有两种状态：要么包含在子集中，要么不包含在子集中。

非空子集数量：非空子集的数量为 $2^N - 1$，因为其中包含了所有可能的子集组合，但不包括空集。

每个元素的贡献：对于集合中的某个特定元素 x，我们想知道它在多少个子集中出现。对于每个子集，我们可以考虑将这个问题分成两个部分：子集中包含该元素，子集中不包含该元素。因为每个元素都有这两种状态，所以每个元素有一半的子集会包含该元素，另一半不会包含。因此，对于每个元素 x，它会出现在一半的所有子集中。

在非空子集中的次数：由于所有子集的数量为 2^N，且每个元素会在其中一

半的子集（即 2^{N-1} 个子集）中出现，所以每个元素在非空子集中出现的次数也是 2^{N-1}。

1/N 是为了计算所有子集的平均值。

（3） Total_sum_averages 列表：

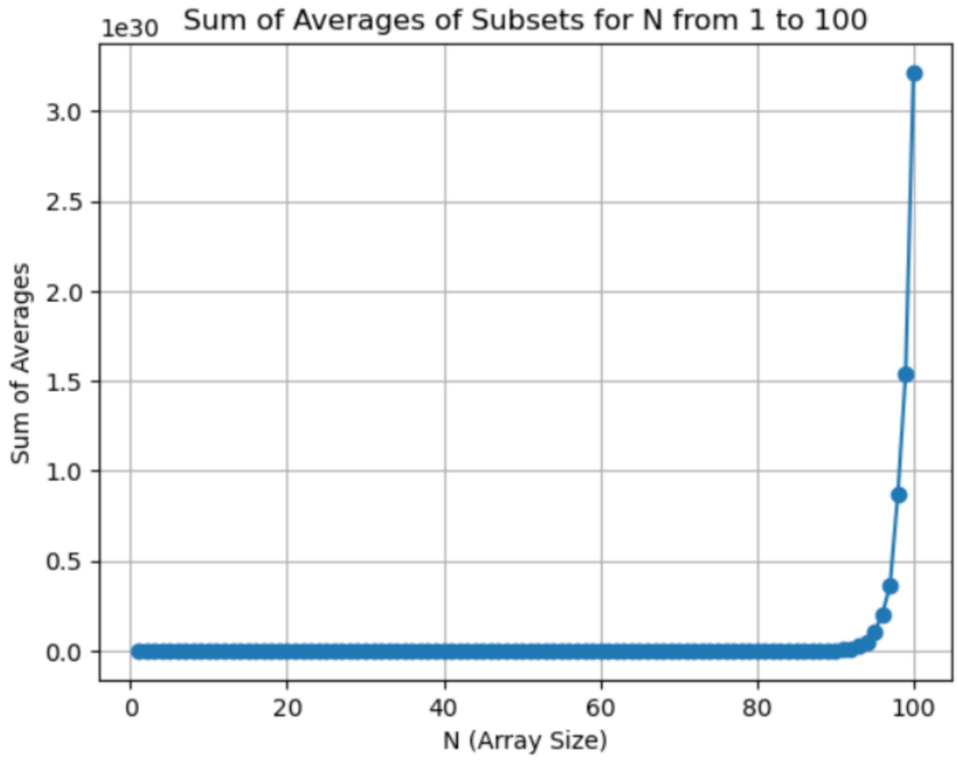
计算 N 从 1 到 100 时，每个随机数组的所有子集平均值之和，并将结果保存到列表中。

通过遍历 N 从 1 到 100，依次生成随机数组，并使用 Sum_averages 计算所有子集的平均值和，将结果存入 Total_sum_averages 列表。

（4）绘制图表：

使用 matplotlib 库中的 plt.plot 函数绘制图表，并设置标题、轴标签、网格等。

（5）结果分析：



从图中可以看到，当数组大小 N 从 1 增加到 100 时，子集平均值总和的变化呈现出极为陡峭的增长趋势。这是因为数组元素的总和随着 N 增加而增大，导致子集的数量和平均值之和成指数增长。具体分析如下：

初始阶段 ($N < 60$)：在 N 较小的范围内，子集平均值的总和保持接近于零。这可能是由于生成的随机整数较小，使得数组总和较小，因此所有子集的平均

值相对较小，导致总和增长缓慢。

快速增长阶段 ($N \geqslant 60$)：从 N 约等于 60 开始，曲线逐渐上升，直到 N 接近 100 时出现了指数级的增长。这是因为随着 N 的增加，数组的长度增加导致子集数量成倍增加，同时数组总和的增大也会导致所有子集平均值的总和迅速上升。

指数增长：在接近 ($N = 100$) 时，图像显示出非常陡峭的上升趋势，表明子集平均值的总和呈现指数增长。这与公式中的 $2^{(N-1)}$ 部分有关，因为每个新增加的元素都会增加大量新的子集，从而显著增加总和。

随着数组大小 N 的增加，子集平均值的总和呈现出指数增长的趋势，特别是在 N 较大的时候。这是因为子集的数量随着数组大小成倍增长，并且数组元素的总和随着 N 增加而变大。

（6）随机生成的数组：[7, 3, 10, 3, 0]，所有子集平均值的总和：8.0。

5. Path counting

解题思路如下：

（1）创建矩阵：

使用 create_matrix 函数创建一个 N * M 的随机矩阵，矩阵中的元素为 0 或 1。

保证矩阵的左上角和右下角的元素为 1，因为它们是起点和终点。

（2）使用动态规划 Count_path 函数来计算从矩阵的左上角到右下角有多少种不同的路径。每次只能向右或向下移动。

（3）如果当前位置的值为 1（可以通过），则检查它是否可以从左边或上方到达，并累加这些路径的数量。

（4）模拟多次实验：

使用 run_simulation 函数，执行多个试验，每次生成一个新的随机矩阵，并计算从左上角到右下角的路径数量。

对所有试验的路径数求均值，以便估计平均路径数量。

（5）返回结果：

最终输出多次实验后的平均路径数量（mean_paths），以及每次试验的路径数列表（total_paths）。

（6）动态规划思路：

创建一个二维数组 dp，其中 dp[i][j] 表示到达坐标 (i, j) 的路径数。

如果当前坐标为 1（即路径是可通行的），我们从上方和左方的路径数累加到当前位置，更新 dp[i][j]。

最终结果保存在 dp[N-1][M-1]，即到达右下角的路径总数。

6. Acknowledgments

Yin zihao explained to me what is asked in problem set 2, 3, 5.

Ouyang wenjian explained to me what is asked in problem set 4.

I sincerely thank the above two students for their detailed answers.