# 12432894 张晨

Task 1

总结：设置三层循环进行运算

when a = 10, b = 5, c = 1, the result is 5； when a =6, b = 4, c = 2, the result is -10

Task 2

使用’//’进行取整实现’ceil’的功能

定义函数 F（x, F(F(ceil(x/3)) + 2x)），使得 F(x) = F(ceil(x/3)) + 2x，从而实现递归运算。

由于第一个函数有两个变量，因此 F(x)应为字典。即 F(x, meno[x])，meno[x] = F(F(ceil(x/3)) + 2x。

输入初始值 F(1) = 1。

最后定义一个索引列表，将字典中的值输出。

总结：可以运行代码，输入一个 list，即可得到 F(ceil(x/3)) + 2x 处理后的目标 list。

Task 3

3.1.1①穷举法

设置 10 个循环，分别从 1-6 取值，代表 6 个骰子可能得到的值，将值相加如果等于目标值则记录，将所有值循环完毕输出记录的次数。!!! 注意若记录的值过大可能会报错。

总结：代码第一行可以设置例子进行测试。如当 X=31 时，方式有 3393610 种。

3.1.2②动态规划法

1.定义状态：我们定义 dp[i][j] 为投掷 i 个骰子得到和为 j 的方式数量。

2.状态转移方程：对于每个骰子和每个可能的投掷数量，我们都有 dp[i][j] = dp[i-1][j-1] + dp[i-1][j-2] + ... + dp[i-1][j-6]（如果 j 大于等于当前骰子的面值）。这表示投掷第 i 个骰子时，我们可以得到从 1 到 6 的任何值，并将其加到前面的 i-1 个骰子的和中。

3.初始化：我们初始化 dp[0][0] = 1，表示投掷 0 个骰子得到和为 0 的方式只有一种（即不投掷任何骰子）。

4.计算最优解：我们填充整个 dp 表，直到找到 dp[10][x]，其中 x 是我们想要找到的和。

5.结果：dp[10][x] 就是我们想要的结果，即投掷 10 个骰子得到和为 x 的方式数量。

总结：代码最后一行可以设置例子进行测试。如当 X=21 时，方式有 147940 种。

3.2 设置循环让 X 分别取 10-60，运行 3.1 的函数将得到的值用 list 储存，用 max 求出 list 中最大的值，索引最大值在 list 中的位置，+10 即为此时 X 的值。

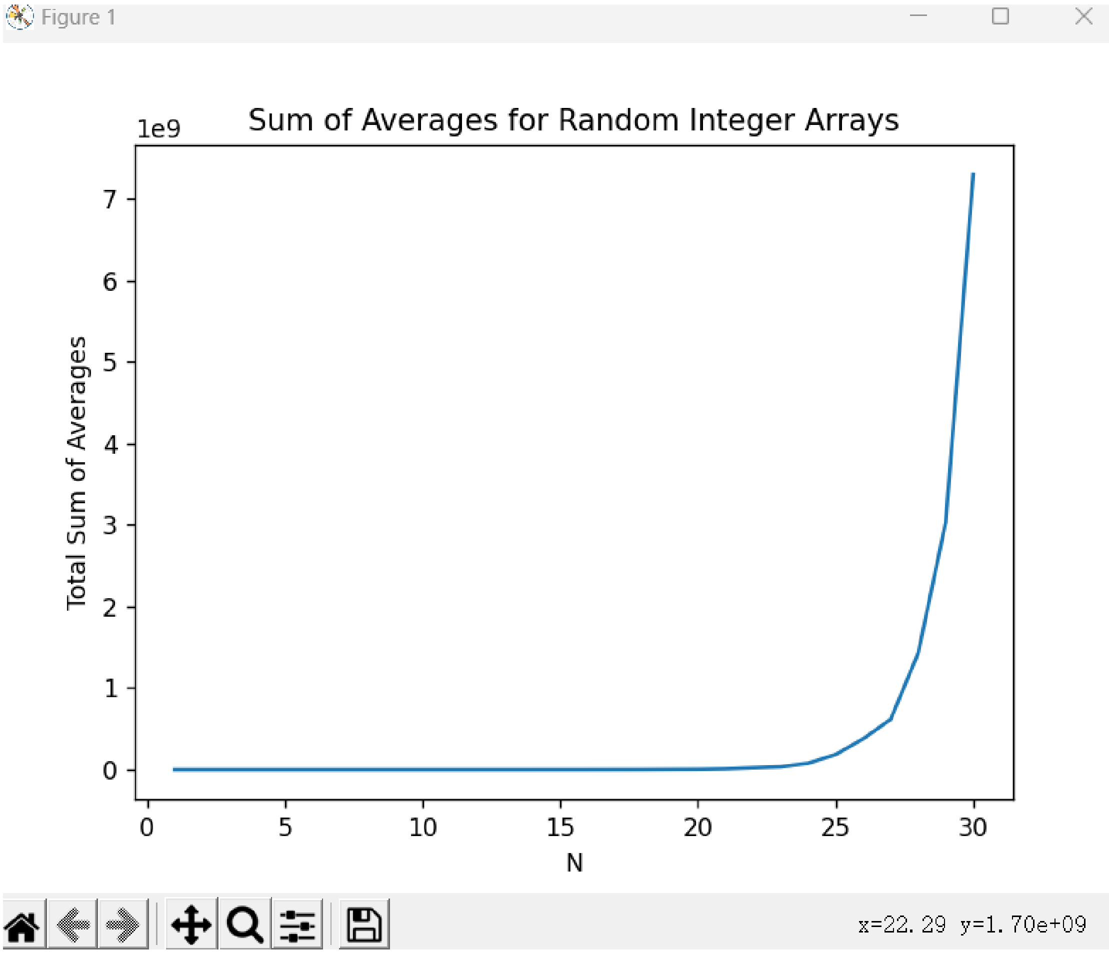
Task 4

4.1 使用 random.randint 从 0-10 中取 N 个整数

4.2 使用 itertools. combinations 取 list 的所有子集（a=list，b=长度），利用 1-list+1 的长度控制生成子集的长度，将所有的子集取平均（÷自身的长度）求和

总结：[1,2,3]均值为 14.0。

4.3 将 N 从 1-100 分别取值，利用 list 将数据储存下来，使用 matplotlib.pyplot 库进行画图。!!! 数据量过大需要很长时间才能得到结果，下图为 N 从 1-30 的结果



Task 5

5.1 使用 numpy 库，np.random.randint(low, high=None, size=None)生成（0，2，size=（N,M））。利用[0,0]和[N-1][M-1]索引赋值为 1

5.2 使用 DFS 深度优先搜索（通过递归或使用栈来探索每一个可能的路径。在每一步，算法会选择一个方向（向右或向下），继续深入探索，直到达到终点或遇到障碍。）（参考 CSDN）

我们需要设置一个边界与终点，然后进行递归计算，将他们加上上一行或上一列（走过来的路）的值，最后就可以得到总路径数。（可以想象有三种情况，若两个位置都为 0 则该点为 0 对应无通路；若上或左为 0 则该点位 1 对应有一条路径；若上和左都为 1 则该点位 2 对应有两条路径）

示例：

```
[[1 0 0 0 1 1 1]
 [1 1 1 0 1 1 0]
 [1 1 1 1 1 1 1]
 [1 0 0 0 0 1 1]
 [1 0 0 0 0 1 1]
 [0 0 0 1 1 0 1]]
Out[27]: 9
```

```
[[1 1 1 1 1]
 [1 1 0 1 1]
 [1 0 1 1 1]
 [1 0 1 0 1]]
Out[86]: 3
```

5.3 设置为 10 行，8 列，循环 1000 次，将得到的值相加取平均。（由于试验次数过小因此概率不稳定）