PS1

Assignment 01 Report

Name: XU JUNPENG (许俊鹏) SID: 12532726

1. Flowchart

Method

The function `Print_values` was implemented according to the conditional logic specified in the flowchart. A helper function, `purple_operator(x, y, z)`, was defined to execute the terminal operation `x + y - 10*z`.

A key implementation detail, based on class instruction。the branch corresponding to `a > b` being False (a <= b) and `b > c` being True was coded to return `None` immediately. This differs from the flowchart diagram, which shows this path leading to the `[c, b, a]` operation.

Result

As required by the assignment, the function was called with a=5, b=15, c=10.

1. The first condition `a > b` (5 > 15) is False.
2. The program enters the `else` block and evaluates the second condition `b > c` (15 > 10), which is True.
3. Following the specific implementation logic mentioned above, the function immediately returns `None`.

The final output is: `Print_values(5, 15, 10): None`

2. Continuous Ceiling Function

Method

The recursive function `F(x) = F(ceil(x/3)) + 2x` with the base case `F(1) = 1,` was implemented. To optimize performance and prevent re-computation of identical subproblems, **memoization** was employed. A global dictionary `memo` was used to store

the results of F(x) once computed. The base case F(1) = 1 was used to initialize this dictionary.

Result

The function was tested using the list `[1, 2, 3, 5, 7, 10]`.

The corresponding outputs are: `[1, 5, 7, 15, 21, 33]`

3. Dice Rolling

Method

3.1 Find_number_of_ways

A function `get_dice_table(n_dice, n_faces)` was created to build a 2D DP table, `dp[i][j]`, which stores the number of ways to obtain a sum j using exactly `i` dice.

1. **Base Case:** `dp[0][0] = 1` (There is one way to get a sum of 0 with 0 dice).
2. **State Transition:** The number of ways to get sum `j` with `i` dice is the sum of ways to get `j-k` with `i-1` dice, for all possible face values $k \in [1,6]$.

   The formula is:

$$dp[i][j] = \sum_{k=1}^{6} dp[i-1][j-k]$$

3. The `Find_number_of_ways(x)` function simply returns the pre-computed value `dice_dp_table[10][x]`.

3.2 Maximum Number of Ways

To find the sum x with the maximum number of ways, the slice of the DP table corresponding to 10 dice (`dice_dp_table[10]`) was examined for all possible sums from 10 to 60. The `np.max` and `np.argmax` functions were used on this slice (`Number_of_ways`).

Result

The analysis of the DP table for 10 dice yielded the following results:

`The sum x with the maximum number of ways = 35`

`Maximum number of ways = 4395456`

4. Dynamic Programming (Subset Averages)

Method

4.1 Random_integer

The function `Random_integer(N)` was implemented using `np.random.randint(1, 11, size=N)` to generate an array of size N with random integers between 1 and 10 (inclusive).

4.2 Sum_averages

A brute-force calculation of all $2^N - 1$ subset averages is computationally infeasible for ***N=100***. Therefore, a highly efficient mathematical formula was derived and implemented. The sum of all subset averages is equivalent to the total sum of all elements multiplied by a factor related to ***N***. The formula is:

$$Total = \left(\sum_{i=0}^{N-1} A[i] \right) \times \frac{2^N - 1}{N}$$

The `Sum_averages` function implements this formula directly.

4.3 Plotting and Analysis

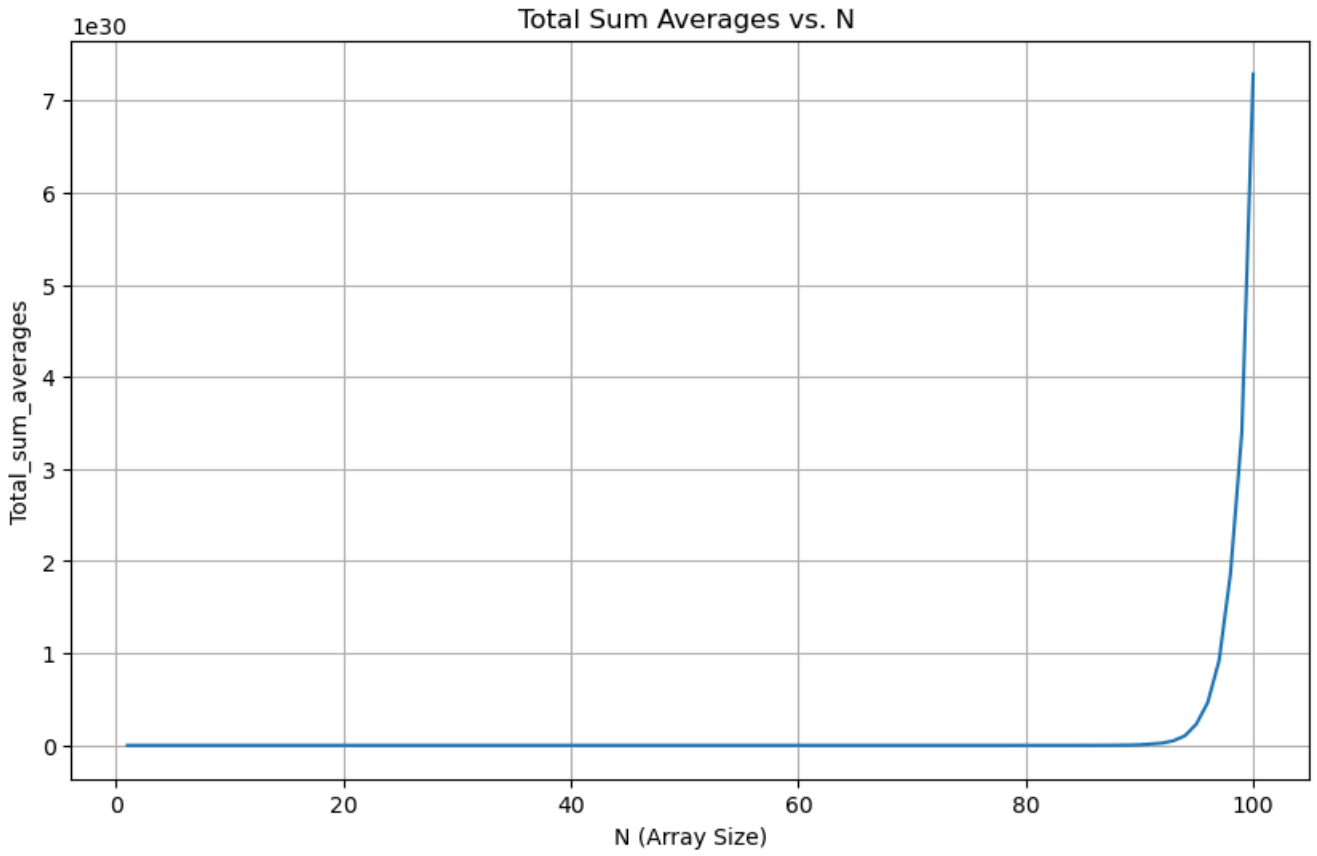
A loop was executed for **N** from 1 to 100. In each iteration, a random array was generated, and Sum_averages was computed. The results were stored in Total_sum_averages and plotted.

Result

- The function was validated with the example `[1, 2, 3]`, which correctly produced the result `14.0`.
- **Plot Analysis:**

  Fig.1 | Total Sum Averages vs. N. The generated plot of `Total Sum Averages vs. N` clearly shows exponential growth. The value of the sum grows slowly for small N but

increases dramatically as ***N*** grows, reaching values on the order of $\mathbf{10^{29}}$ by ***N=100***. This behavior is consistent with the $O(2^N)$ term in the mathematical formula.



5. Path Counting

Method

5.1 create_matrix

The `create_matrix(N, M)` function was implemented to generate an $N \times M$ grid of random 0s and 1s. It specifically sets the top-left `mat[0, 0]` and bottom-right `mat[N-1, M-1]` cells to 1, as required.

5.2 count_path

The `count_path` function uses dynamic programming to find the number of valid paths. A `paths` grid of the same size is used to store the number of ways to reach cell `(i,j)`.

- **Base Case:** `paths[0, 0]` is set to 1 if `matrix[0, 0]` is 1, otherwise 0.

- **First Row/Column:** The first row and column are filled. A cell `paths[i, 0]` receives the value from `paths[i-1, 0]` only if `matrix[i, 0]` is 1 (not a blockage); otherwise, it's 0. The same logic applies to the first column `paths[0, j]`.
- State Transition: For any other cell `(i, j)`, if it is a blockage `(matrix[i, j] == 0)`, `paths[i, j] = 0`. If it is passable (`matrix[i, j] == 1`), the number of paths is the sum of paths from the cell above and the cell to the left: `paths[i, j] = paths[i-1, j] + paths[i, j-1]`.

The final answer is the value in the bottom-right corner, `paths[N-1, M-1]`。

5.3 Simulation

A simulation was run 1000 times for a grid of size N=10, M=8. For each run, a new random matrix was generated and `count_path` was called. The mean of these 1000 results was calculated.

Result

The average number of paths found over the 1000 simulation runs is:

`N=10, M=8, 随机运行 1000 次,路径总数的平均值: 0.221`.

Collaboration

- I discussed Problem 1 (Flowchart) with my classmate Nuowen Zhou (周诺文). We debated the behavior when **a > b? is False** and subsequently **b > c? is True**:
  - **Zhou’ s view:** proceed to the next test **a > c?**.
  - **My view:** directly compute the purple-operator result on the ordered triple **(c, a, b)**.

During the following class, Prof. Zhu clarified that **b > c? = True** leads to termination with no output. We adopted the instructor’ s guidance accordingly to complete this assignment.

- **Concern.** By standard flowchart conventions, termination should be indicated with a terminal (ellipse) symbol. In the provided diagram, the arrow from **b > c? = True** does not point to a terminal symbol, which I believe is a diagrammatic inconsistency.
- Apart from the discussion above, I also used ChatGPT to help me translate this report from Chinese to English.