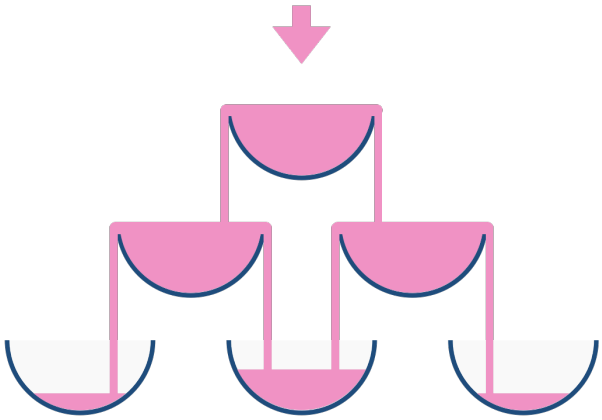
Champagne Tower

We stack glasses in a pyramid, where the first row has 1 glass, the second row has 2 glasses, and so on until the 100th row. Each glass holds one cup (250ml) of champagne.

Then, some champagne is poured in the first glass at the top. When the top most glass is full, any excess liquid poured will fall equally to the glass immediately to the left and right of it. When those glasses become full, any excess champagne will fall equally to the left and right of those glasses, and so on. (A glass at the bottom row has it's excess champagne fall on the floor.)

For example, after one cup of champagne is poured, the top most glass is full. After two cups of champagne are poured, the two glasses on the second row are half full. After three cups of champagne are poured, those two cups become full - there are 3 full glasses total now. After four cups of champagne are poured, the third row has the middle glass half full, and the two outside glasses are a quarter full, as pictured below.



Now after pouring some non-negative integer cups of champagne, return how full the j-th glass in the i-th row is (both i and j are 0 indexed.)

Example 1:
Input: poured = 1, query_glass = 1, query_row = 1
Output: 0.0
Explanation: We poured 1 cup of champange to the top glass of the tower (which is indexed as (0, 0)). There will be no excess liquid so all the glasses under the top glass will remain empty.

Example 2:
Input: poured = 2, query_glass = 1, query_row = 1
Output: 0.5
Explanation: We poured 2 cups of champange to the top glass of the tower (which is indexed as (0, 0)). There is one cup of excess liquid. The glass indexed as (1, 0) and the glass indexed as (1, 1) will share the excess liquid equally, and each will get half cup of champange.

Note:

- `poured` will be in the range of `[0, 10 ^ 9]`.
- `query_glass` and `query_row` will be in the range of `[0, 99]`.

Solution 1

We use a table to record the result.
Simple idea:
If the glass >=1, we should split the diff (glass - 1) into next level.

```
double champagneTower(int poured, int query_row, int query_glass) {
        double result[101][101] = {0.0};
        result[0][0] = poured;
        for (int i = 0; i < 100; i++) {
            for (int j = 0; j <= i; j++) {
                if (result[i][j] >= 1) {
                    result[i + 1][j] += (result[i][j] - 1) / 2.0;
                    result[i + 1][j + 1] += (result[i][j] - 1) / 2.0;
                    result[i][j] = 1;
                }
            }
        }
        return result[query_row][query_glass];
    }
```

written by suilan0602 original link here

Solution 2

C++:

```
double champagneTower(int poured, int query_row, int query_glass) {
    vector<double> dp(101, 0); dp[0] = poured;
    for(int row=1; row<=query_row; row++)
        for(int i=row; i>=0; i--)
            dp[i+1] += dp[i] = max(0.0, (dp[i]-1)/2);
    return min(dp[query_glass], 1.0);
}
```

Java :

```
public double champagneTower(int poured, int query_row, int query_glass) {
    double[] dp = new double[101]; dp[0] = poured;
    for(int row=1; row<=query_row; row++)
        for(int i=row; i>=0; i--)
            dp[i+1] += dp[i] = Math.max(0.0, (dp[i]-1)/2);
    return Math.min(dp[query_glass], 1.0);
}
```

written by DDev original link here

Solution 3

```
class Solution:
    def champagneTower(self, poured, query_row, query_glass):
        # DP solution
        res = [[0.0 for _ in range(i)] for i in range(1, query_row + 2)]
        res[0][0] = poured

        for i in range(query_row):
            for j in range(len(res[i])):
                if res[i][j] > 1 :
                    res[i+1][j] += (res[i][j] - 1) / 2.0
                    res[i+1][j+1] += (res[i][j] - 1) / 2.0

        return res[query_row][query_glass] if res[query_row][query_glass] <= 1 else
1
```

written by pl04351820 original link here

From Leetcoder.