Dynamic Programming | Set 30 (Dice Throw)

Given n dice each with m faces, numbered from 1 to m, find the number of ways to get sum X. X is the summation of values on each face when all the dice are thrown.

The **Naive approach** is to find all the possible combinations of values from n dice and keep on counting the results that sum to X.

This problem can be efficiently solved using **Dynamic Programming (DP)**.

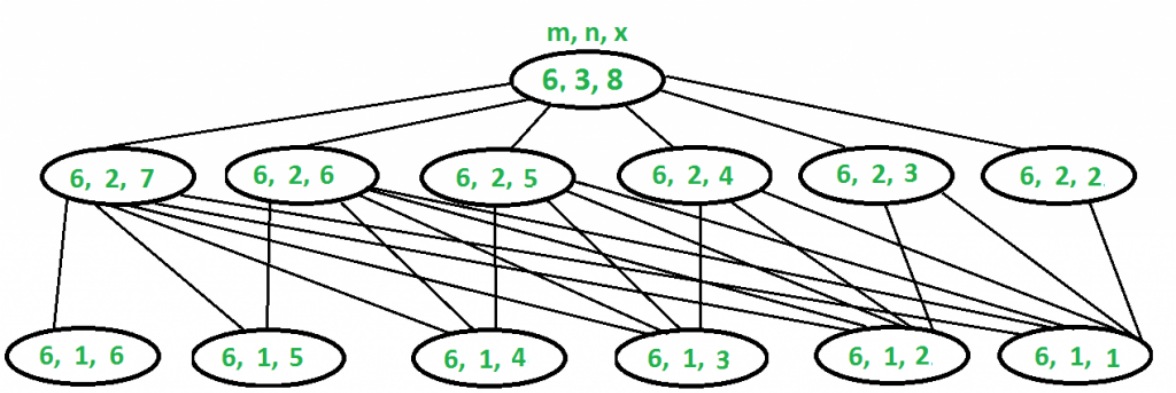
```
Let the function to find X from n dice is: Sum(m, n, X)
The function can be represented as:
Sum(m, n, X) = Finding Sum (X - 1) from (n - 1) dice plus 1 from nth dice
               + Finding Sum (X - 2) from (n - 1) dice plus 2 from nth dice
               + Finding Sum (X - 3) from (n - 1) dice plus 3 from nth dice
```

```
                    ................................................
                    ................................................
                    ................................................
                + Finding Sum (X - m) from (n - 1) dice plus m from nth dice

So we can recursively write Sum(m, n, x) as following
Sum(m, n, X) = Sum(m, n - 1, X - 1) +
               Sum(m, n - 1, X - 2) +
               ..................... +
               Sum(m, n - 1, X - m)
```

Why DP approach?

The above problem exhibits overlapping subproblems. See the below diagram. Also, see this recursive implementation. Let there be 3 dice, each with 6 faces and we need to find the number of ways to get sum 8:




```
Sum(6, 3, 8) = Sum(6, 2, 7) + Sum(6, 2, 6) + Sum(6, 2, 5) +
               Sum(6, 2, 4) + Sum(6, 2, 3) + Sum(6, 2, 2)

To evaluate Sum(6, 3, 8), we need to evaluate Sum(6, 2, 7) which can
recursively written as following:
Sum(6, 2, 7) = Sum(6, 1, 6) + Sum(6, 1, 5) + Sum(6, 1, 4) +
               Sum(6, 1, 3) + Sum(6, 1, 2) + Sum(6, 1, 1)

We also need to evaluate Sum(6, 2, 6) which can recursively written
as following:
Sum(6, 2, 6) = Sum(6, 1, 5) + Sum(6, 1, 4) + Sum(6, 1, 3) +
               Sum(6, 1, 2) + Sum(6, 1, 1)
.........................................
.........................................
Sum(6, 2, 2) = Sum(6, 1, 1)
```

Please take a closer look at the above recursion. The sub-problems in RED are solved first time and sub-problems in BLUE are solved again (exhibit overlapping sub-problems). Hence, storing the results of the solved sub-problems saves time.

Following is C++ implementation of Dynamic Programming approach.

```
// C++ program to find number of ways to get sum 'x' with 'n'
// dice where every dice has 'm' faces
#include <iostream>
#include <string.h>
using namespace std;
```

```
// The main function that returns number of ways to get sum 'x'
// with 'n' dice and 'm' with m faces.
int findWays(int m, int n, int x)
{
    // Create a table to store results of subproblems.  One extra
    // row and column are used for simpilicity (Number of dice
    // is directly used as row index and sum is directly used
    // as column index).  The entries in 0th row and 0th column
    // are never used.
    int table[n + 1][x + 1];
    memset(table, 0, sizeof(table)); // Initialize all entries as 0

    // Table entries for only one dice
    for (int j = 1; j <= m && j <= x; j++)
        table[1][j] = 1;

    // Fill rest of the entries in table using recursive relation
    // i: number of dice, j: sum
    for (int i = 2; i <= n; i++)
        for (int j = 1; j <= x; j++)
            for (int k = 1; k <= m && k < j; k++)
                table[i][j] += table[i-1][j-k];

    /* Uncomment these lines to see content of table
    for (int i = 0; i <= n; i++)
    {
      for (int j = 0; j <= x; j++)
        cout << table[i][j] << " ";
      cout << endl;
    } */
    return table[n][x];
}

// Driver program to test above functions
int main()
{
    cout << findWays(4, 2, 1) << endl;
    cout << findWays(2, 2, 3) << endl;
    cout << findWays(6, 3, 8) << endl;
    cout << findWays(4, 2, 5) << endl;
    cout << findWays(4, 3, 5) << endl;

    return 0;
}
```

Output:

```
0
2
21
4
6
```

Time Complexity: O(m * n * x) where m is number of faces, n is number of dice and x is given sum.

We can add following two conditions at the beginning of findWays() to improve performance of program for extreme cases (x is too high or x is too low)

```
// When x is so high that sum can not go beyond x even when we
// get maximum value in every dice throw.
if (m*n <= x)
    return (m*n == x);

// When x is too low
if (n >= x)
    return (n == x);
```

With above conditions added, time complexity becomes O(1) when x >= m*n or when x <= n.

Exercise:
Extend the above algorithm to find the probability to get Sum > X.

This article is compiled by Aashish Barnwal. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Dynamic Programming

Writing code in comment? Please use **ideone.com** and share the link here.

Shubham Sharma • a month ago
Number of ways - Subset problem with choosing the items multiple times
• Reply • Share ›

deeksha jain • a month ago

Within the innermost for loop dont we have to check for conditions when j-k is less than zero and handle it separately?

```
for (int i = 2; i <= n; i++)
for (int j = 1; j <= x; j++)
for (int k = 1; k <= m && k < j; k++)
{
if ((j-k) < 0)
value = 0;
else
value = table[i-1][j-k];

table[i][j] += value;
value = 0;
}
```

• Reply • Share ›

Aditya Goel • 3 months ago

Recursive implementation(Memoization needed)-

http://ideone.com/50BSVY

• Reply • Share ›

nlogn • 3 months ago

is this the right assumption that loop k is not necessary?

```
for (int i = 2; i <= n; i++)
for (int j = i; j <= x; j++)
table[i][j] = table[i][j-1]+table[i-1][j-1];
```

Code above produce same result as the code in this posting. link : http://ideone.com/5erjyh

• Reply • Share ›

Aditya Sharma • 8 months ago

time complexity when using dynamic programming solution can be further reduced to O (n*x). Note that table[i][j] = table[i-1][j] + table[i-1][j-1] .(except boundary cases.). Hence every element table is traversed once and O(1) time for addition.

• Reply • Share ›

Anonymous ➚ Aditya Sharma • 4 months ago

top down implementaion in java - http://ideone.com/1pXwC8

• Reply • Share ›

sneha • 9 months ago

canr understand this
table[i][j]+=table[i-1][j-k];

• Reply • Share ›

dev21 • 9 months ago

Solution top-down approach instead of bottom-up approach

http://ideone.com/Oz9Qzc

• Reply • Share ›

falak dev21 • 9 months ago

hey dev21 nice work

• Reply • Share ›

AlienOnEarth • a year ago

Recursive method in C:

```
void sum(int m, int n, int x, int *total)

{

int cur_sum = 0;

// base case - 1

if(n==0 && x== 0)

{

*total = *total + 1;

return;

}

// base case 2

if((n==0 && x!=0) || (n!=0 && x==0))

return;

for(int i=1;i<=m;i++)

sum(m,n-1,x-i,total);

}
```

• Reply • Share ›

Arnab sen • a year ago

```
public static void main(String[] args){

int m=4,n=3,x=5;

int[][] count=new int[x+1][n+1];

for(int[] row:count)

Arrays.fill(row,0);

for(int i=1;i<=x;i++){

if(i<=m)

count[i][1]=1;

}

for(int l=1;l<=n;l++)

{
```

see more

1 • Reply • Share ›

prashant jha • a year ago

```
void fun(int n,int m,int s,int &cnt)

{

if(n==1)

{

if((s<=m)&&(s>=1))

cnt++;

return;

}

for(int k=1;k<=m;k++)

{

fun(n-1,m,s-k,cnt);
```

see more

prashant jha • a year ago

```
void fun(int n,int m,int s,int &cnt)

{

        if(n==1)

          {

                if((s<=m)&&(s>=1))

                 cnt++;

                return;

          }
```


prashant jha • a year ago

```
/*

#include<iostream>

using namespace std;

int fun(int n,int m,int s)

{

if((!n)&&(s))

return 0;

if((!n)&&(!s))

return 1;

for(int k=1;k<=m;k++)

{
```

see more

• Reply • Share ›

Abhiram • a year ago

theres actually a formula for this
http://equax.blogspot.in/2013/...

• Reply • Share ›

Nikhil Gulumkar • a year ago

My answer for 6,3,8 is coming 10

Cant figure out the problem

The code is in Java

```
package DiceThrow;

public class mainClass {

public static void main(String[] args) {

int x=8,n=3,m=6;

System.out.println(findNum(x,n,m));

}

private static int findNum(int x, int n, int m) {

int table[][]=new int[n+1][x+1];
```

see more

• Reply • Share ›

Vinodhini • 2 years ago

How could we extend it to print all the combinations?
lets say for no of dice =2 of face 6, we need all combination that sums to 6
then I need to print all combinations.
(1,5) (2,4) (3,3) (4,2) (5,1)

• Reply • Share ›

Garg • 2 years ago

We can do better! Time complexity if this sol is O(n*min(X-n+1))
using f(m,n,X) = f(m,n,X-1) + f(m,n-1,X-1) hence calculating any f() in O(1) time.
Here is the code!

```
#include<stdio.h>
```

```
#include<string.h>
#define min(a,b) ((a<b)? (a) : (b))

int findWays(int m, int n, int X){
        if(X<n || X>n*m)                return 0;
        int dp[min(X-n+1,m)],i,j;
        int len = sizeof(dp)/sizeof(dp[0]);
        for(i=0; i<len; i++)
                dp[i] = 1;
        for(i=1; i<n; i++){
                for(j=1; j<len; j++)
                        dp[j] += dp[j-1];
        }
```

see more

• Reply • Share ›

the_c0der • 2 years ago

```
 hey i wanted to ask some questions :
1. what if the value of face i is not i ..i.e all faces have           different values(uns

2.should we not find the table for all possible sums in above post i.e m*n in one time only
```

• Reply • Share ›

Born Actor • 2 years ago

```
 #include <iostream>
#include<string>
#include<sstream>
#include<iomanip>
# include <stdio.h>
# include <math.h>
#include <vector>
#include <stdlib.h>
using namespace std;
int n;
int a[50][50];
int m;
int s;
int function(int number, int sum);
int each_dice[50];
void print();
int main()
```

```
{
```

see more

• Reply • Share ›

Born Actor Born Actor • 2 years ago

The above mentioned code gives one of the feasible solutions and not the number of ways of finding the solution. It follows Back-tracking.
I am not able to estimate the complexity of the algorithm.
Are all back-tracking algorithms exponential in the worst case?

• Reply • Share ›

C Programming Techies • 2 years ago

alternate implementation :

```
/* Data Structure Includes */.
#include <vector>
#include <list>
#include <map>
#include <set>
#include <queue>
#include <deque>
#include <bitset>
#include <stack>

/*other Includes */.
#include <algorithm>
#include <functional>
#include <numeric>
#include <utility>
#include <sstream>
```

see more

• Reply • Share ›

Crunchbang • 2 years ago

Will there be any difference in performance if I used std::map instead of a 2-D array ?

• Reply • Share ›

binary001 • 2 years ago

friends the below solution(in my comment) is wrong as it is only valid
if
we have a zero face on dice.
or

we consider n or less than n dice.
which are not the conditions in the question.

• Reply • Share ›

chester binary001 • 2 years ago

I agree with you and additionally this actually gets the number where each dice is distinguished from one another i.e. there is repetition in counting if all the dices are the same.

```
/* Paste your code here (You may delete these lines if not writing code) */
```

• Reply • Share ›

Anon_001 • 2 years ago

```
int dp[10000][1000];
int fun(int n,int m,int sum,int cur_sum,int cur_coun)
{
        if(dp[cur_coun][cur_sum]!=-1)
        return dp[cur_coun][cur_sum];
        if(cur_coun==n)
        {
                if(cur_sum==sum)
                {
                        return dp[cur_coun][cur_sum]=1;
                }
                else
                return dp[cur_coun][cur_sum]=0;
        }
        int ans=0;
        for(int i=1;i<=m;i++)
```

see more

• Reply • Share ›

supora • 2 years ago

For me answer for (4,3,5) is coming as 4.

```
 #include <vector>
#include <list>
#include <map>
#include <set>
#include <deque>
#include <stack>
```

```
#include <bitset>
#include <algorithm>
#include <functional>
#include <numeric>
#include <utility>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cstdio>
#include <cmath>
```

see more

• Reply • Share ›

binary001 • 2 years ago

can be solved with memory O(m*n) .... m*n is worst case for x

```
int dp [x];
memset(dp,0,sizeof(dp))
// no of dice
for(int i=1;i<=n;i++){
    for(int j=m*i;j>=0;j--){  // from max sum till now to zero
       for(int k=1;k<=m;k++){  //for each possible value of dice
    if(j+k > x )continue;
         dp[j+k]+=dp[j];
      }
}
```

• Reply • Share ›

amarkrdubedy ➚ binary001 • 2 years ago

I don't believe this solution works. You would need two dp[x] arrays, one to store previous results and other to store present results.

• Reply • Share ›

Anon_001 ➚ binary001 • 2 years ago

Answer for (4,3,5) should be 4 or 6 ???

• Reply • Share ›

yifenliu ➚ Anon_001 • 2 years ago

I have the same question. My code yields 4 for (4, 3, 5)

```
[sourcecode language="java"]
/* Paste your code here (You may delete these lines if not writing code) */
int[][] dp = new int[x + 1][n + 1];
// dp[i][j] represents how many ways to get sum i with j dice with m facec.
int i, j, k;
// init dp. use only one dice.
for (i = 1; i <= Math.min(m, x); ++i) {
dp[i][1] = 1;
}

//
for (i = 2; i <= n; ++i) { // Using i dice
for (j = 1; j <= x; ++j) { // To get sum j
for (k = 1; k < j; ++k) { // And index of cur dice is k, k within[1, k - 1], inclusive
inclusive.
dp[j][i] += dp[j - k][i - 1];
}
}
}
```

▲ | ▼ • Reply • Share ›

kartik ➚ yifenliu • 2 years ago

For (4, 3, 5), answer seems to be 6.

1, 1, 3
3, 1, 1
1, 3, 1
2, 2, 1
2, 1, 2
1, 2, 2

▲ | ▼ • Reply • Share ›

Anon_001 ➚ kartik • 2 years ago

Ohh ...n & m are passed at different positions.

▲ | ▼ • Reply • Share ›

kartik ➚ binary001 • 2 years ago

Thanks for the comment. Could you please provide more details, like how does this work, how is time complexity O(m*n)?

▲ | ▼ • Reply • Share ›

kk ➚ kartik • 2 years ago

memory requirement is O(m*n)
time complexity is still O(m*n*x)

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 2 hours ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 2 hours ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 2 hours ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago