Solution-Combinatorics-V2

April 3, 2025

1 Solution Combinatorics - V2

Arthur Ryman, last updated 2025-04-03

1.1 Introduction

This notebook shows how to compute the total number of combinations of the Instant Insanity puzzle. The branch of mathematics that focuses on how to count combinations is called *Combinatorics*. The number of combinations for any puzzle is very interesting since it indicates how hard it is to solve the puzzle.

1.1.1 Change History

- Initial version, 2025-01-29 : focused on explaining the numbers 41,472 and 82,944 which are cited as the number of combinations
- V2, 2025-02-06: starts with the idea of a random arrangement of cubes, computes the number of such arrangements, and shows how the operation of reordering an arrangement sends solutions to solutions

1.2 41,472 or 82,944?

1.2.1 41,472

The 1947 Eureka paper The Coloured Cubes Problem by F. de Carteblanche which gives the elegant graph theory solution to the problem states that if you pick a random arrangement of the cubes your chance of it being a solution is 1 in 41,472. At that time, the puzzle was called the *Tantalizer*. Carteblanche was the pseudonym of a group of four Cambridge mathematics students which included the future eminent graph theorist Bill Tutte.

1.2.2 82,944

The 2008 Gresham College lecture The Four Cubes Problem by Robin Wilson, Gresham Professor of Geometry, gives the number as 82,944.

The 2018 PBS Infinite Series video Instant Insanity Puzzle by Tai-Danea Bradley also gives the number as 82,944.

The 2024 Winning Solutions Instant Insanity package similarly advertises 82,944 combinations.

1.2.3 Who's Correct?

Let's apply some combinatorics and compute the answer for ourselves.

1.3 Stacks and Solutions

First we need to precisely define what we mean by a combination of the cubes. The 1947 Carteblanche paper describes the goal of the puzzle as follows:

> The problem is to stack these cubes in a vertical pile (thus forming a square prism) in such a way that each of the four vertical faces of this pile contains all four colours.

We'll call any vertical pile of the four cubes a *stack* and any stack that solves the problem a *solution*.

Let the number of stacks be N and the number of solutions be S. If we pick a stack at random, the probability that it is a solution is S/N. Carteblanche makes the following claim:

> The chance of obtaining the solution by a random arrangement of the cubes is only 1/41472.

In terms of N and S, Carteblanche's claim is:

$$\frac{S}{N} = \frac{1}{41472}$$

Let's compute N and S.

1.4 Counting the Number of Stacks

To count the number of stacks, consider the following procedure. Put the four cubes in a bag, shake it, draw the cubes out one at a time without looking, and stack them. This procedure will eventually produce all possible stacks.

Animation #1: random draws

Each stack consists of the cubes in some order, with each cube having some orientation. We need to count the number of cube orderings and the number of orientations that each cube can have.

1.4.1 Counting the Number of Cube Orderings

One of the most basic theorems of combinatorics is that if you have a set of n distinct objects then you can order them in

$$n \times (n-1) \times \cdots \times 2 \times 1$$

ways. This quantity is denoted $n!$ which is spoken as n *factorial*. To see this observe that you have n ways to choose the first object. Now there are $n-1$ objects remaining so for each of the n first objects there are $n-1$ ways to choose the second object. Therefore there are $n \times (n-1)$ orderings of two objects. Continuing this reasoning, we get that there are $n!$ orderings n objects.

Let's compute 4!.

[1]:
```
4 * 3 * 2 * 1
```

[1]:
```
24
```

Python has a built-in `factorial` function.

[2]:
```
from math import factorial

number_of_cube_orderings = factorial(4)

print(number_of_cube_orderings)
```

```
24
```

1.4.2 Counting the Number of Cube Orientations

How many ways can a single cube be oriented? We can specify an orientation by saying which face is on top and which face is in front. Let's count the number of orientations.

We'll use another basic theorem of combinatorics. If we have two sets of things, say X containing n things and Y containing m things then the number of ways we can pair things from X with things from Y is $n \times m$.

A cube has 6 faces. Therefore, there are 6 ways to pick the top face.

Animation #2: 6 choices for top face

[3]:
```
number_of_top_face_choices = 6
```

Having picked the top face, we can spin the cube one quarter turn about its vertical axis to pick the front face. We can do this four ways.

Animation #3: 4 choices for front face

[4]:
```
number_of_front_face_choices = 4
```

Therefore, the total number of cube orientations is 6×4.

[5]:
```
number_of_cube_orientations = number_of_top_face_choices *␣
↪number_of_front_face_choices

print(number_of_cube_orientations)
```

```
24
```

1.4.3 Counting the Number of Stacks

A stack is specified by the order of the four cubes and by the orientation of each of them. There are 24 orderings of the cubes. Each cube has 24 orientations. Apply the basic combination rule again to get the total number of stacks. Multiply the number of cube orderings and the number of cube orientations multiplied by itself four times which is the same as raising the number of cube orientations to the power 4.

[6]:
```
number_of_stacks = number_of_cube_orderings * number_of_cube_orientations ** 4

print(number_of_stacks)
```

```
7962624
```

We get a whopping total of $7,962,624$ stacks which is much bigger than the claimed number of combinations. Clearly, we are overcounting.

Relating this back to our variable N we have

[7]:
```
N = number_of_stacks

N
```

[7]:
```
7962624
```

1.4.4 When Are Two Stacks Equivalent?

Do we really care about the order of the cubes? If we have found one solution then we can produce another solution simply by swapping the positions of the cubes without changing their orientations.

Animation #4: changing the order of the cubes

In fact, given a solution, there are a total of 24 related orderings of the cubes that are also solutions. That seems like artificially inflating the number of solutions. Two solutions that differ simply in the order of the cubes should be considered equivalent. Therefore, let's simply pick an order for the cubes. We'll simply number the cubes from 1 to 4 and always stack them in that order

Swapping the positions of the cubes without changing their orientations is an example of a *symmetry operation* on the set of solutions since it sends solutions to solutions. A set of related symmetry operations is referred to mathematically as a *group* In general, we'll consider solutions that are related to each other by some symmetry to be equivalent.

Let O be the set of all order-changing stack symmetries, that is, operations that swap cube positions but not cube orientations. As mentioned above, the number of stacks related to each other O operations is 24. We refer to this as the *size* of the set of related symmetry operations. The set of order-changing symmetry operations evenly divides the set of all stacks into families of size 24 such that each stack within a family is related to every other stack in the family by some order-changing symmetry. Compute the number of order-changing symmetry families.

[8]:
```
size_O = 24
number_of_stacks_O = number_of_stacks // size_O

number_of_stacks_O
```

[8]:
```
331776
```

We get 331,776 which is much bigger than 82,944. Why the difference? The reason is that we should regard more arrangements as being essentially the same.

Suppose we have a solution. Then we can easily get another solution simply by rotating the stack one quarter of a turn about the long axis. Furthermore, we can do this four times before ending up with the arrangement we started from. Therefore each solution belongs of a family of four essentially equivalent solutions.

Animation #5 rotating the stack of cubes one-quarter turn about the long axis

Let Q be the set of these quarter-turn symmetry operations. Clearly, this operation does not change the order the cubes and it sends one O family to another. In fact, it groups the set the O families into families of four related families. We refer to these families of families as OQ families. Compute the number of OQ families

[9]:
```
size_Q = 4
number_of_stacks_OQ = number_of_stacks_O // size_Q

print(number_of_stacks_OQ)
```

```
82944
```

Progress! Now we understand where the number 82,944 comes from. It is the number of stacks where we consider stacks related by operations in O or Q to be equivalent.

But what about 41,472? The explanation is similar to the above. We have yet another symmetry.

Suppose we have a solution. We can obtain another solution by rotating each cube by one half turn about its vertical axis.

Animation #6: rotating each cube one-half turn perpendicular to the long axis

Let H be this half-turn operation. We can apply the H operation twice before returning to the starting arrangement. Therefore, we need to further divide the number of arrangements by 2 to get the number of essentially distinct arrangements.

[10]:
```
size_H = 2
number_of_stacks_OQH = number_of_stacks_OQ // size_H

print(number_of_stacks_OQH)
```

```
41472
```

Success! We now understand where the number 41,472 comes from. It is the number of essentially distinct stacks if we consider stacks related by O, Q, or H operations to be equivalent. Furthermore, as we will soon prove, there is exactly one OQH family that solves the problem. Carteblanche was right.

Relating this back to our variable S we have

[11]:
```
S = size_O * size_Q * size_H

print(N, S, N // S)
```

```
7962624 192 41472
```

There are around 8 million stacks but these are divided into families each consisting of 192 mutually equivalent stacks. The total number of essentially distinct stacks is $41,472$.

1.5 Brute-Force Search

We now know how many arrangements, namely 41,472, we'd have to check in order to find a solution. Given enough time and patience, we could systematically generate every possible essentially distinct arrangement and check if it was a solution. Mathematicians call this approach a *brute-force search*.

How long would a brute-force search take? Suppose a human was doing the search. As a rough estimate, suppose it takes 1 second to rotate a cube and 1 second to check if the arrangement is a solution. Let's compute the total time.

[12]:
```
seconds_per_stack = 5
total_seconds = number_of_stacks_OQH * seconds_per_stack

print(total_seconds)
```

```
207360
```

[13]:
```
seconds_per_minute = 60
total_minutes = total_seconds / seconds_per_minute

print(total_minutes)
```

```
3456.0
```

[14]:
```
minutes_per_hour = 60
total_hours = total_minutes / minutes_per_hour

print(total_hours)
```

```
57.6
```

[15]:
```
hours_per_day = 24
total_days = total_hours / hours_per_day

print(total_days)
```

```
2.4
```

Therefore, a human would find the solution if they worked nonstop for 2.4 days and made no errors. This is called the *worst case* time since it assumes that you are unlucky and have to generate all the arrangements before you find the solution.

Clearly, this puzzle is challenging which accounts for its popularity. The package states that over 20 million copies have been sold!

1.6 Next

How long would a computer take to find the solution? One of us actually tried this in 1967 using a Fortran 4 program running on their high school's IBM 1130 computer. Next, we'll write some Python code to perform the brute-force search and then optimize it for speed.