FiveThirtyEight's June 19, 2020 Riddler

Emma Knight

June 19, 2020

This week's riddler, from Dean Ballard, is about a fair division of spheres:

Question 1. *King Auric adored his most prized possession: a set of perfect spheres of solid gold. There was one of each size, with diameters of 1 centimeter, 2 centimeters, 3 centimeters, and so on. Their brilliant beauty brought joy to his heart. After many years, he felt the time had finally come to pass the golden spheres down to the next generation - his three children.*

He decided it was best to give each child precisely $1/3^{rd}$ of the total gold by weight, but he had a difficult time determining just how to do that. After some trial and error, he managed to divide his spheres into three groups of equal weight. He was further amused when he realized that his collection contained the minimum number of spheres needed for this division. How many golden spheres did King Auric have?

Extra credit: How many spheres would the king have needed to be able to divide his collection among other numbers of children: two, four, five, six or even more?

First off, notice that the volume of a sphere of diameter d is Cd^3 where C is some constant[1] that comes out in the wash and so will be ignored from here on out. This question reduces to:

Question 2. *Find the smallest n such that you can partition the set $\{1, 2, \ldots, n\}$ into 3 disjoint sets S_1, S_2, and S_3 such that* $\sum_{k \in S_1} k^3 = \sum_{k \in S_2} k^3 = \sum_{k \in S_3} k^3$.

Since the sum of all the cubes up to n is $\frac{n^2(n+1)^2}{4}$, this sum must necessarily be equal to $\frac{n^2(n+1)^2}{12}$ and n must necessarily be congruent to 0 or 2 (mod 3). I have zero insight about how to think about this problem, so I will just hit it with a computer. I wrote the following python code to compute the solutions to this problem. The auxiliary function is almost surely awfully programmed (but it does work). If you want to upscale this program, it's probably better to find some way to do tail recursion in the first function. However, I wouldn't lose too much sleep over optimizing this code as it's a special case of bin packing, an NP-complete problem.

[1] $\pi/6$

```
##This function computes all possible ways to write
##n as the sum of cubes of distinct positive integers
##which are at most a.  It outputs this as a list of
##lists, the internal lists are sets of integers in
##increasing order such that the sum of the cubes of
##these integers is n.

def f(a, n):
    results = []
    if (a == 0):
        return(results)
    if (n == (a**3)):
        results.append([a])
    if (n > (a**3)):
        for l in f(a-1, n-(a**3)):
            c = l.copy()
            c.append(a)
            results.append(c)
    for l in f(a-1, n):
        c = l.copy()
        results.append(c)
    return(results)

##This is the main loop.  If the sum of all the cubes
##up to i is divisible by three, then it lets n be
##this sum divided by three.  Then, it calls the
##previously definied function on i and n, and gets
##a list of lists.  It then looks to see if you can
##find two lists of integers such that each integer
##from 1 to i is represented at most once, and if so
##it prints out the triple of lists.  Why 31?  This
##code is slow, and thats high enough to get the first
##solution :P (originally it was higher, but then my
##computer ran out of memory).
for i in range(31):
    if (((i % 3) == 0) or ((i % 3) == 2)):
        n = (((i+1)*i)**2)//12
        possibles = f(i, n)
        length = len(possibles)
        for j in range(length):
            for k in range(j+1, length):
                accept = True
                for m in range(1, i+1):
                    if ((possibles[j].count(m) +
                         possibles[k].count(m)) > 1):
                        accept = False
                if accept:
                    print(i, n, possibles[j], possibles[k])
```

There are no solutions until $n = 23$, where there is exactly one solution: $\{2, 5, 9, 11, 14, 15, 17, 23\}$, $\{1, 4, 7, 8, 12, 16, 20, 22\}$, and $\{3, 6, 10, 13, 15, 19, 21\}$. There are no solutions for $n = 24$, and then the floodgates open, with there being approximately five bazillion solutions for anything larger that can have solutions (this is a slight exaggeration).

This is also easy to find solutions for two children: the first solution occurs for $n = 12$ and is $\{1, 2, 4, 8, 9, 12\}$ and $\{3, 5, 6, 7, 10, 11\}$. It isn't too bad to make this work for four children: the first solution occurs at $n = 24$. There are three solutions, but the first one spit out by the program is $\{1, 2, 3, 4, 14, 18, 24\}$, $\{7, 9, 21, 23\}$, $\{8, 10, 11, 16, 17, 22\}$, and $\{5, 6, 12, 13, 15, 19, 20\}$. At this point I (and my poor CPU) would advise the royal couple to ease up on their procreation habit.