ACM Summer Challenge 2020

Elementary Mathematics

Editorial

Ghoul's Dice

Author: Krunal Rank

Consider a die other than the top-most one.
As the sum of numbers on the opposite faces of a die is always **9**, the sum of numbers on visible faces is always **18**.
For the top-most die, the numbers on the sides also add up to **18**, and there is an additional number on top of the die.
Sum of visible dots should be equal to **N**.
So N can be represented as

N=18d+t , where **d=number of dice** and **t=number on top**.

So compute **d = floor(N/18)** and **t = N mod 18**.
The answer will be **YES** only if **d≥1** and **2 ≤ t≤ 7**.
This can be done in **O(1)** per query.

Maximize the operation

Author: Jitendra Jat

After every operation **N** reduces. To maximize the number of operations it is always optimal to divide N by the least possible number.
So code goes as below

while(N>1) N /= “lowest factor of N”, count++;

In every loop **N** will be **divided by at least 2**. So while the loop will run **O(logN)** time complexity and computing the lowest factor will take **sqrt(N)** . So overall complexity will become **O(logN * sqrt(N))**.

We can **precompute the lowest prime factor of every number ≤ 10^6** to reduce our complexity.
We can modify classic **Sieve of Eratosthenes** to compute the lowest prime factor.
Instead of maintaining true or false correspondence to the prime number we will maintain the lowest prime factor.
This part is left as an exercise for the reader.
Now we can answer each query in **O(logN)** time.

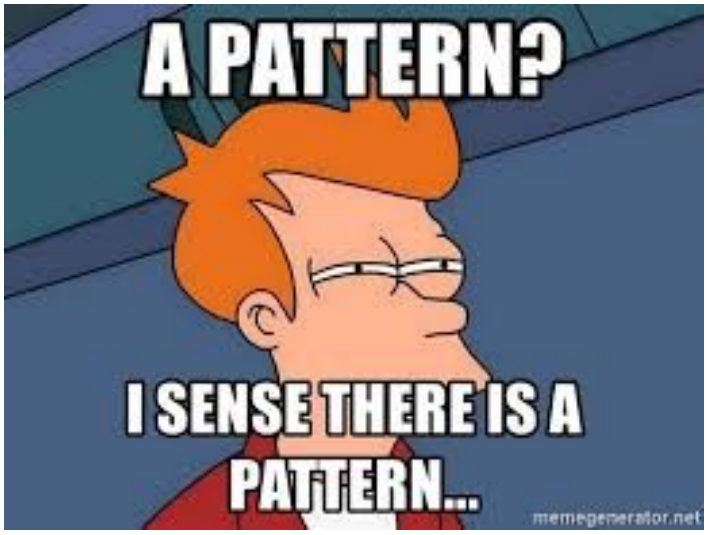
Find Pattern

Author: Jitendra Jat

It is easy to note that
for **N < 10,** $\alpha_1 + \alpha_2 + \alpha_3 + ... = 10$ never be satisfied. So print **-1**.
For **N ≥ 10**, consider below examples
N = 12 -> $2^2 * 1^8$
N = 13 -> $2^3 * 1^7$
.
.
N = 19 -> $2^9 * 1^1$
N = 20 -> 2^{10}
N = 21 -> $2^9 * 3^1$
N = 22 -> $2^8 * 3^2$
We can generalize it as below

N = ab -> $a^{10-b} * (a+1)^b$

Print required answer after taking modulo with **10^9+7**.



Gotta catch em All!

Author: Krunal Rank

The question is based on the concept of **Convex Hull**. Convex hull is the polygon that is defined in the problem statement.

For finding convex hull from a given set of points, follow the following procedure:

- **Sort the given set of points according to increasing y coordinate and if any two points have the same y coordinate, then sort them using increasing x coordinate.**
- Now, the first point in the given set of points is considered the **base point**.
- Again sort, in increasing order, the set of given points except the first point, this time, using a comparator wherein, you find the angle of that point with the line passing through the base point and parallel to the x axis. If any two points have the same angle, then sort the farthest point first.
- Make a new list of points wherein you store only the points with a unique angle formed with base point and these points must be the farthest one with that angle.
- If this new list has less than 3 points, it is impossible to form a convex hull.
- Now that you have this list ready, use a stack and push the first three points in the stack from the new list.
- Now iterate over the list from the 4^{th} point onwards and follow a certain procedure:
    - Pop the points from stack until the orientation of the point that is being iterated, the top-most point of the stack, and the penultimate point of stack is not counter-clockwise. This can be done using trigonometric formulae.
    - Push the point that is being iterated into the stack.
- The stack now contains all the points that are a part of the convex hull that is required. Store these points in a vector.

For calculating twice the area of the polygon formed by these points, use the Shoelace formula. You can refer more about it here.

For more information regarding convex hulls, you can visit the link. This link provides the same algorithm but in a different way. The process might be slightly different.

This procedure can be done in **O(NLogN)**.

The algorithm has contributed immensely to the field of computer vision. Those of you who are deeply interested must refer to it thoroughly.

Image by **Ishaan Mehta** :)