J261 - Cubic Date

Isaac Wong {WongChun1234}

2026-02-14

Table of Contents

1 The Problem

2 Main Solution: Construction

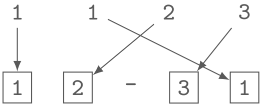
3 Alternative Solution: Brute Force

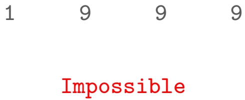
Background

Problem Idea by `mtyeung1`
Preparation by `QwertyPi`, `WongChun1234` (Thanks!)

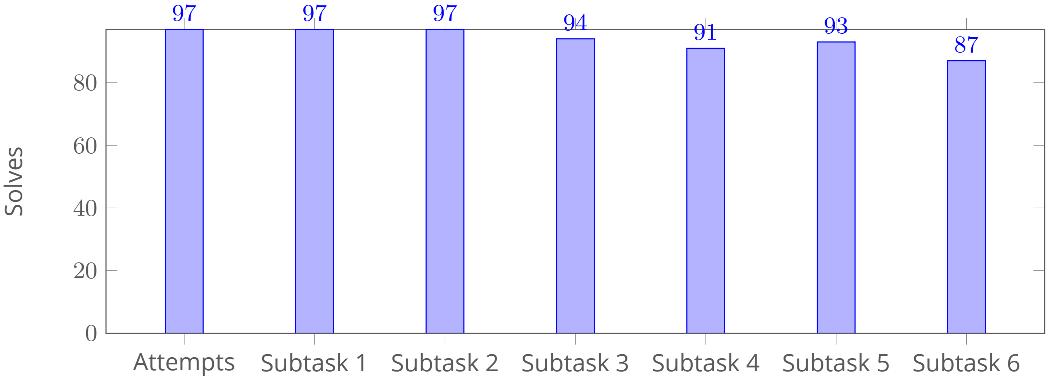
Problem Restatement

- Given 4 digits, display a valid date in Year 2026 in `MM-DD` format.
- If it is impossible to display such a date, output `No`.





Statistics



First solved by **Archso** (Guan Hui) at **4m 48s**.

Subtasks

For all cases: $0 \leq D_1 \leq D_2 \leq D_3 \leq D_4 \leq 9$.

Subtask	Points	Constraints
1	12	$D_1 = D_2 = 0$, $D_3, D_4 \neq 0$
2	19	$D_1 = D_2 = 0$
3	14	$D_1 = 0$, $D_2 = 1$, $D_3, D_4 \geq 1$
4	27	$D_1 = 0$
5	21	$D_1 = 1$, $D_2, D_3, D_4 \geq 1$
6	7	No additional constraints

Table of Contents

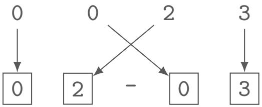
1 The Problem

2 Main Solution: Construction

3 Alternative Solution: Brute Force

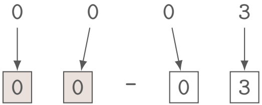
Subtask 1 (12%): $D_1 = D_2 = 0$, $D_3, D_4 \neq 0$

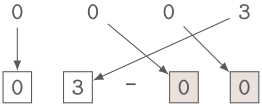
- To construct a valid date, we need
  - The month `MM` to be between `01` and `12`.
  - The day `DD` to be between `01` and the number of ways in the month ($\geq$ 28).
- For both numbers, the ten digit (must be `0`/`1`/`2`) is more restrictive than the unit digit (can be `1`-`9`).
- One possible way to construct is to fill `0`-s to the ten digit, and D_3, D_4 to the unit digit.
  - This is always a valid date!



Subtask 2 (19%): $D_1 = D_2 = 0$

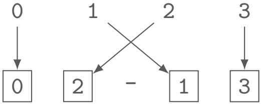
- If both D_3 and D_4 are non-zero, we can apply the solution of Subtask 1.
- What if D_3 is zero?
  - We are assigning 3 zeros to 4 different slots.
  - Therefore, either the month (`MM`) or the day (`DD`) will be assigned 2 zeros (`00`).
  - This is definitely not a valid date ⇒ Impossible!





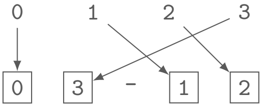
Subtask 3 (14%): $D_1 = 0$, $D_2 = 1$, $D_3, D_4 \geq 1$

- Now we only have one zero. Is it better to assign it to the month or the day?
    - Month must be between 01 and 12, so it is more restrictive.
- Assign 0 to the month's ten digit and 1 to the day's ten digit. After that, the unit digits can be assigned arbitraily.
    - It is always a valid date!



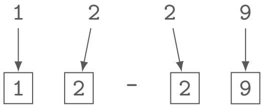
Subtask 4 (27%): $D_1 = 0$

- Assign 0 to the month's ten digit. Assign D_2 to the day's ten digit.
  - If D_2 is 0/1/2, then it is always possible.
  - If D_2 is $\geq$ 3, then the day consists of two digits ≥ 3, so it can never be a valid date.
- Can we assign the unit digit arbitraily?
  - Special case! The date 02-29 does not exist in 2026!
  - To solve this issue, we assign D_3 (the smaller one) to the day and D_4 (the larger one) to the month. 02-29 $\Rightarrow$ 09-22.



Subtask 5 (21%): $D_1 = 1$, $D_2, D_3, D_4 \geq 1$

- This time, we have no choice but to place a 1 in the month's ten digit.
- This means that **both** the month's unit digit **and** the day's ten digit must be ≤ 2.
    - If $D_3 > 2$, then we can output impossible.
- Otherwise, we can fill D_2 and D_3 into the month's unit digit and the day's ten digit, and the resulting date is always valid.



Subtask 6 (7%): No additional constraints

- Recall what we did in the earlier subtasks:
  - Subtask 4: $D_1 = 0$.
  - Subtask 5: $D_1 = 1$.
- The remaining case would be $D_1 > 1$. This implies all digits are ≥ 2.
  - This implies we can never construct a valid month (from `01` to `12`)!
  - In such case, we can always output impossible.

Noticed something weird...?

Did you notice that we never used the days 30 and 31, and the month does not matter most of the time (except February)?

- In this task, we are only aiming to get **any possible construction**, if one exists.
- It is easier to replace 30 and 31 by 03 and 13, as we do not have to worry about the unit digit after placing the ten digit (any value from 1 to 9 works!).
- This is how we usually approach constructive tasks – we prefer the cleanest construction!

Table of Contents

1 The Problem

2 Main Solution: Construction

3 Alternative Solution: Brute Force

Alternative Solution

- Note that the constraints are small, which allows us to **brute force** all possibilities of filling in the dates.
- **Solution 1**: We exhaust all permutations of D_1, D_2, D_3, D_4 to form the `MM-DD`.
  - There are $4! = 24$ such permutations.
  - For each permutation, we check whether it is a valid date.
- **Solution 2**: We exhaust all 365 days of the year (the resulting `MM-DD` pattern).
  - For each day, we check whether the digits used match D_1, D_2, D_3, D_4.
- These solutions might be more straightforward but slightly harder to implement.