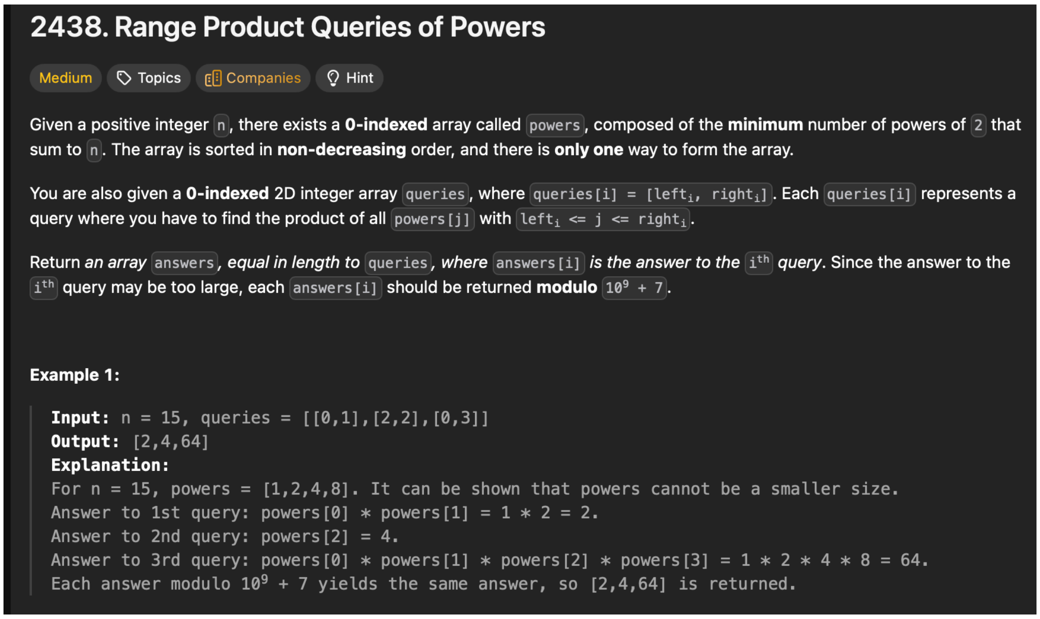


2438. Range Product Queries of Powers

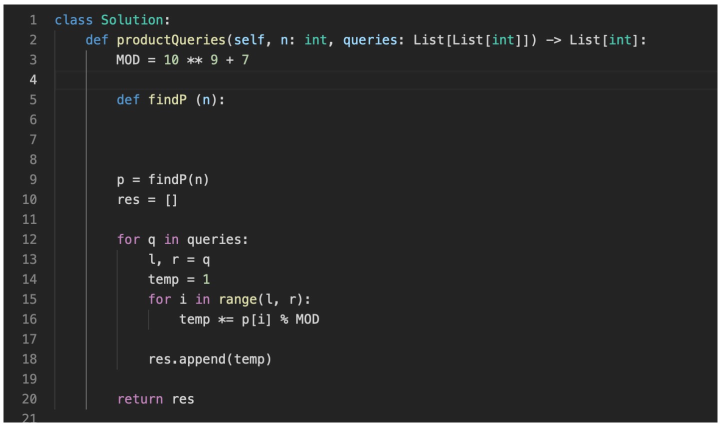
Given a positive integer `n`, there exists a **0-indexed** array called `powers`, composed of the **minimum** number of powers of `2` that sum to `n`. The array is sorted in **non-decreasing** order, and there is **only one** way to form the array.

You are also given a **0-indexed** 2D integer array `queries`, where `queries[i]` = [$left_i$, $right_i$]. Each `queries[i]` represents a query where you have to find the product of all `powers[j]` with $left_i$ <= j <= $right_i$.

Return *an array* `answers`*, equal in length to* `queries`*, where* `answers[i]` *is the answer to the* i^{th} *query*. Since the answer to the i^{th} query may be too large, each `answers[i]` should be returned **modulo** $10^9 + 7$.

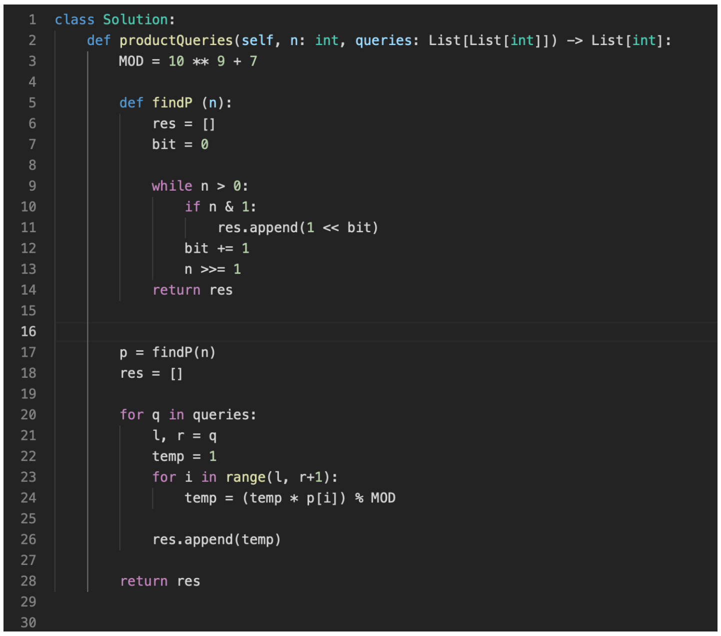
Example 1:

```
Input: n = 15, queries = [[0,1],[2,2],[0,3]]
Output: [2,4,64]
Explanation:
For n = 15, powers = [1,2,4,8]. It can be shown that powers cannot be a smaller size.
Answer to 1st query: powers[0] * powers[1] = 1 * 2 = 2.
Answer to 2nd query: powers[2] = 4.
Answer to 3rd query: powers[0] * powers[1] * powers[2] * powers[3] = 1 * 2 * 4 * 8 = 64.
Each answer modulo 10^9 + 7 yields the same answer, so [2,4,64] is returned.
```



```python
class Solution:
    def productQueries(self, n: int, queries: List[List[int]]) -> List[int]:
        MOD = 10 ** 9 + 7

        def findP (n):



        p = findP(n)
        res = []

        for q in queries:
            l, r = q
            temp = 1
            for i in range(l, r):
                temp *= p[i] % MOD

            res.append(temp)

        return res
```

bit manipulation



```python
class Solution:
    def productQueries(self, n: int, queries: List[List[int]]) -> List[int]:
        MOD = 10 ** 9 + 7

        def findP (n):
            res = []
            bit = 0

            while n > 0:
                if n & 1:
                    res.append(1 << bit)
                bit += 1
                n >>= 1
            return res


        p = findP(n)
        res = []

        for q in queries:
            l, r = q
            temp = 1
            for i in range(l, r+1):
                temp = (temp * p[i]) % MOD

            res.append(temp)

        return res

```