Intuition

When we're given a list of unique numbers and we want to form binary trees where each non-leaf node's value is the product of its children's values, it's intuitive to think about factors. If a number can be formed by multiplying two other numbers in the list, then it can be the root of a tree with those two numbers as children. This naturally leads to thinking about combinations of factors and how to efficiently determine how many trees each number can form as the root.

Approach

1. **Sorting and Set Creation:** Start by sorting the `arr` in ascending order. This allows us to efficiently determine when to stop considering pairs of factors (especially with our optimization). Also, create a set `s` from `arr` for O(1) lookup times.
2. **Dynamic Programming:** Use a dictionary `dp` where each key is a number from `arr` and its value represents the number of binary trees with that number as the root. Initialize each number's count to 1 to account for a single-node tree.
3. **Calculating Combinations:** For every number i in `arr`:
   - Check every number j in `arr` (where $j \leq \sqrt{i}$ because of optimization).
     - If i is divisible by j and $\frac{i}{j}$ is in `s`:
       - If $\frac{i}{j} = j$ (i.e., both factors are the same), then increment the count for i by $\text{dp}[j] \times \text{dp}[j]$.
       - Otherwise, increment the count for i by $\text{dp}[j] \times \text{dp}[\frac{i}{j}] \times 2$. We multiply by 2 to account for the two possible trees (with children j and $\frac{i}{j}$ or $\frac{i}{j}$ and j).
4. **Final Answer:** The final answer is the sum of all values in `dp` modulo $10^9 + 7$.

Complexity

- **Time complexity:**
  - Sorting the `arr` takes $O(n \log n)$.
  - The nested loop, because of our optimization, will run in average less than $n \times \sqrt{n}$, so it's $O(n\sqrt{n})$.
  - Overall, the time complexity is $O(n\sqrt{n})$.
- **Space complexity:**
  - The `dp` dictionary will have at most `n` elements, so it's $O(n)$.
  - The set `s` will also have `n` elements, so $O(n)$ for it too.
  - Combined, the space complexity is $O(n)$.