◆ Matchsticks to Square - Full Documentation

1. Problem Statement

You are given an array matchsticks where matchsticks[i] represents the length of the i-th matchstick. You need to determine if you can use all the matchsticks exactly once (without breaking them) to form a square.
Each side of the square must have equal length.

- Input: List[int] matchsticks
- Output: True if it's possible to make a square; False otherwise.

2. Intuition

To form a square:

- The total length of matchsticks must be divisible by 4.
- We need to partition the matchsticks into 4 groups where the sum of each group is equal.

This is a variation of the subset sum problem, which we solve using backtracking.

3. Key Observations

- If the sum of all matchsticks is not divisible by 4, it's impossible to form a square.
- Sorting the matchsticks in descending order helps place longer matchsticks first, which improves pruning efficiency in backtracking.
- Each matchstick must be used once and only once.

4. Approach

- Calculate total sum of matchsticks.
- If total % 4 != 0, return False.
- Sort the matchsticks in descending order.
- Use backtracking to try placing each stick into one of the 4 sides.
- If a configuration results in all sides having equal length, return True.

✅ Key Backtracking Logic:

- Try placing the current matchstick into one of the 4 sides.
- Only proceed if the current side's length plus the matchstick ≤ side_length.
- If placing the matchstick leads to a valid solution, return True.
- Use early termination if we try a stick on an empty side and it doesn't work (to avoid redundant cases).

5. Edge Cases

Edge Case	Expected Behavior
Only 1 matchstick	Cannot form square
All matchsticks are equal	Should return True
One stick is too long	Return False
Total sum not divisible by 4	Return False
Many small matchsticks	Works only if total is divisible by 4 and they can be grouped

6. Complexity Analysis

 Time Complexity:

- $O(4^n)$ in the worst case — each stick can potentially be tried on all 4 sides.
- Pruning using sorting and early stopping significantly reduces the actual time.

 Space Complexity:

- O(n) for the recursion stack (where n is number of matchsticks).
- O(1) additional space used apart from recursion and input.

7. Alternative Approaches

Approach	Feasibility
Dynamic Programming	Complex due to multiple group partitions
Greedy	Not sufficient, because ordering matters
Bitmask + Memoization	Possible for optimization, but overkill for n <= 15

8. Test Cases

Input	Output	Explanation
[1,1,2,2,2]	True	Forms square with sides of length 2
[3,3,3,3,4]	False	Cannot divide into equal sides
[1,1,1,1]	True	Each stick makes one side
[5,5,5,5,4,4,4,4,3,3,3,3]	True	Complex but possible
[1]	False	Single matchstick cannot form a square

9. Final Thoughts

- This problem is a good test of backtracking with pruning.
- Sorting in descending order significantly boosts performance.
- Although brute force is exponential, the problem constraints (max 15 matchsticks) allow it.
- Always check if early termination (like if total % 4 != 0) can be applied to reduce work.