

Partition Equal Subset Sum

0/1 Knapsack disguised as partition ■

● MEDIUM

■ Hook

Can you partition array into two subsets with equal sum? If total sum is odd $\rightarrow$ impossible. If even $\rightarrow$ find subset with sum = total/2. This is exactly 0/1 knapsack: can we pick items summing to target?

■ Core Intuition — Reduce to Subset Sum

■ Transformation

If total sum is S , we need subset summing to $S/2$. $dp[j]$ = true if subset summing to j exists. For each number: $dp[j] |= dp[j - num]$ (iterate j backward to avoid reuse).

■ DP Trace — [1,5,11,5]

```
sum=22, target=11
dp = [T,F,F,F,F,F,F,F,F,F,F,F] (size 12)
num=1: dp[1]=dp[0]=T → [T,T,F,F,F,F,F,F,F,F,F,F]
num=5: dp[6]=dp[1]=T, dp[5]=dp[0]=T
      → [T,T,F,F,F,T,T,F,F,F,F,F]
num=11: dp[11]=dp[0]=T!
      → dp[11]=TRUE → ANSWER = true ✓
Partition: {1,5,5}=11 and {11}=11
```

■ Code

```

public boolean canPartition(int[] nums) {
    int sum = 0;
    for (int n : nums) sum += n;
    if (sum % 2 != 0) return false;
    int target = sum / 2;
    boolean[] dp = new boolean[target+1];
    dp[0] = true;
    for (int num : nums) {
        for (int j=target; j>=num; j--) { // backward!
            dp[j] = dp[j] || dp[j-num];
        }
    }
    return dp[target];
}

```

■ TIME

$O(n \times \text{sum})$

■ SPACE

$O(\text{sum})$

■ Practice

#	Problem	Diff	Pattern
1	Partition Equal Subset Sum (LC 416)	Medium	0/1 Knapsack
2	Target Sum (LC 494)	Medium	Subset Sum / DFS
3	Last Stone Weight II (LC 1049)	Medium	Subset Sum
4	Subarray Sum Equals K (LC 560)	Medium	Prefix Sum
5	Count of Subset Sum	Medium	0/1 Knapsack count

- ✓ Reduce: can partition → find subset summing to total/2
- ✓ Odd total → impossible immediately
- ✓ $dp[j]$ = can we make sum j using elements so far?
- ✓ Backward iteration prevents reusing element
- ✓ Time: $O(n \times \text{sum})$ | Space: $O(\text{sum})$

■ MENTAL MODEL — Remember This Forever

This problem is a costume party: the 'partition' costume is hiding a 0/1 Knapsack underneath. Whenever you see 'can we pick elements summing to X', think knapsack subset sum. The boolean DP array is your 'reachable sums tracker' — each element expands what sums are achievable.