Leetcode 1011 — Capacity To Ship Packages Within D Days

# Problem Title:

# 1011. Capacity To Ship Packages Within D Days 🔗 [Leetcode Link](https://leetcode.com/problems/capacity-to-ship-packages-within-d-days/)

### Problem Statement:

You are given an array weights of positive integers where weights[i] represents the weight of the ith package and an integer D.

You need to ship all the packages within D days. The packages must be shipped in the order given. The **ship capacity** must be at least the heaviest package, and cannot exceed the total weight of all packages.

Return the **minimum capacity** needed to ship all the packages within D days.

### Input:

* weights = [1,2,3,4,5,6,7,8,9,10]
* D = 5

### Output:

* 15

### Key Insight:

We apply **Binary Search on the answer (ship capacity)**.

* **Left bound** = max weight in the array
* **Right bound** = total sum of weights

We check if it is **possible to ship within D days** for each mid capacity, and binary search accordingly.

### Java Code:

class Solution {

public int shipWithinDays(int[] weights, int D) {

int left = getMax(weights);

int right = getSum(weights);

while (left < right) {

int mid = left + (right - left) / 2;

if (canShip(weights, D, mid)) {

right = mid; // try to reduce capacity

} else {

left = mid + 1; // increase capacity

}

}

return left;

}

private boolean canShip(int[] weights, int D, int capacity) {

int days = 1;

int curr = 0;

for (int w : weights) {

if (curr + w > capacity) {

days++;

curr = 0;

}

curr += w;

}

return days <= D;

}

private int getMax(int[] arr) {

int max = arr[0];

for (int val : arr) max = Math.max(max, val);

return max;

}

private int getSum(int[] arr) {

int sum = 0;

for (int val : arr) sum += val;

return sum;

}

}

### Dry Run (weights = [1,2,3,4,5,6,7,8,9,10], D = 5)

### Explanation:

The best minimum ship capacity that allows shipping in 5 days is **15**.

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | mid | canShip | Action |
| 1 | 55 | ✅ | right = 55 |
| 2 | 32 | ✅ | right = 32 |
| 3 | 20 | ✅ | right = 20 |
| 4 | 13 | ❌ | left = 14 |
| 5 | 17 | ✅ | right = 17 |
| 6 | 15 | ✅ | right = 15 |
| 7 | 14 | ❌ | left = 15 |
| Done | 15 | ✅ | return 15 |

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | mid | canSplit(nums, 2, mid) | Action |
| 1 | 21 | ✅ yes (split into [7,2,5] and [10,8]) | right = 21 |
| 2 | 17 | ❌ no | left = 18 |
| 3 | 19 | ✅ yes (split into [7,2,5], [10,8]) | right = 19 |
| 4 | 18 | ✅ yes (same split) | right = 18 |
| 5 | 18 | Exit loop (left == right) | ✅ Return 18 |

### Time Complexity:

* O(N \* log(S))  
  where S = sum(weights) - max(weights)