Capacity to ship packages within D days

You are the owner of a Shipment company. You use conveyor belts to ship packages from one port to another. The packages must be shipped within ***'d'*** days.

The weights of the packages are given in an array ***'weights'***. The packages are loaded on the conveyor belts every day in the same order as they appear in the array. The loaded weights must not exceed the maximum weight capacity of the ship.

Find out the least-weight capacity so that you can ship all the packages within 'd' days.

Detailed explanation (Input/output format, Notes, Images)

Sample Input 1:

8 5
5 4 5 2 3 4 5 6

Sample Output 1:

9

Explanation for Sample Input 1:

In the test case, the given weights are [5,4,5,2,3,4,5,6] and these are needed to be shipped in 5 days. We can divide these weights in the following manner:

Day	Weights	Total
1	5, 4	9
2	5, 2	7
3	3, 4	7
4	5	5
5	6	6

The least weight capacity needed is 9, which is the total amount of weight that needs to be taken on Day 1.

→ Brute force :-

↳ Check all the possibility from max of array to sum of all the elements.

↳ Return the value which satisfies the condition.

```
int findDays(vector<int> &weights, int cap) {
    int days = 1;
    int load = 0;
    int n = weights.size();
    for (int i = 0; i < n; i++) {
        if (load + weights[i] > cap) {
            days += 1;
            load = weights[i];
        }
        else {
            load += weights[i];
        }
    }
    return days;
}

int leastWeightCapacity(vector<int> &weights, int d) {
    int maxi = *max_element(weights.begin(), weights.end());
    int sum = accumulate(weights.begin(), weights.end(), 0);

    for (int i = maxi; i <= sum; i++) {
        if (findDays(weights, i) <= d) {
            return i;
        }
    }
    return -1;
}
```

```
int leastWeightCapacity(vector<int> &weights, int d) {
    int low = *max_element(weights.begin(), weights.end());
    int high = accumulate(weights.begin(), weights.end(), 0);
    while (low <= high) {
        int mid = (low + high) / 2;
        int numberOfDays = findDays(weights, mid);
        if (numberOfDays <= d) {
            high = mid - 1;
        }
        else {
            low = mid + 1;
        }
    }
    return low;
}
```

↑

O(n) to O(log n) → using binary search.