BINARY SEARCH. -15.

CAPACITY TO SHIP PACKAGES WITHIN D DAYS

least capacity to ship packages within D days.

weights [ ]= [1 2 3 4 5 6 7 8 9 10] days=5.

→ N products and each product has certain weights. we have one ship and that ship runs Once per day. And you have to ensure that all products have to be Shipped within 5 days.

→ Now suppose, if a ship have capacity = 100 and all products that are present upload on it total weight will be = 55. (Summation of all product weight) So we can shipped everything in a "One day". But the problem says you can take 5 days.

→ But the question says "FIND THE LEAST CAPACITY".

→ Now suppose, if the Capacity of ship is 10. than on:→

first day = {1, 2, 3, 4} = 4 products.
second day = {5} because {5 + any other no. > 10}
Third day = {6}
fourth day = {7}
fifth day = {8}
sixth day = {9}
Seventh day = {10}

we end up taken 7 days but maximum ~~weight~~ day taken will be have to be 5 days.

Step3: Let the capacity of ship be 15.

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

1st day = {1, 2, 3, 4, 5}

2nd day = {6, 7}

3rd day = {8} because we cant go 8+9 = 17 > 15

4th day = {9}

5th day = {10}

So for solving first thing that we have to do, it to find that least capacity.

↳ ST → first point allways remember, what will be the maximum weight the ship has to be of atleast that size.
So size capacity of the ship has to be atleast 10 capacity in our case.

→ max. capacity ~~of the (size/capacity~~ of the ship is the summation of the weight of all the product.
Capacity is 55 in our case.

∴ So the ANSWER lies b/w the max. ~~cap~~ weight and summation of all wts.

∴ Now, how to find the least capacity of the ship.

for (cap → (max, sum)).

(10 → 55)

So start looking from 10,

```
for(cap → (max → sum))
{
  daysReq = fun(wt, cap)
    if (dayReq <= days)
      return daysReq;
}
```

LINEAR SEARCH

```
// This will gare
   NUMBER OF DAYS REQUIRED.
int fun (wt, cap)
  {
   day = 1 , load = 0;
   for (i = 0 → n-1)
    {
     if (load + wt[i] > cap)   // if capa < load+wt[i] move on the
                                  next day.
       {
        day = day + 1 ;
        load = wt[i]
       }
     else
        load += wt[i];
    }
    return days ;
  }
```

~~TIME COMPLEXITY = O(sum-max)+1~~

TC = |O(sum-max)+1 |x O(N)

↳ This is Somewhat about, linear * linear so it is Quadratic $O(n^2)$.

```
weights = [1 2 3 4 5 6 7 8 9 10]
    f (weight, days)
        {
          low = max, high = sum of arr)
        while (low <= high)
            {
              mid = (low+high)/2;
              no.of days = find(wt, mid)
              if ( no.of days <= days)
                 {
                   high = mid-1;
                 }
               else
                 low = mid+1;
                }
          return low;
```

T.C = $\log_2(\text{sum} - \text{max}+1) * O(N)$ ↑ still using (fun(wt, mid)) to calculate number of days.

S.C = O(1).