

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 has capacity = 100 and all products
that are present upload on it total weight will be = 55.
(Summation of all product weight). So we can ship
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. then or →

first day = $\{1, 2, 3, 4\}$ = 4 products.

second day = $\{5\}$ because $\{5 + \text{any other no.} > 10\}$

third day = $\{6\}$

fourth day = $\{7\}$

fifth day = $\{8\}$

sixth day = $\{9\}$

seventh day = $\{10\}$

We end up taking 7 days but maximum ^{day} weight taken will be have to be
5 days.

Step 3: 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 can't 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 all ways remembers, what will be the maximum weight the ship has to be of at least that size.
So size of the ship has to be at least 10 capacity in our case.

→ max. 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.

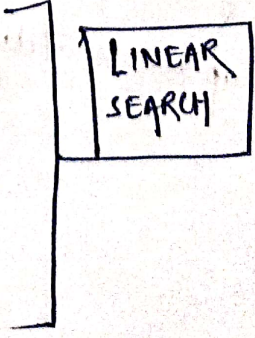
for (cap → (max, sum)).

(10 → 55)

So start looking from 10,

for (cap → (max → sum))

```
{ daysReq = fun(wt, cap)
  if (daysReq <= days)
    return daysReq;
}
```



// This will give
NUMBER OF DAYS REQUIRED.

int fun (wt, cap)

{

day = 1, load = 0;

for (i = 0 → n-1)

{

if (load + wt[i] > cap) // if cap < load + wt[i] move on the next day.

{

day = day + 1;

load = wt[i]

}

else

load += wt[i];

}

return day;

}

~~TIME COMPLEXITY = $O(\text{sum} - \text{max} + 1)$~~

~~TC = $(O(\text{sum} - \text{max}) + 1) \times O(N)$~~

↳ 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 = fun(wt, mid)

if (no. of days <= days)

{

high = mid - 1;

}

else

low = mid + 1;

}

return low;

T.C = $\log_2(\text{sum} - \text{max} + 1) * O(N)$

↳ since using (fun(wt, mid))
to calculate number of days.

S.C = $O(1)$.