368 Largest Divisible Subset

Dynamic Programming

Idea: For every element, it can belong to largest divisible set or it will not belong.

dp[i] → represent maximum size of largest divisible subset for 0.. i

$$dp[i] = \max_{\text{for } 0 \le j < i \text{ and } (nums[j] \,\%\, nums[i] = 0 \text{ or } nums[i] \,\%\, nums[j] = 0)} (1 + dp[j])$$

Every time we check

nums[i] % nums[j]

nums[j] % nums[i]

Instead we will sort nums array and only check 1 condition

nums[i] % nums[ij] == 0

$dp[i] = \max(dp[i], 1 + dp[j])$ where $0 \le j < i$ and $nums[j] \% nums[i] = 0$

Rec formula (i)

After getting DP array in bottom up approach we get optimal answer

To get optimal solution we do Top Down on built dp array

eg nums: 2 | 1 | 5 | 8

sort nums: 1 | 2 | 5 | 8

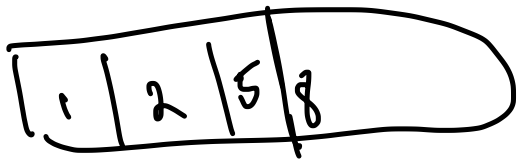
Initialize dp: 1 | 1 | 1 | 1

Initialized to 1 as any element itself is a divisible subset

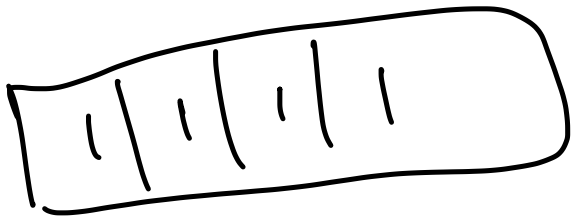
```
for x in range(1, len(dp)):
```

Rec formula (1)

nums



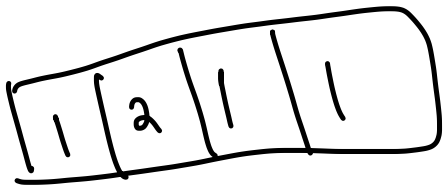
dp



for $x = 1$

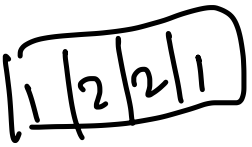
$dp[1] = MAX(dp[1], 1 + dp[0])$ as $nums[1] > nums[0]$

$= 2$



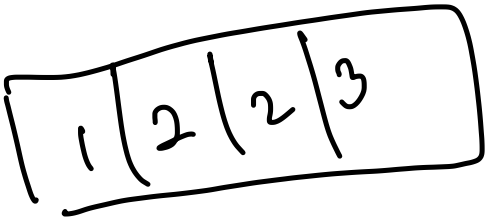
for $x = 2$

$dp[2] = MAX(dp[2], 1 + dp[0]) = 2$



for $x = 3$

$dp[3] = MAX(dp[3], 1 + dp[1], 1 + dp[0]) = 3$



nums

1	2	5	8

dp

1	2	2	3

now from dp array we know that size of largest divisible subset is 3

How to find that subset

we find max in dp array

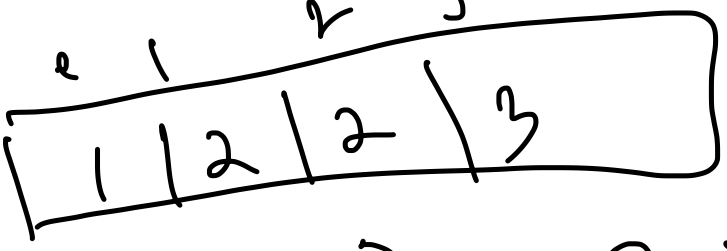
```
index = dp.index(max(dp))

ans = []
ans.append(nums[index])

current = max(dp)
for x in range(index-1, -1, -1):
    if dp[x]+1 == current and (ans[len(ans)-1] % nums[x] == 0):
        ans.append(nums[x])
        current = dp[x]
```

index of max value in dp

dp



ans = [nums[3]] = [8]

current = 3

for x = 2

dp[2] + 1 = current True

8 % nums[2] = 0

8 % 5 = 0 False

For x = 1

dp[1] + 1 = current True

8 % 4 = 0 True

ans = [8, 4]

current = dp[1] = 2

For $x = 0$

$dp[0] + 1 = 2$ true

$4 \% 1 = 0$ true

$ans.append(1)$ $[8, 4, 1]$

return $ans[::-1]$