740- Delete and Earn

Brute force appraoch:- Again create a decision tree, but again before that it's important to notice, that order of elements is not important.
And the repitition of elements does not have a big significance as.

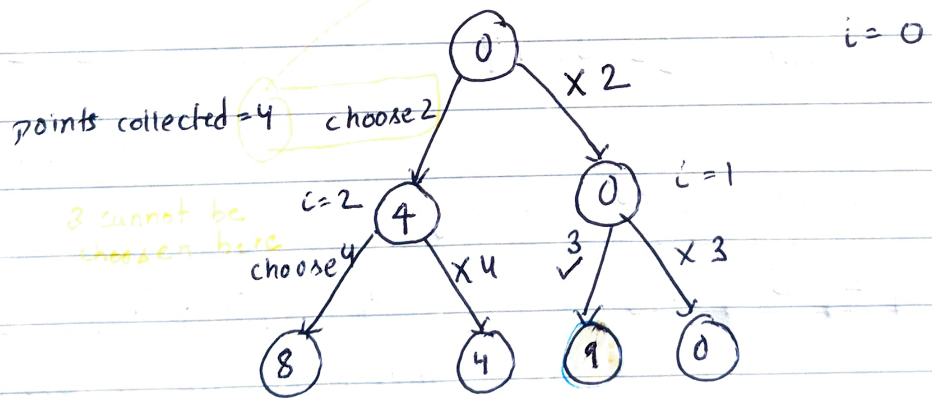
nums = [2, 2, 3, 3, 3, 4]

removing one 3 will remove all 2 and 4. So we can collect all the points all at one $3 \times 3 = 9$

1) Therefore first remove duplicates and store the count in hashmap/dict.

[2, 3, 4] dict= {2:2, 3:3, 4:1}

2) Decision Tree



But this will take 2^n computation in worst case if there are no repitions. Seems like memorization cannot happen here.

Creation of recurrence relation (Another approach)

$F(x)$ = max points at x

$F(x) = \max\left(F(x-1),\ F(x-2) + x * dict[x]\right)$

If x is choosen then points that can be earned is

$$F(x) = F(x-2) + x \times dict[x]$$ ← point for choosing x

else it would be same as-

$$F(x) = F(x-1)$$

∴ max should be choosen

⇒ Base case can be created and it can be solved by using both memorization and tabulation technique.

→ But again above approch doesn't consider the gaps between nums. for example.

nums = [1, 3, 4, 10000]

The above algorithm starts (top-down) with 10000 and goes till base case, many iterations are unnecessary.

bottom-up (~~100~~ 0 → 10000) = $O(N + \text{maximum number})$

⇒ Sort the elements and iterate (more Efficient)

nums = [2, 3, 4]   dict = { 2 : 2, 3 : 3, 4 : 1 }

DP can be applied this way

2	3	4

dp =

4	9	9

max_points $\boxed{\text{till dp[0:1]}}$

max_points $\boxed{\text{till dp[0:2]}}$

max points $\boxed{\text{till dp[0:3]}}$

Base case:

dp[0] = nums[0] × dp[nums[0]] → 2 × 2 = 4

dp[1] = max(dp[0], nums[1] × dp[nums[1]]) = max(4, 9) = 9

Remember dp[1] cannot be choosen if dp[0] is choosen. as 2+1 == 3 if num[1] was not 3 then (>3)

$$dp[1] = \max(dp[0],\ dp[0] + nums[1] \times dict[nums[1]])$$

$$\cancel{dp[i]} = A = \max\begin{pmatrix} nums[i] \times dict[nums[i]] + dp[i-2], \\ dp[i-1] \end{pmatrix}$$

$\Rightarrow$ if $nums[i] == nums[i] + 1$

$$B = nums[i] \times dict[nums[i]] + dp[i-1]$$

$\Rightarrow$ else.

$$dp[i] = \max(A, B)$$

$\rightarrow O(N \log N)$