Q.12

// A is an input array of A.length = n, containg +ve integers

// A is assumed to be global

(1) Given i, S1, S2 compute

3PART(i, S1, S2) = whether A[1...i] can be divided into 3 parts with target sum S1 and S2, for first two partitions respectively.

(2)

$$
3PART(i, S1, S2) = \underset{\text{OR}}{\text{logical}} \begin{cases} 3PART(i-1, S1-A[i], S2) & ; \text{if } S1 \geq A[i] \\ 3PART(i-1, S1, S2-A[i]) & ; \text{if } S2 \geq A[i] \\ 3PART(i-1, S1, S2) & ; \text{otherwise} \end{cases}
$$

3PART(i, 0, 0) = True ← [3PART(i,0,0) = True]

~~3PART(i, S1, 0) = True~~

3PART(0, S1, S2) = False

Base cases

(3) 3PART(i, S1, S2) tells whether A[1...i] can be divided into 3 parts where the first two partitions formed with target sum S1 and S2.

Now the ith element may be included in the 1st partition or in the 2nd partition or may not be included in first two partitions (i.e., it will be in the 3rd partition). Because either of these three cases will take place at a time, logical OR operation is needed in between these three for the recurrance. If for the first partition the target sum has not decreamented to zero, then we might consider including the ith element in the 1st partition. Thus 3PART(i-1, S1-A[i], S2) returns whether A[1... i-1] can be divided into 3 partitions where first two partitions will have target sum S1-A[i] and S2 respectively, so that the target sum of first partition of A[1...i] effectively becomes S1. Now if A[i] is included in first partition i.e., 3PART (i-1, S1 - A[i], S2) returns True, then the other two reccurences will not be checked, because A[i] can't be included in more than one partition due to the disjoint property of the three partitions.

If 3PART (i-1, S1 - A[i], S2) is returning False then we check for including A[i] in 2nd partition or the 3rd partition one by one. similarly.

In case of Base condition, 3PART (i, 0, 0) will be True, because there A[1...i] is being divided into 3 partitions where first two partitions have target sum = 0, which is trivially possible if first two partitions are considered as empty sets.

3PART (0, S1, S2) will be False, because A[1...0] can't be divided into three partitions with first two partitions having target sum S1 and S2.

(4) P [0...n] [0... Sum/3] [0... Sum/3] is an 3D array, where n = A.length, and sum = sum of all elements of A[1...n]

P[i][j][k] will store value of 3PART (i, j, k)

(5) If Sum (A[1...n]) i.e., sum of all array elements is not divisible by 3, i.e., Sum (A[1...n]) % 3 != 0 then return False.

$\forall S1 \in [0 \dots s/3]$ and $\forall S2 \in [0 \dots s/3]$ (here s = sum of $A[1] \dots A[n]$)

initialise $P[0][S1][S2]$ = False

$\forall i \in [0 \dots n]$

initialise $P[i][0][0]$ = True

for $i = 1 \dots n$

for $S1 = 1 \dots s/3$

for $S2 = 1 \dots s/3$

compute $P[i][j][k]$ using recursive formula on array A.

(6) Whether A can be divided into 3 disjoint partitions B, C, D such that $B \cup C \cup D = A$ and total value of B, C, D are equal, is equivalent to DP problem 3PART(n, $s/3$, $s/3$) where n = A.length and s = Sum($A[1 \dots n]$)

We are just considering target sum of two partitions because if the partitions have equal sum = $s/3$, i.e., $(2s/3)$ becomes sum of first two partitions, then the third partition will definitely have a sum of $s/3$.

Thus finding whether A can be divided into 3 partitions where first two partitions having sum of $s/3$ each, will actually meet the purpose of the problem i.e., finding if B, C, D exists or not with $B \cup C \cup D = A$ and

sum(B) = sum(C) = sum(D) = sum(A)/3

(7) Space Complexity = $O(n^3)$

Time Complexity = $O(n^3)$ since computing 3PART(i, j, k) will require $O(1)$ time complexity and there are $O(n^3)$ entries in the 3D array $P[\,][\,][\,]$.