Understanding Problem Statement

Given an array of n-integers, construct product array such that prod[i] is equal to product of all elements except arr[i] without using division operator. Consider array [10,20,30,40]. Now prod[0] should be product of arr[1],arr[2],arr[3].we should not include arr[0] And prod[1] should be product of arr[0],arr[1] and arr[2] prod_arr = [24000,12000,8000,6000] #ans

Approach 1

1. First we can find the product of all elements. all_prod
2. while finding product array, we can add prod[i] by dividing all_prod with arr[i].
3. But condition is we should not use division operator.So we cannot use this approach

Approach 2

1. Find left product array i.e product of elements starting from left till that element. left prod[i] = arr[0] x arr[1] x .. arr[i-1] x arr[i] leftprod = [10,200,6000,24000]
2. Similarly find right product array i.e product of elements starting from right till that elements right prod[i] = arr[n] x arr[n-1] x ... arr[i+1] x arr[i]
3. Now to find actual prod[i] we can make use of left prod and right prod. prod[i] = left prod[i-1] x right prod[i+1]
4. To find left prod we have to scan array once. O(n) To find right prod we have to scan array once. O(n) To find actual prod we have to iterate and make use of left prod and right prod. O(n)
5. We use three arrays of space n i.e 3n i.e O(n)

Time Complexity : O(n)

Space Complexity : O(n)

Implementation

In [15]:
```
def prod_array(arr,size):
    if size == 1:
        return []
    prod_arr = [0]* size
    left_prod = []
    right_prod = [0]* size
    lprd = 1
    for i in range(size):
        lprd *= arr[i]
        left_prod.append(lprd)
    rprd = 1
    for i in range(size-1,-1,-1):
        rprd *= arr[i]
        right_prod[i]= rprd
    prod_arr[0] = right_prod[1]
    prod_arr[-1] = left_prod[size-2]
    if size == 2:
        return prod_arr
    for i in range(1,size-1):
        prod_arr[i] = left_prod[i-1]* right_prod[i+1]
    return prod_arr
```

In [16]:
```
arr = [10,20,30,40]
size = len(arr)
print(prod_array(arr,size))
# ans
# [24000, 12000, 8000, 6000]
```

```
[24000, 12000, 8000, 6000]
```

In [17]:
```
arr = [10]
size = len(arr)
print(prod_array(arr,size))
# ans
# []
```

```
[]
```

In [28]:
```
arr = [10,20]
size = len(arr)
print(prod_array(arr,size))
# ans
# [20, 10]
```

```
[20, 10]
```

Optimized logic

```
Instead of using different left prod, right prod and actual prod, we
can use single array to directly
compute result.
Given array [10,20,30,40]. First compute left such that element is not
included i.e [1,10,200,6000]
Then iterate from right and update the same list.
```

In [29]:
```
def prod_arr_opt(arr,size):
    prod_arr = [None]*size
    temp = 1
    for i in range(size):
        prod_arr[i] = temp
        temp = temp* arr[i]
    temp = 1
    for i in range(size-1,-1,-1):
        prod_arr[i] = prod_arr[i] * temp
        temp = temp* arr[i]
    return prod_arr
```

In [30]:
```
arr = [10,20,30,40]
size = len(arr)
print(prod_arr_opt(arr,size))
# ans
# [24000, 12000, 8000, 6000]
```

```
[24000, 12000, 8000, 6000]
```

In [31]:
```
arr = [10]
size = len(arr)
print(prod_arr_opt(arr,size))
# ans
# [1]
```

```
[1]
```

In [32]:
```
arr = [10,20]
size = len(arr)
print(prod_arr_opt(arr,size))
# ans
# [20, 10]
```

```
[20, 10]
```

In [ ]:
```
```