Understanding the problem statement

Given an unsorted array of positive integers, find the number of triangles that can be formed with three different elements as three sides of traingle input : [2,4,3,1] output : 1 ans : [2,3,4] only can form traingle Triangular property : sum of any two sides should be greater than third side. Sides of traingle [a,b,c] --> a+b > c and a+c > b and b+c > a [1,2,3] cannot form triangle as 1+2 = 3 .

Approach 1

1. Find all the possibilities of 3 elements from given n elements. nc3 i.e O(n^3)
2. For each possibility check Triangular property condition i.e a+b > c and a+c > b and b+c > a . O(1)
3. So total time complexity is O(n^3) * O(1) = O(n^3)

Approach 2

1. Sort the given array. O(nlogn)
2. when elements are sorted we need not check 3 conditions. When [a,b,c] are in ascending order just checking a+b > c will be enough because as c is already greater than a and b , a+c > b and b+c > a will be true.
3. Take each element from the array i , take next element as j and check for the elements k which are smaller than a[i]+a[j]. When bigger element is encountered we can stop iterating and update the count.K need not be initialized in the algorithm because once it is true for smaller sum of i,j then it will be true for i and bigger j. O(n^2)
4. Once bigger element is encountered we can increase sum a[i]+a[j] by incrementing j.
5. Repeat steps 3 and 4 i,e move i and j and keep updating count.
6. See reference screenshot for complete algorithm.
7. so Total time complexity = O(nlogn) + O(n^2) = O(n^2)

Implementation

In [8]:
```python
def findPossibleTriangleCount(arr,n):
    arr = sorted(arr)
    count = 0
    for i in range(n-2):
        k = i+2
        for j in range(i+1,n-1):
            while((k<n) and ((arr[i]+arr[j])>arr[k])):
                k+=1
            count = count+(k-j-1)
    return count
```

In [15]:
```python
arr = [6,7,8,10,12,14,50]
findPossibleTriangleCount(arr,len(arr))
# ans 18
```

Out[15]:
```
18
```

In [16]:
```python
arr = [7, 3, 6, 4]
findPossibleTriangleCount(arr,len(arr))
# ans 3
```

```
Out[16]: 3
```

```
In [17]: arr = [10, 21, 22, 100, 101, 200, 300]
         findPossibleTriangleCount(arr,len(arr))
         # ans 6
```

```
Out[17]: 6
```

```
In [ ]:
```