TRIPLET WITH GIVEN SUM PROBLEM

Problem definition:

Given an (unsorted) integer array, find a triplet (a sub-array with 3 elements) such that summation of them is equal to the given sum.

Example:

Inputs: **array** = {1, 3, 2, 5, 4, 7} and **sum** = 10
Output: {1, 2, 7} or {3, 2, 5} or {4, 5, 1}

TRIPLET WITH GIVEN SUM PROBLEM

Solution 1 (Naïve solution):

```
findTriplet (array, sum):
  n ← length of the array
  for i from 0 to n-2:
     for j from i+1 to n-1:
        for k from j+1 to n:
           if array[i] + array[j] + array[k] == sum then
              return [array[i], array[j], array[k]]
           end if
        end for
     end for
  end for
  return -1
```

TRIPLET WITH GIVEN SUM PROBLEM

Solution 1 (Naïve solution):

The algorithm uses a loop for each element of the output.

```
findTriplet (array, sum):
    n ← length of the array
    for i from 0 to n-2:
        for j from i+1 to n-1:
            for k from j+1 to n:
                if array[i] + array[j] + array[k] == sum then
                    return [array[i], array[j], array[k]]
                end if
            end for
        end for
    end for
    return -1
```

TRIPLET WITH GIVEN SUM PROBLEM

Solution 1 (Naïve solution):

```
findTriplet (array, sum):
    n ← length of the array
    for i from 0 to n-2:
        for j from i+1 to n-1:
            for k from j+1 to n:
                if array[i] + array[j] + array[k] == sum then
                    return [array[i], array[j], array[k]]
                end if
            end for
        end for
    end for
    return -1
```

The algorithm uses a loop for each element of the output.

Example:
inputs: {1, 3, 2, 5, 4, 7}, sum=10

	i	j	k	sum
Step 1	1	3	2	6
Step 2	1	3	5	9
Step 3	1	3	4	8
Step 4	1	3	7	11
Step 5	1	2	5	8
Step 6	1	2	4	7
Step 7	1	2	7	10

1,2,7 -> 1 + 2 + 7 = 10
then return [1,2,7]

TRIPLET WITH GIVEN SUM PROBLEM

Solution 1 (Naïve solution):

```
findTriplet (array, sum):
  n ← length of the array
  for i from 0 to n-2:
     for j from i+1 to n-1:
        for k from j+1 to n:
           if array[i] + array[j] + array[k] == sum then
              return [array[i], array[j], array[k]]
           end if
        end for
     end for
  end for
  return -1
```

Best Case Time Complexity: **Ω(1)**
(occurs when the first three elements meet the condition)

Worst Case Time Complexity: **O(n^3)**
(occurs if there is no such triplet or the triplet is at the end of the array)

TRIPLET WITH GIVEN SUM PROBLEM

Solution 2 (hashing):

```
findTriplet (array, sum):
    n ← length of the array
    dictionary = { }
    for i from 0 to n
        append array[i] to dictionary
    end for
    for i from 0 to n – 1
        for j from i +1 to n
            x = sum - (array[i] + array[j]
            if x is in dictionary then
                return [array[i], array[j], x]
            end if
        end for
    end for
    return -1
```

TRIPLET WITH GIVEN SUM PROBLEM

Solution 2 (hashing):

```
findTriplet (array, sum):
    n ← length of the array
    dictionary = { }
    for i from 0 to n
        append array[i] to dictionary
    end for
    for i from 0 to n – 1
        for j from i +1 to n
            x  = sum - (array[i] + array[j])
            if  x is in dictionary then
                return [array[i], array[j], x]
            end if
        end for
    end for
    return -1
```

The algorithm saves the array in a dictionary, in this way it eliminates the need for one loop. Thus, two nested loops become enough.

TRIPLET WITH GIVEN SUM PROBLEM

Solution 2 (hashing):

Example:
inputs: {1, 3, 2, 5, 4, 7}, sum=10

```
findTriplet (array, sum):
    n ← length of the array
    dictionary = { }
    for i from 0 to n
        append array[i] to dictionary
    end for
    for i from 0 to n – 1
        for j from i +1 to n
            x = sum - (array[i] + array[j])
            if x is in dictionary then
                return [array[i], array[j], x]
            end if
        end for
    end for
    return -1
```

TRIPLET WITH GIVEN SUM PROBLEM

Solution 2 (hashing):

```
findTriplet (array, sum):
    n ← length of the array
    dictionary = { }
    for i from 0 to n
        append array[i] to dictionary
    end for
    for i from 0 to n – 1
        for j from i +1 to n
            x = sum - (array[i] + array[j])
            if x is in dictionary then
                return [array[i], array[j], x]
            end if
        end for
    end for
    return -1
```

Example:
inputs: {1, 3, 2, 5, 4, 7}, sum=10

dictionary = {1, 3, 2, 5, 4, 7}

TRIPLET WITH GIVEN SUM PROBLEM

Solution 2 (hashing):

```
findTriplet (array, sum):
    n ← length of the array
    dictionary = { }
    for i from 0 to n
        append array[i] to dictionary
    end for
    for i from 0 to n – 1
        for j from i +1 to n
            x = sum - (array[i] + array[j]
            if x is in dictionary then
                return [array[i], array[j], x]
            end if
        end for
    end for
    return -1
```

Example:
inputs: {1, 3, 2, 5, 4, 7}, sum=10

dictionary = {1, 3, 2, 5, 4, 7}

	i	j	x	is x in dictionary
Step 1	1	3	6	no
Step 2	1	2	7	yes

return [1,2,7]

TRIPLET WITH GIVEN SUM PROBLEM

Solution 2 (hashing):

```
findTriplet (array, sum):
    n ← length of the array
    dictionary = { }
    for i from 0 to n
        append array[i] to dictionary
    end for
    for i from 0 to n – 1
        for j from i +1 to n
            x = sum - (array[i] + array[j]
            if x is in dictionary then
                return [array[i], array[j], x]
            end if
        end for
    end for
    return -1
```

Best Case Time Complexity: **Ω(n)**
(occurs when the first two elements and any other element meet the condition)
(it is not constant because we have spent O(n) time for hashing)

Worst Case Time Complexity: **O(n^2)**
(occurs if there is no such triplet)