**4 - Arithmetic Slices**

A sequence of number is called arithmetic if it consists of at least three elements and if the difference between any two consecutive elements is the same.

For example, these are arithmetic sequence:

1, 3, 5, 7, 9

7, 7, 7, 7

3, -1, -5, -9

The following sequence is not arithmetic.

1, 1, 2, 5, 7

A zero-indexed array A consisting of N numbers is given. A slice of that array is any pair of integers (P, Q) such that 0 <= P < Q < N.

A slice (P, Q) of array A is called arithmetic if the sequence:  
A[P], A[p + 1], ..., A[Q - 1], A[Q] is arithmetic. In particular, this means that P + 1 < Q.

The function should return the number of arithmetic slices in the array A.

**Example:**

A = [1, 2, 3, 4]

return: 3, for 3 arithmetic slices in A: [1, 2, 3], [2, 3, 4] and [1, 2, 3, 4] itself.

**Breaking down the problem:**

In order to solve this problem, we would want to deal with the simplest case:

If A = [1, 2, 3] then we could just check if 1 - 2 = 2 - 3 and then return sum which would be 1 since there is only three elements to check.

So now that we have identified the simplest case, we could go a step further and analyze another case:

If A= [1,2,3,4,5] then we would have to check

1. A[0] - A[1] = A[1] - A[2] which would set the sum + 1.
2. A[0] - A[1] = A[1] - A[2] = A[2] - A[3] which would set the sum + 1.
3. A[0] - A[1] = A[1] - A[2] = A[2] - A[3] = A[3] - A[4] which would set the sum + 1.
4. A[1] – A[2] = A[2] – A[3] which would set the sum +1
5. A[1] – A[2] = A[2] – A[3] = A[3] – A[4] which would set the sum + 1
6. A[2] – A[3] = A[3] – A[4] which would set the sum +1

From this analysis we could notice that there is a lot of repetition. One of the main things we can conclude is that you need to re calculate the length of the arithmetic slice each time you iterate. Not only do we have to re calculate the length but also re conclude that it is an arithmetic slice.

We could identify that the simple case is recalculated every time you iterate through the array. You could use that subproblem to solve other subproblems like in the case above.

**Solution:**

From the analysis of the problem, we could use Dynamic Programming to solve this problem.

* For each subproblem, we can conclude that if two neighboring subproblems are arithmetic slices then it would be the sum of those subproblems plus one for that range of indices.
  + Since this is true, we will iterate through the array and store the sum of the arithmetic slices up to that point.
  + If the next check is an arithmetic slice it would add one to the previous sum. This is done so that we don’t must re calculate how many arithmetic slices exist up to that point.
* If there is a case in which one of the neighboring subproblems are not arithmetic slices, then the sum would not increase for that range and continue to the next index to check if they are arithmetic slices.
* For the structure to store the sum of the range, we will use an array with the same length.
* Once we are done then we could simply return the sum of the new structure.