Sherlock Holmes Tutorial

Pre-requisites:

This problem will require dynamic programming to solve. Make sure your basics of recursion and dynamic programming are clear before attempting this problem. The following links can help you learn the prerequisites:

- http://www.geeksforgeeks.org/recursion/
- https://www.topcoder.com/community/data-science/data-science-tutorials/an-introduction-to-recursion-part-1/
- https://www.topcoder.com/community/data-science/data-science-tutorials/an-introduction-to-recursion-part-2/
- https://www.topcoder.com/community/data-science/data-science-tutorials/dynamic-programming-from-novice-to-advanced/

Problem Description:

We have two arrays, A and B. A_i and B_i are the respective elements of A and B.

We have to find the number of ways such that the difference of values in A and B is less than or equal to a given threshold Q.

Difficulty Level:

Easy-Medium

Editorial:

The problem can indeed be solved by dynamic programming (DP). The DP state can be defined as **F(idx, diff, flag)** where **idx** represents the index of the problem, **diff** represents the absolute difference b/w the difficulties of the puzzles solved by the two of them and **flag** represents whether the difference as a whole is positive or negative. So, the state as a whole represents the number of ways of having given absolute difference as **diff** of **flag** parity b/w the two persons till the $\mathbf{i^{th}}$ index.

So, now at a particular state **F(idx, diff, flag)**, we have four choices to where we can make a transition. Let us look at the pseudo code to see how the transitions are made in the recursive function.

```
int F(idx, diff, flag) {
    if ( idx == n+1 ) return (diff <= Q);
    if ( flag ) diff = -diff;
    ans = 0;

    //both of them attempt this puzzle
    act_diff = diff + A[idx] - B[idx];
    ans += F(idx+1, abs(act_diff), act_diff < 0);

    //Mycroft attempts, Sherlock does not
    act_diff = diff + A[idx];
    ans += F(idx+1, abs(act_diff), act_diff < 0);

    //Mycroft does not attempt, Sherlock does
    act_diff = diff - B[idx];
    ans += F(idx+1, abs(act_diff), act_diff < 0);

    //both of them do not attempt
    act_diff  = diff;
    ans += F(idx+1, abs(act_diff), act_diff < 0);

    return ans;
}
```

In the above pseudocode, **flag** is set to 1 if diff is negative (< 0) or set to 0 if diff is non-negative (>= 0). We are actually updating our diff to new_diff as soon as we make a new choice and updating the flag variable accordingly. The updated flag value in the new state will again depend on the parity of the new_diff.

Complexity of solution:

Overall complexity of the solution is in fact **O(N*Q)**. N is the number of problems attempted by them in a particular week and Q is the threshold value given.