**Problem Name:** Three sum closest

**Topics:**

**Companies:**

**Level:** Easy

**Language:** C++

**Problem Statement**:

**Input Format:**

**Output Format:**

**Constraints:**

**Examples:**

**Brute force Solution:**

**Explanation:**

**Code:**

**Time Complexity**: O(N3)

**Space Complexity: O(1)**

**Optimized Solution:**

Explanation: Sort the vector and then no need to run *O(N^3)* algorithm as each index has a direction to move.

The code starts from this formation.

----------------------------------------------------

^ ^ ^

| | |

| +- second third

+-first

if *nums[first] + nums[second] + nums[third]* is smaller than the *target*, we know we have to increase the sum. so only choice is moving the second index forward.

----------------------------------------------------

^ ^ ^

| | |

| +- second third

+-first

if the *sum* is bigger than the *target*, we know that we need to reduce the *sum*. so only choice is moving '*third*' to backward. of course if the *sum* equals to *target*, we can immediately return the *sum*.

----------------------------------------------------

^ ^ ^

| | |

| +- second third

+-first

when *second* and *third* cross, the round is done so start next round by moving '*first*' and resetting *second* and *third*.

----------------------------------------------------

^ ^ ^

| | |

| +- second third

+-first

while doing this, collect the *closest sum* of each stage by calculating and comparing delta. Compare *abs(target-newSum)* and *abs(target-closest)*. At the end of the process the three indexes will eventually be gathered at the end of the array.

----------------------------------------------------

^ ^ ^

| | `- third

| +- second

+-first

if no exactly matching *sum* has been found so far, the value in *closest* will be the answer

**Code:**

**Time Complexity**: O(N2)

**Space Complexity:** O(1)