Deep Shah (002766755)

Program Structure & Algorithms

Spring 2023(Sec 03)

Assignment-2

Task:

There are two task to be performed.

Task list:

- Implement the 3-SUM code using the Quadrithmic, Quadratic, and QuadraticWithCalipers approaches for the experiment.
- Comparing the time complexities of 3-SUM for the above three approaches and adding Cubic.

Relationship Conclusion:

There are mainly three approaches for solving 3-SUM problem.

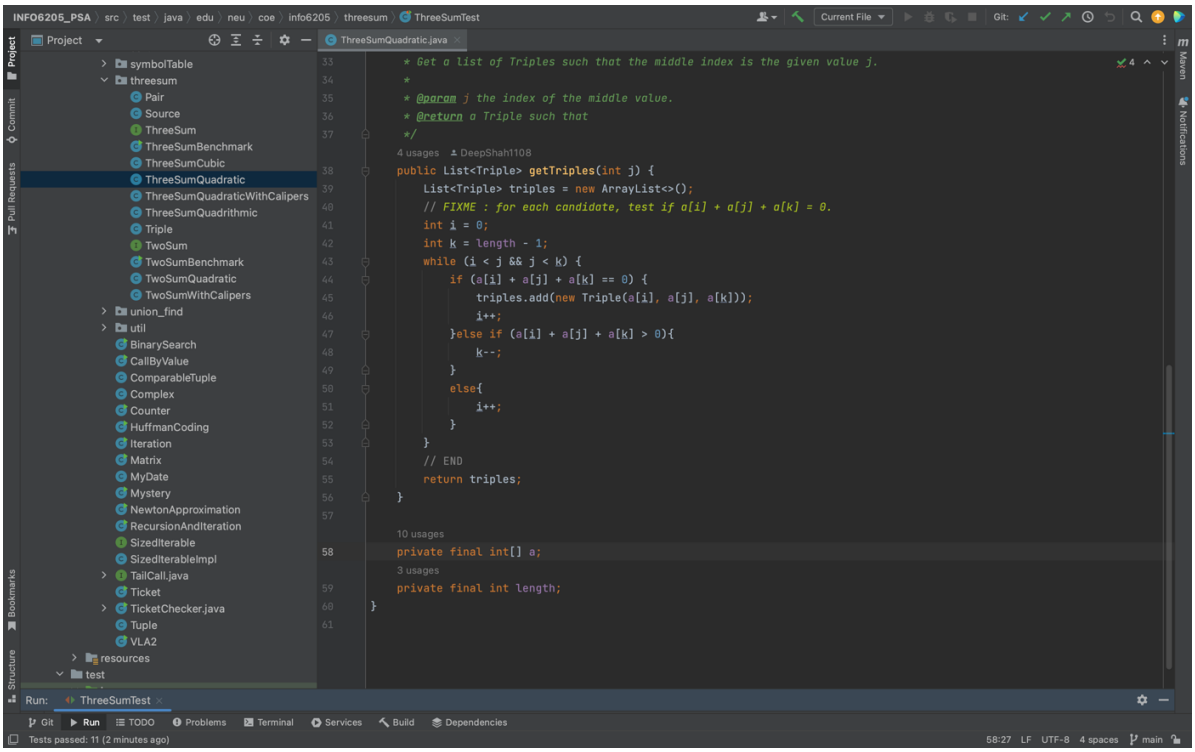
1. Cubic
2. Quadratic
3. Quadramithic

Looking at the time complexity for the above approaches. Cubic approach has the worst time complexity with $O(N^3)$. Quadrithmic approaches is a bit better than the Cubic but not the best having a time complexity of $O(N^2 log N)$.
The best and the efficient approach is the Quadratic approach that has the time complexity of $O(N^2)$. Also, we tried the Quadratic approach with calipers that is basically a two-pointer approach. In this the time complexity is same as that of Quadratic (i.e., $O(N^2)$), but the run time is much faster than the quadratic. Hence, we get the result in much less time.

Evidence to Support that conclusion:

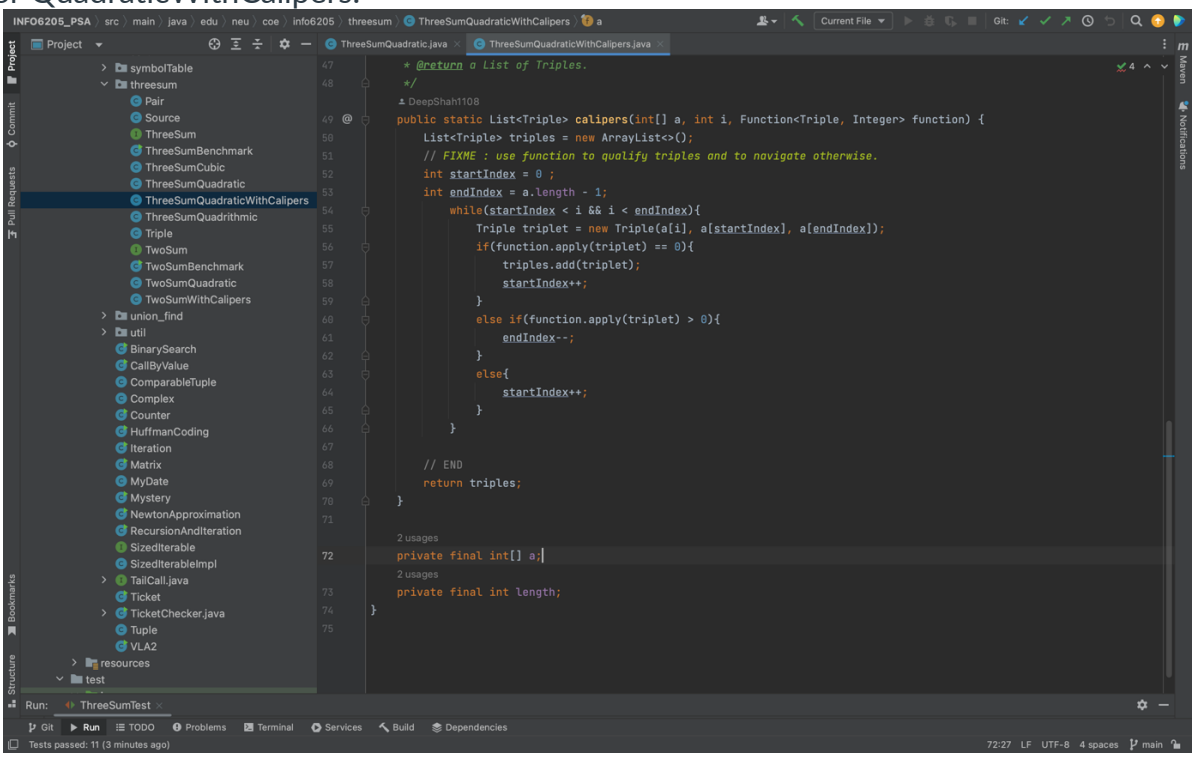
After solving the 3-SUM problem, there were few observations made for Quadratic, Quadrithmic and QuadraticWithCaplipers and created a table for the following data.

For Quadratic:



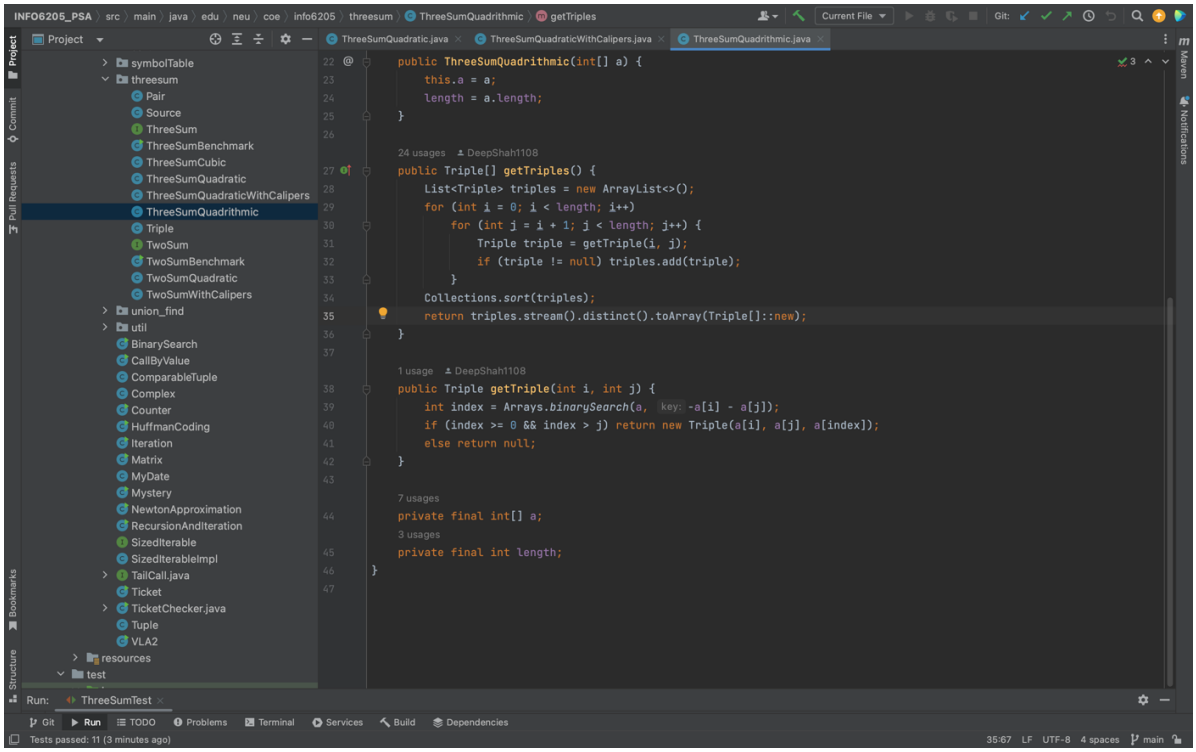
N	log N	Quadratic Time (mSecs)	Quadratic log T
250	5.52146092	1.04	0.039220713
500	6.2146081	1.78	0.576613364
1000	6.90775528	4.6	1.526056303
2000	7.60090246	18.1	2.895911938
4000	8.29404964	81.8	4.404277244
8000	8.98719682	402.33	5.997272647
16000	9.680344	1950	7.575584652

For QuadraticWithCalipers:



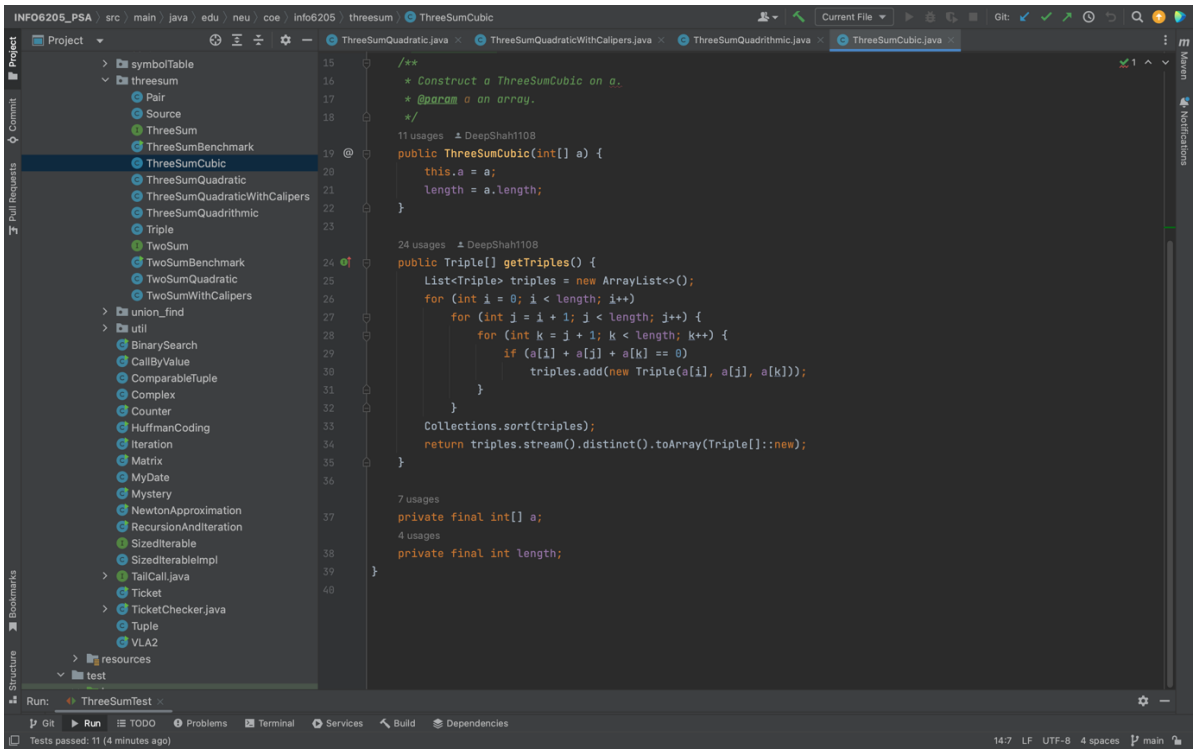
N	log N	QudraticWithCalipers Time (mSecs)	QudraticWithCalipers log T
250	5.52146092	0.6	-0.510825624
500	6.2146081	1.16	0.148420005
1000	6.90775528	3.8	1.335001067
2000	7.60090246	18.3	2.90690106
4000	8.29404964	86	4.454347296
8000	8.98719682	335.67	5.816128534
16000	9.680344	2024.5	7.613078035

For Quadrithmic:



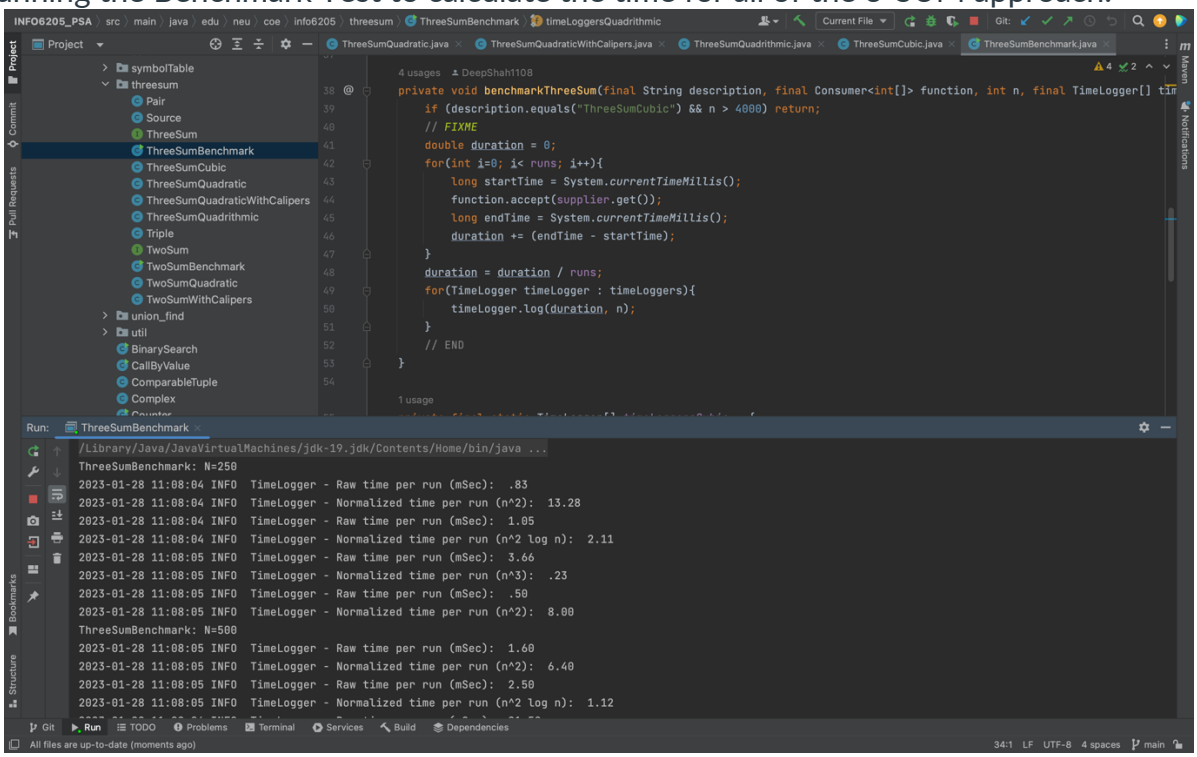
N	log N	Quadrimithic Time (mSecs)	Quadrimithic log T
250	5.52146092	1.34	0.292669614
500	6.2146081	2.84	1.043804052
1000	6.90775528	12.95	2.561095788
2000	7.60090246	69.4	4.239886868
4000	8.29404964	321	5.771441123
8000	8.98719682	1374	7.225481473
16000	9.680344	5917	8.685584843

For Cubic:



N	log N	Cubic Time (mSecs)	Cubic log T
250	5.52146092	3.97	1.37876609
500	6.2146081	24.68	3.2059932
1000	6.90775528	188.55	5.23936322
2000	7.60090246	1416.5	7.25594432
4000	8.29404964	10670.4	9.27522883
8000	8.98719682		
16000	9.680344		

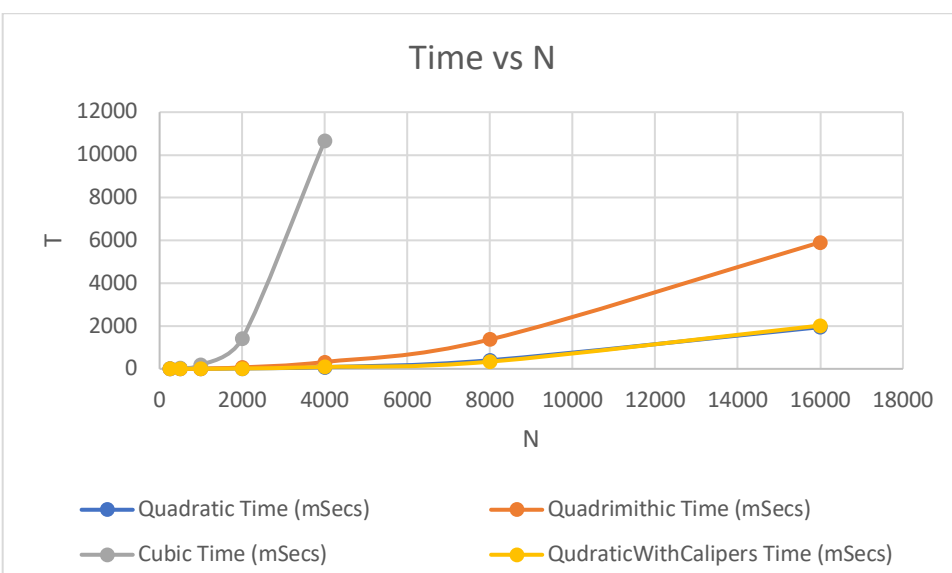
Running the Benchmark Test to calculate the time for all of the 3-SUM approach.



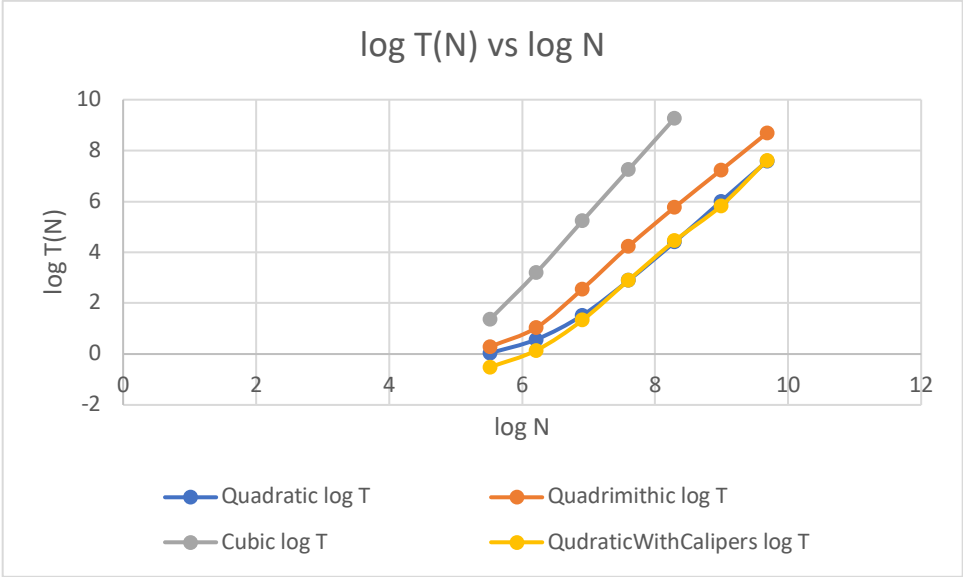
Graphical Representation:

As per the observations made, created the detailed excel sheet providing graphical representation of the data.

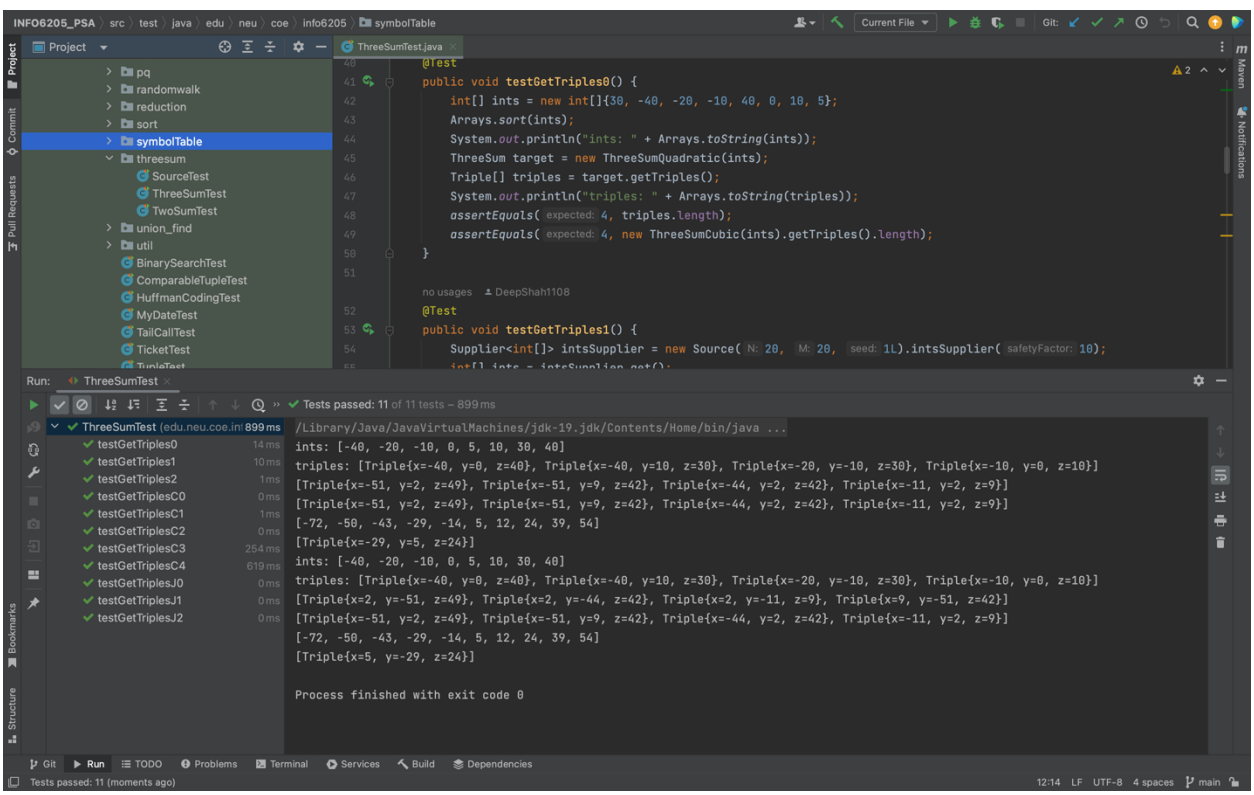
1. Graph between N(Size) and Time



2. Graph between Natural log(N) and Natural log(T(N)



Unit Tests Result:



Explanation of why Quadratics work:

Quadratic approach is the best as it is most efficient approach with a time complexity of $O(N^2)$. In this approach we use a two-pointer approach, such that we iterate through the array from both the start as well as from the end which takes a less time to finish the task.

To better understand it, let us consider an example.

Taking the array as {-40, -20, -10, 0, 5, 10, 30, 40}

So in this we have the middle element of the triplet of the 3-SUM.
Suppose the index of the middle element is 5. (j=5)

1. Initializing the start Index (i) as index of first element of the array and end Index (k) as index of last element of the array.
2. Iterating through the array we calculate:
   A[i] + A[j] + A[k] = 0
   If this true, then the value for the pair A[i], A[j], A[k] will be added to the triplet
3. If A[i] + A[j] + A[k] is greater than 0, we decrement the value of end Index (k)
4. If both steps 2 and 3 are false, we then increment the value of start Index (i).

In this we iterate the above steps until the array is fully processed.

Hence Quadratic approach is the best and the most efficient approach.