Program Structures and Algorithms

Spring 2023(SEC –8)

NAME: Junlong Qiao

NUID: 002784609

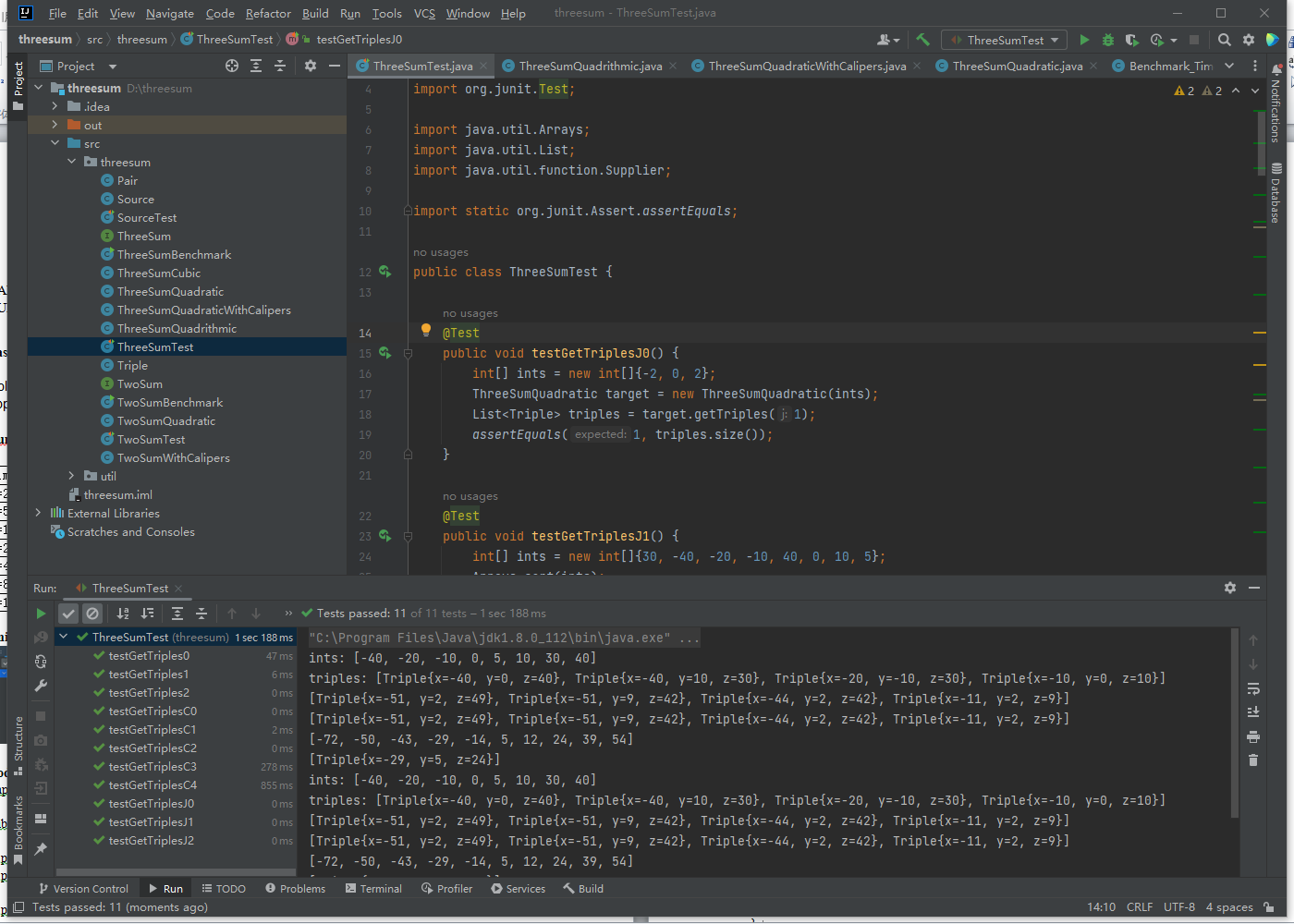
**Task: Assignment 2(3-Sum)**

Solve 3-SUM using the Quadrithmic, Quadratic, and (bonus point) quadraticWithCalipers approaches, as shown in skeleton code in the repository.

**Runtie Relationship Conclusion:**

|  |  |  |  |
| --- | --- | --- | --- |
| Time\Method | ThreeSumQuadratic | ThreeSumQuadrithmic | ThreeSumCubic |
| n=250 | 1.36 | 1.33 | 6.97 |
| n=500 | 1.7 | 4.58 | 52.07 |
| n=1000 | 6.88 | 19.77 | 409.51 |
| n=2000 | 29.2 | 85.4 | 3250.2 |
| n=4000 | 134.5 | 369.5 | 25945 |
| n=8000 | 692 | 1798.5 |  |
| n=16000 | 3752.5 | 8607 |  |

**Unit Test Screenshots:**



**Code of 3-Sum:**

**ThreeSumQuadratic.java**

public List<Triple> getTriples(int j) {

List<Triple> triples = new ArrayList<>();

int left = j-1;

int right = j+1;

while(left>=0&&right<length){

int sum = a[j] + a[left] + a[right];

if(sum==0){

Triple t = new Triple(a[left],a[j],a[right]);

triples.add(t);

left--;

right++;

}else if(sum>0){

left--;

}else{

right++;

}

}

// FIXME : for each candidate, test if a[i] + a[j] + a[k] = 0.

// END

return triples;

}

**ThreeSumQuadraticWithCalipers.java:**

public static List<Triple> calipers(int[] a, int i, Function<Triple, Integer> function) {

List<Triple> triples = new ArrayList<>();

if(a[i]>0){

return triples;

}

int right = a.length - 1;

int left = i + 1;

while(left<right){

int sum = a[i] + a[left] + a[right];

if(sum==0){

Triple t = new Triple(a[i],a[left],a[right]);

triples.add(t);

while(left<right&&(a[left]==a[left+1])){

left++;

}

while(left<right&&(a[right]==a[right-1])){

right--;

}

left++;

right--;

}else if(sum<0){

left++;

}else{

right--;

}

}

// FIXME : use function to qualify triples and to navigate otherwise.

// END

return triples;

}

**How the quadratic method work:**

At first the program use index I to traversal the whole array to set each of the number in array to be the middle value of the 3-Sum. In each loop, the program set the index of the smallest value(called s) start from i-1, the index of the biggest value(called b) start from i+1.If sum = nums[i]+s+b < 0, it means we should increase the sum, so we set the index of the b from i+1 to i+2. Instead, we should decrease the sum by set s from i-1 to i-2. If sum==0, we should record the combo and move both s and b. Repeat this step until the double traversal is complete.