Gajentaan & Overmars [3] used the following problem to show that other problems are probably not in subquadratic time.

Problem 0.1. 3SUM *INSTANCE: n integers. QUESTION: Do three of the integers sum to 0? NOTE: We consider any arithmetic operation to be unit cost.*

The following theorem gives better and better algorithms for 3SUM.

Theorem 1.

1. 3SUM *can be solved in $O(n^3)$ time.*
2. 3SUM *can be solved in $O(n^2 \log n)$ time.*
3. 3SUM *can be solved in randomized $O(n^2)$ time.*
4. 3SUM *can be solved in deterministic $O(n^2)$ time.*

Proof. Let A be the original input of n integers.
1) The trivial algorithms suffice to get $O(n^3)$ time: check all $O(n^3)$ 3-sets of A to see if any of them sum to 0.

2) First compute all the pairwise sums of A and sort them into an array B. This takes $O(n^2 \log n)$ steps. Then, for each element of A, use binary search to see if its negation is in B. This takes $O(n \log n)$ steps. If you find a negation in B then the answer is YES, otherwise NO.

3) Let p be a prime close to n^2. We will have a hash table of size p. The number z will go into cell $z \pmod p$ of the hash table.

For all $1 \le i < j \le n$ put $-(x_i + x_j)$ (along with (x_i, x_j)) into the hash table. This takes $O(n^2)$ steps. Then, for each element of $x \in A$ hash it into the table. See if there is at least one pair already there. If there is then with high probability there are $O(1)$ paris there. See if any of the sums in that entry of the hash table, sum with x to 0. If so then output YES and halt. If for no $x \in A$ do you get a YES then output NO. For each x, With high probability every x will be involved with $O(1)$ checks, so the expected run time is $O(n^2)$.

4) First sort A. This takes $O(n \log n)$ steps. Place a pointer at both the front and the end of A. Then, for each $x \in A$, do the following: If the sum of the integers at the two pointers and x is smaller than 0, then move the first array's pointer forward; if the sum is larger than 0, then move the second array's point backwards; otherwise, we have the three integers sum to 0, and we output YES and are done. If the two pointers crossover, then move onto the next integer in A. This algorithm clearly takes $O(n^2)$ time. □

Exercise 1. *Code up all four algorithms in Theorem 1. Run them on data and see which ones do well when.*

Is there an algorithm for 3SUM that runs in time better than $O(n^2)$? This depends on your definition of "better". The following are known:

1. If the integers are in $[-u, u]$ then 3SUM can be solved in $O(n + u \log n)$ time. We leave this an an exercise.

2. Baran et al. [1] showed the following: Assume the word-RAM model which can manipulate $\log n$-bit words in constant time. Then there is a randomized algorithm for 3SUM that takes time

$$O\left(\frac{n^2(\log\log n)^2}{(\log n)^2}\right).$$

3. Gronlund & Pettie [4] have shown that there is a randomized algorithm for 3SUM that takes time

$$O\left(\frac{n^2 \log\log n}{\log n}\right)$$

and a deterministic algorithm that takes time

$$O\left(\frac{n^2(\log\log n)^{2/3}}{(\log n)^{2/3}}\right).$$

4. Chan [2] has shown there is a deterministic algorithm for 3SUM that runs in time

$$O\left(\frac{n^2(\log\log n)^{O(1)}}{\log^2(n)}\right).$$

5. Gronlund & Pettie [4] have also shown that *there exists* a decision tree algorithm that had depth (so time) $O(n^{1.5}\sqrt{\log n})$ for 3SUM. Their proof does not show how to actually construct the decision tree in subquadratic time.

Exercise 2. *Show that if the integers are in* $[-u, u]$ *then* 3SUM *can be solved in* $O(n + u \log n)$ *time.*

While the algorithms above are impressive and very clever none are that much better than $O(n^2)$. We need a definition for "much better than $O(n^2)$".

Definition 1. *An algorithm is* ***subquadratic*** *if there exists* $\epsilon > 0$ *such that it runs in time* $O(n^{2-\epsilon})$.

Despite enormous effort nobody has obtained a subquadratic algorithm for 3SUM. Hence the conjecture is that there is no such algorithm.

ADDED LATER IN RESPONSE TO COMMENT: There is also no randomized algorithm for 3SUM. Perhaps the conjecture should be modified to also exclude that possibility.

References

[1] I. Baran, E. D. Demaine, and M. Patrascu. Subquadratic algorithms for 3sum. *Algorithmica*, 50(4):584–596, 2008.
`https://doi.org/10.1007/s00453-007-9036-3`.

[2] T. M. Chan. More logarithmic-factor speedups for 3sum, (median, +)-convolution, and some geometric 3sum-hard problems. *ACM Trans. Algorithms*, 16(1):7:1–7:23, 2020.
`https://doi.org/10.1145/3363541`.

[3] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 45(4):140–152, 2012.
https://doi.org/10.1016/j.comgeo.2011.11.006.

[4] A. Grønlund and S. Pettie. Threesomes, degenerates, and love triangles. *Journal of the Association of Computing Machinery (JACM)*, 65(4):22:1–22:25, 2018.
https://doi.org/10.1145/3185378.