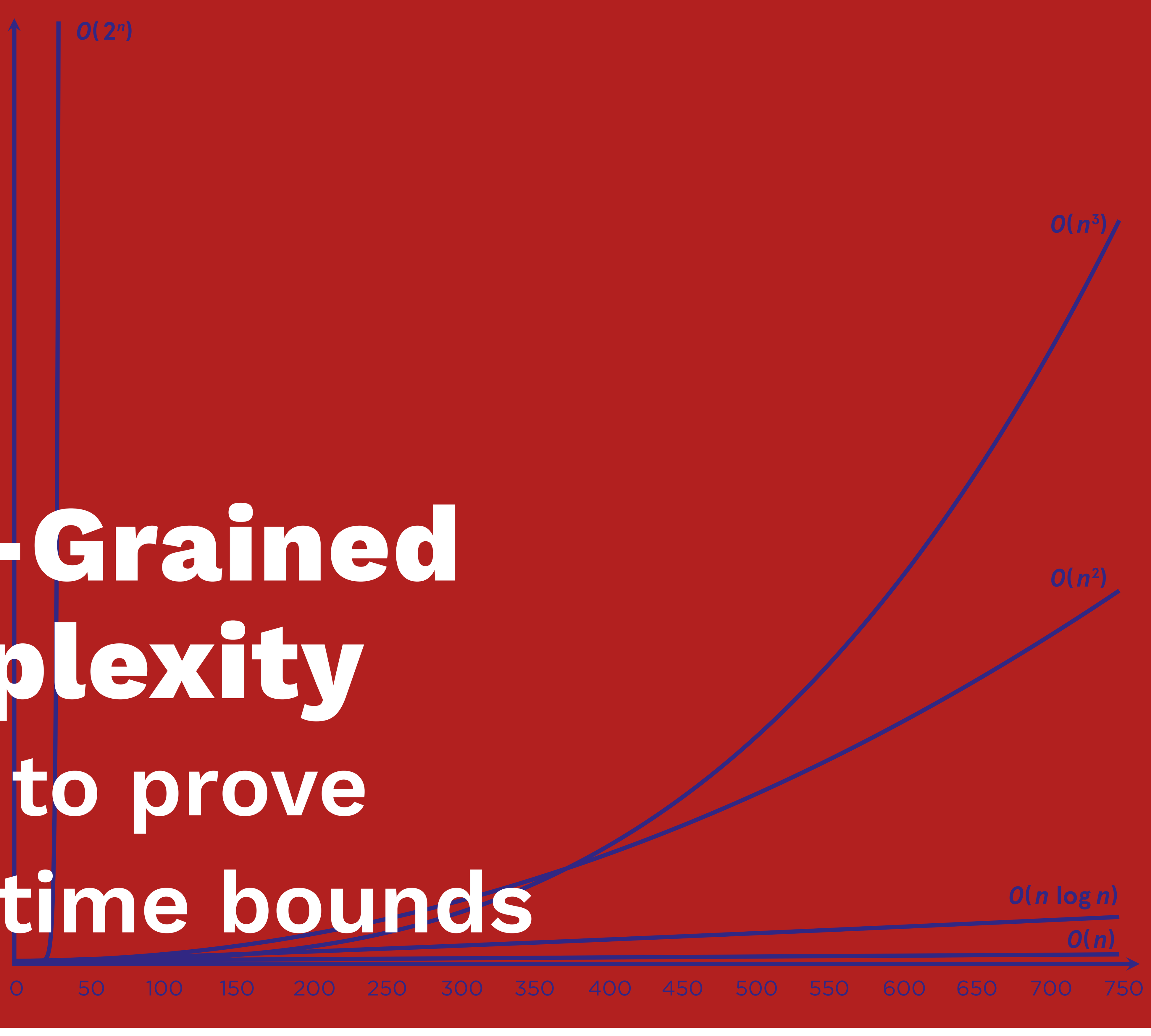
# Fine-Grained Complexity

## A way to prove exact time bounds




For any computational problem, the two most important factors for designing an algorithm are its efficiency and optimality. However, one of the major challenges in complexity theory has been the inability to prove unconditional time lower bounds. Nevertheless, we would like to provide evidence that say a problem A with a running time $T(n)$ that is not been improved in decades, also requires $T(n)^{1-o(1)}$ time, thus explaining the lack of progress on the problem. Unfortunately, such unconditional time lower bounds seem very difficult to obtain. Towards that, the area of fine-grained complexity has been developed.

### WHAT IS FINE-GRAINED COMPLEXITY THEORY?

Fine-Grained complexity theory is based on fine-grained reductions that focus on the exact running times for computational problems. The techniques mimic the idea of proving NP-hardness for problems, except that in the latter case we don't care about the exact hardness.

Over decades, using fine-grained reductions, many meaningful relationships between problems in the classical setting have been made. More recently, similar connections gave been explored in the quantum setting as well.

### THE APPROACH

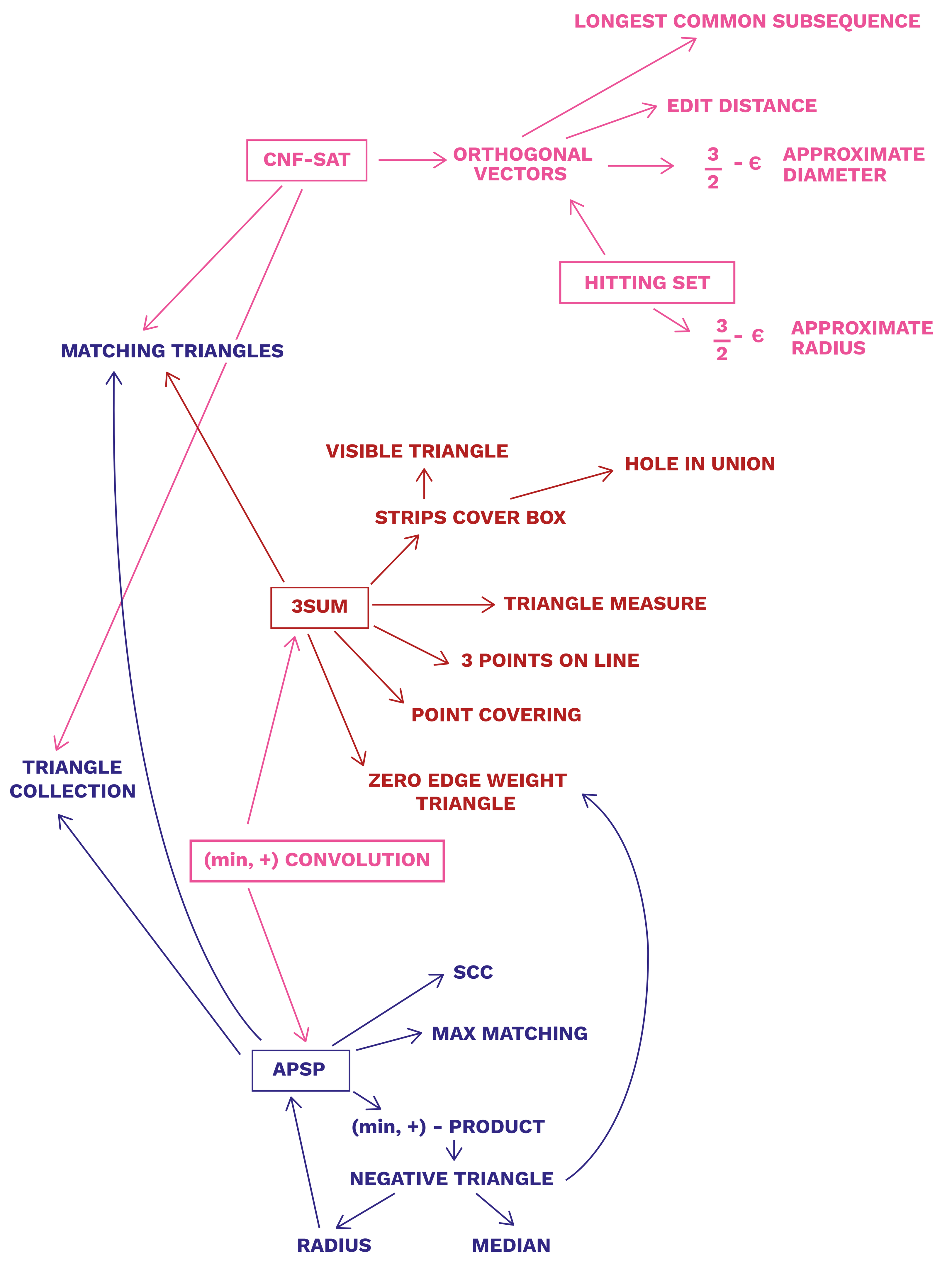
The approach is:

**1 —** To select a key problem X that for some function T, is conjectured to not be solvable by any $O(T(n)^{1-\varepsilon})$ time algorithm for $\varepsilon > 0$, and

**2 —** To reduce X in a fine-grained way to many important problems, thus giving (mostly) tight conditional time lower bounds for them.

Some of the key problems for example are the CNF-SAT problem, the 3-SUM problem, and the All Pairs Shortest Paths Problem (APSP).

### SOME KEY PROBLEMS AND THEIR FINED-GRAINED REDUCTIONS




#### GLOSSARY

**CNF-SAT**
Given a Boolean formula in its conjunctive normal form on n variables, is there an assignment to these variables such that the formula evaluates to true.

**3-SUM**
Given a list of n integers, is there a triple a, b, c in the list such that a + b + c = 0?

**All Pairs Shortest Path**
Given a graph of n nodes with weighted edges, output the shortest path between all the pairs of nodes in the graph.