The program is implemented based on the possible position of each ship, which means that variables are ship, and domain values are board coordinates. From the beginning, the **constraints are already been reduced**, since the number and length of each ship are fixed and ships will be placed on the board as a whole, which avoids checking the constraints such as LL or LMT, or if there exceed a given number of a ship on the final board.

Secondly, some preprocessing methods are applied to the initial board, such as columns/rows that have constraints with a value of zero will be assigned to water and forbidden to be occupied as a ship, as well as considering the side coordinates of the given ship segments on the initial board have to surrounded by water. For example, the surrounding of an initial “L” should all be water except the box on the right. The domain values of each ship are generated after preprocessing the initial board. For each ship, the data structure keeps track of the size of the ship as well as the domain values. The positions in which each ship can be placed only keep track of the coordinate values where “L” or “T” are landed, which decreases the complexity and space. If any ship are already been fixed in the initial board, that ship will be directly assigned to the final value, such as one size=1 ship with “S” at position (1,2).

Except for the submarine that has already been fixed in place. The longest ship in size often has the least options to be placed on the board, since the remaining open spaces constrain it. These ships with fewer domain values will be assigned before the other ships based on the **minimum remaining variable heuristic.** Also, in the algorithm, when deciding the following variable to assign, it will be based on the minimum remaining variable heuristic since it picks the variables that are most likely to cause a failure soon, thereby pruning the search tree.

Also, then **ordering the constraints** is important to reduce the calculation time. For example, after each ship has been placed on the board, the boxes occupied by that ship and surrounding boxes cannot be occupied by other ships. These coordinates will eliminate other unassigned ship domain values and we will check this first. Then, check and calculate if the remaining domain of other unassigned ships will exceed the row/col constraint or not. It decreases the calculation time by eliminating the domain first by the forbidden boxes first.

Furthermore, some necessary other methods are included to fasten the algorithm by checking and eliminating the domain values before running it. For example, if the initial board has “L” at position (1,2). The final board should give an output that has at least one ship at that position with an “L” on it. However, it has its possibility to have LR, LMR or LMMR on that position, which falls into the domain values of multiple variables. Some methods have to be applied. After assigning a ship, it will check if the current assigned ship list has these initial “LRTBM” constraints satisfied. If not, then among all the unsigned ships, there should have at least one ship to satisfy each unsatisfied constraint. Otherwise, the current just assigned ship is invalid. These necessary processes fasten the algorithm to a large extent.