# Gogen Puzzle

Below is a **Gogen** **puzzle**.

The letters A-Y must appear **once** each in such a way that all the listed words can be spelt out. Words must be spelt by following the lines between letters, like in the game Boggle.

Do **not** solve the puzzle. Instead, think about how you would write a program which could solve **any** Gogen puzzle. The solution should be conceptual and no code is expected, but feel free to use concepts from languages with which you are familiar. Focus should be on data storage, procedural programming and perhaps performance, rather than on user interface.



Steps to Solve the Gogen Puzzle:

1. Split Letters in word list up into pairs so we can test those pairs appear in the result (Letter Pair List)
   1. BOW will become (B,O) and (O,W)
2. Create a list of all the letters and the locations where they could possibly go (Candidate List)
   1. At the start this will be the crossjoin of the letters A-Y and the positions (0,0) through (4,4)
   2. E.g. (‘A’,0,0) , (‘A’,0,1) ……… (‘Y’,4,3),(‘Y’,4,4)
3. Rule out all the letters we are given to start with. Any letter with a fixed position should be stored separately
   1. We can keep a separate list with the locations of letters we are 100% sure of (Solved List)
4. Iterate over the next few ideas for lowering the possibilities:
   1. If a letter pair appears in our Letter Pair List and one of those pair is solved we can rule out any candidates for the other not adjacent to the solved letter
   2. For each entry in the Letter Pair List we have for a certain letter, the final position of that letter must have that many neighbours.
      1. We can rule out edge spaces for most popular letters. For example O has to pair (W,X,L,A,B,D) and thus cannot be at the edge as it needs 6 neighbours
      2. Any Solved letters next to our candidate position that are not paired with the candidate letter reduce the possible neighbours
5. After each candidate reduction we need to do the following:
   1. If a Letter has only 1 candidate position, it is solved
   2. If a position had only 1 candidate Letter, it is solved
   3. If a candidate is occupying the position of a solved letter it is discarded
   4. Once a letter has been solved all other candidates for that letter can be discarded
6. Finally we need to resolve more complex cases
   1. One option when the methods above have been exhausted is to pick one letter that has not been solved and mark it as solved at random. We would need to store the progress of the program and revert any changes should there be no success, discarding the candidate if so and picking another.
   2. There are likely algorithms for solving the disputed candidates, this is due to the fact it is possible to deduce the positions of more letters by hand, but it is hard to convert that logic into code.