This code first transforms the input into a graph whose maximum flow will be the maximum number of dominos that we can place in the original input. We then use the Ford-Fulkerson algorithm to compute the maximum flow of this graph and return the result.

For our transformation, we consider the board in which the dominos are to be placed to have a tiling scheme like a checkerboard, except with block-outs at squares where we have x’s. We note that any way that we could place a domino on such a board will have the domino touching exactly one “red” square and one “black” square.

Thus, we see that the solution to the first problem is identical to the problem, “what is the maximum number of dominos we can place on this checkerboard such that each domino touches exactly one red square and one black square and the domino does not rest on any block out squares” We see that to solve this, each domino must start on a black square and end on a red square, and we do not want to double count cases where that same domino starts on a red square and end on a black square. Thus, we can map this to the following graph:

1. The source node, s, has edges of weight 1 to all the possible black square-nodes that are not blocked in the input graph
2. Each black square node maps to any red square-nodes that are both adjacent to it and non-blocked in the original board. Each of these edges also has size of exactly 1
3. Each red square node has an edge of weight 1 connecting to the target

Note that the amount of flow leaving the source and entering into the black square-nodes must be the same as the flow leaving the red edges and entering the target. We see this because the source only connects to the black square-nodes and the target only has in-edges from the red square-nodes. Because edges have capacity of at most 1, we thus see that the number of black square-nodes with flow entering them must be same as the number of red square-nodes with flow leaving them. Because each activated red square-node can have at most 1 unit of flow leaving it, we see that by PHP, each red square-node must correspond to a single unique black square-node. Thus, each non-blocked black square-node with an edge activated in this graph must correspond one domino. Because each black-square nodes have at most one unit of flow entering it (all from the source), the flow to these activated black-square nodes must be the number of dominos.

Accordingly, the max-flow of our transformed graph would yield the correct number of dominos if our Ford-Fulkerson algorithm is correct, then our solution should be correct.

Our Ford-Fulkerson implementation is an implementation of max-flow for a sparse graph. We construct a sparse graph as a vector nodes, each containing a vector of connected edges. When we want to add an edge to a node in one direction and there is no corresponding edge in the opposite direction, we initialize a back-edge of weight 0 in the opposite direction for the other node. Just as we did in lecture, we add augmenting flows from the source to the target until we are no longer able to. When we are done, we return the sum of the flows that we created.