The algorithm used by the program is recursive and computes the set of matchstick graphs with *n* edges starting from the set of matchstick graphs with *n-1* edges.
The starting set is obviously the only matchstick graph with a single edge.

It can be split into three steps:

Step # 1: Generating graphs

For each matchstick graph with *n-1* edges, we do the following actions:

1. For each couple of nodes of the graph, if they are not already connected by an edge, add a new edge connecting these two nodes.

2. Add a new node in the graph, and connect this node to successively all other nodes of the graph.

This algorithm generates a set of **potential matchstick graphs** with *n* edges, which contains the set of matchstick graphs with *n* edges (as well as possibly other graphs which are not matchstick graphs).

Due to the algorithm used, these graphs respect the following properties:
- The graph is connected.
- Two nodes are connected by at most one edge.

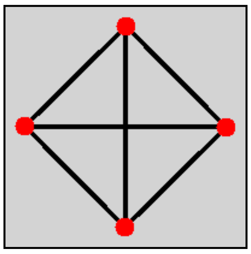
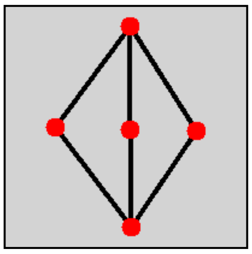
Step # 2: Removal of graphs which are not matchstick graphs

Graphs with 1 to 5 edges

Up to 5 edges inclusive, all graphs generated this way are matchstick graphs.

Graphs with 6 edges

With 6 edges, we meet the following cases:

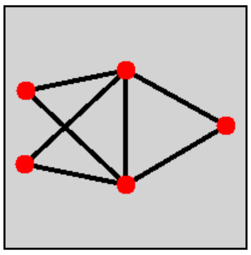
We're going to detect these cases and forbid them by introducing the following rules:

Rule # 1:	If two triangles have an edge in common, it is not possible to add an edge linking the two opposite nodes of the two triangles

Rule # 2:	Two nodes cannot be linked to the same three other nodes

Graphs with 7 edges

With 7 edges, the problematic case is the following:



It leads to the following rule:

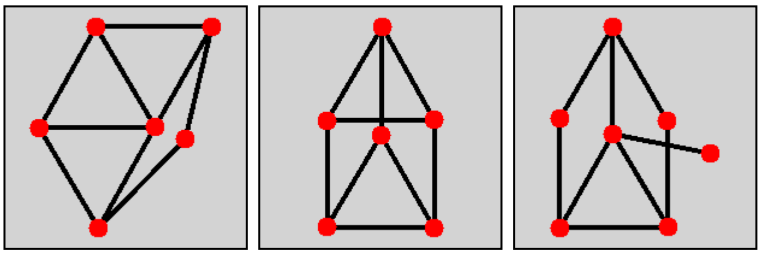
Rule # 3: An edge cannot be shared by more than two triangles

Graphs with 8 edges

By using the three rules above, all generated 8-edge graphs are matchstick graphs.

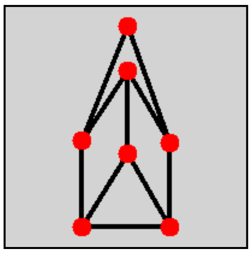
Graphs with 9 edges

Three cases are detected and rejected by the program:



Graphs with 10 edges

One case is detected by the program to reject it:



Step # 3: Detecting isomorphic graphs

The next step consists of detecting the isomorphic graphs to keep a single version.

For each potential graph generated in the previous step, we compare it to every potential graph previously generated.

The algorithm is quite brute force. It recursively tries to match each node of the first graph to a node of the second graph.

If it succeeds, we obtain a bijection between the nodes of the first graph and the nodes of the second graph so that two nodes of the first graph are linked by an edge if and only their images are also linked by an edge.

If it fails, we have the guarantee that the two graphs are not isomorphic.