OSMO

The constraints in the task are small enough that we can check, for each position, each of the eight directions and each word, if the word starts at the chosen position and in the chosen direction. A separate matrix helps us keep track of the elements used.

Testing each of the eight directions can be done with eight blocks of code, but this approach allows copy-paste errors to creep in. A safer approach uses the same block of code with variable offsets in the x and y directions.

The overall time complexity is O(N^2*letters), where letters is the sum of lengths of all of the strings.

WACLAW

There are two types of triangles: those facing upwards (ending with the number 4) and those facing downwards (ending with numbers 1, 2 or 3). Clearly a upwards-facing triangle can share sides only with downwards-facing ones. Additionally, a triangle can only share a side with at most 3 triangles, all of which are of the same size or larger.

If a triangle is facing downwards (ending with 4) then it is surrounded by three upwards-facing triangles and cannot be leaning on any other triangles. Thus, the only triangles a downwards-facing triangle $A4$ is leaning on are $A1$, $A2$ and $A3$.

If the given triangle is facing upwards then it may be leaning on up to three other triangles. Consider the case when it is of the form $A1$. If it is leaning on some triangle to its upper left, then that triangle shares a side with the smallest parent triangle of $A1$ ending in 3. A similar argument can be made for the other cases.

LAMPICE

If the first row consists of A groups of consecutive ones and the second consists of B groups of consecutive ones, then we can surely solve the problem using A+B moves that change the states of lights in the same row.

Changing the state of a column is a single move so we should only use it when it decreases both A and B by one. Each such change saves one move.

SIBICE

For each subset of all possible squares we can calculate the number of matches required to build all those squares. Since there are only 14 possible squares we can recursively examine all such subsets. Using bitmasks to keep track of which square requires which matches simplifies implementation.

Another caveat is that building some subset of squares may result in other squares being built inadvertently. For example, 4 small squares may form a fifth, larger one. This must be handled appropriately.