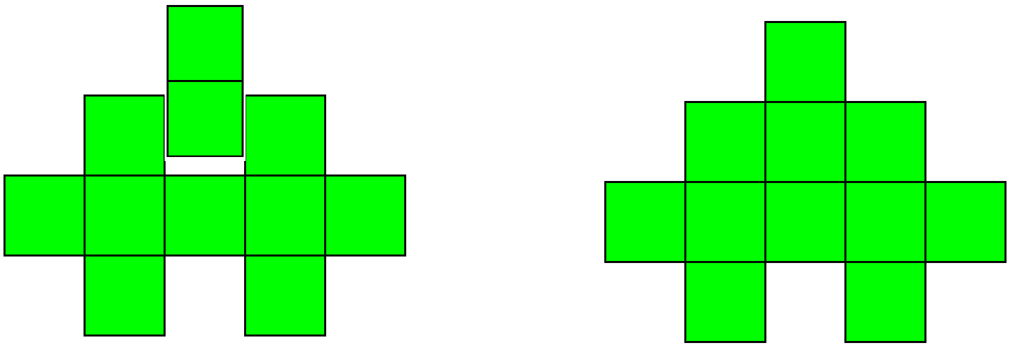
Counting unitary polyominoes without holes (A245660). Version 3, May 10, 2023.

Terminology

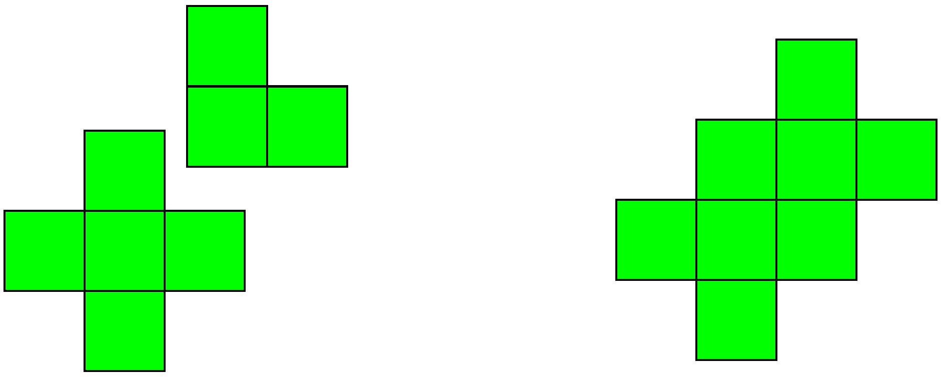
1. A unitary polyomino is one whose edges all have length 1 (https://sicherman.net/nunitary/nunitary.html)
2. Golomb played on the word unholey and so used the adjective “profane” for polyominoes without holes. So we can say that unitary polyominoes without holes are PUPs, profane unitary polyominoes.

We will first prove that any PUP bigger than a monimino (which is itself a PUP) can be built from a smaller PUP by one of the following construction methods:

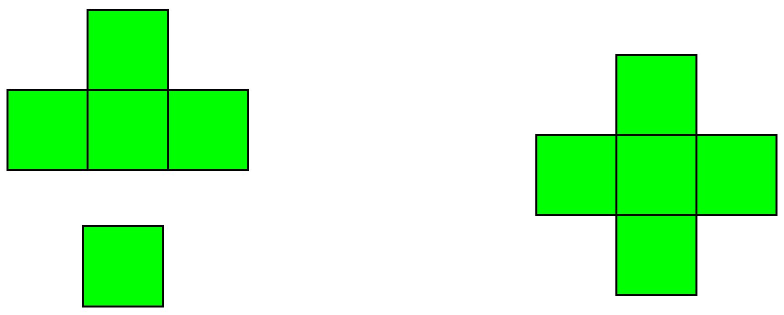
M2: add a domino to a cavity of size 1x1, taking care that it does not touch, even at a corner, any other square than those that surround the cavity:



M3: add an L-shaped tromino to an open corner of size 1x1, taking care that it does not touch, even at a corner, any other square than those that surround the corner:



M4: add a T-shaped tetromino to an exposed leaf of size 1x1, taking care that it does not touch, even at a corner, any other square than the leaf itself:



With this theorem in mind, it becomes possible to generate PUPs relatively efficiently by applying recursively the above construction methods. The principal inefficiency of the algorithm is that it becomes necessary to test for duplicates after each step.

The complete algorithm becomes:

1. From the size 1 PUP, generate the size 5 PUP
2. From the size n PUPs, generate size n+2, n+3, n+4 PUPs.
3. Once all PUPs of a certain size are generated, reflect and rotate to remove duplicates.

**Theorem**: any PUP, larger than the monomino, may be constructed from a smaller PUP by one of the above methods M2, M3, M4.

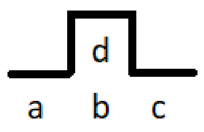
**Proof by contradiction**:

Suppose we have a PUP, P, of size > 1, that cannot be so constructed. We can therefore say that P has no removable leaf. A removable leaf is one such that it is the highest cell of one of the three forms used in the construction methods, such that the removal of said form would leave an intact PUP.

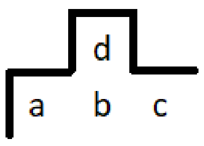
It is clear that any PUP of size > 1 must have at least one leaf on each "side" – that is its top and bottom rows and its leftmost and rightmost columns.

Consider then the leftmost top leaf of the PUP that has no removable leaves.

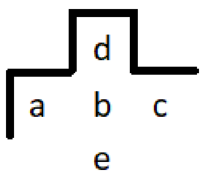
Draw what we know so far about this leaf: the black lines indicate the external boundary; the letters indicate cells that must exist:



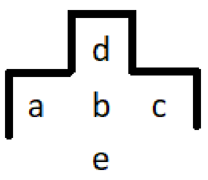
But we know that the leaf d is the leftmost top leaf so we can add a boundary:



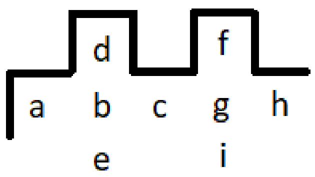
We also know that directly under b there cannot be a boundary, as the external cells of a unitary polyomino are all of the same colour if chessboard colouring is applied. Therefore:



Considering what happens to the right of c, we can take two forks, B1 and B2:

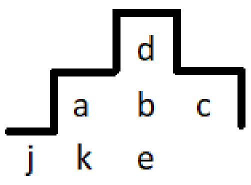


B1

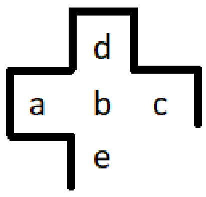


B2

Take the fork B1, and consider what happens under a:

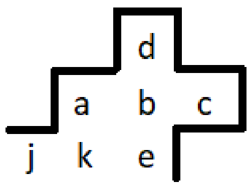


B11

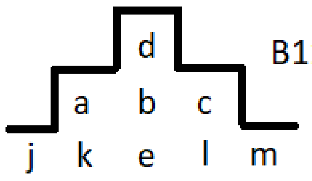


B12

Take the fork B11 and consider what happens under c:



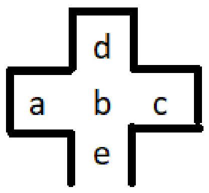
B111



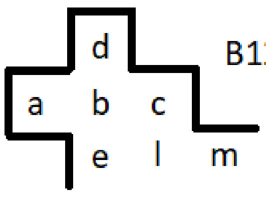
B112

But in B111, the d leaf is removable, as part of the tromino dbc, and in B112, the d leaf is removable, as part of the domino db. This contradicts the initial assumption that there is no removable leaf, so we know that the fork B11 is impossible.

Consider the fork B12 and what happens under c:



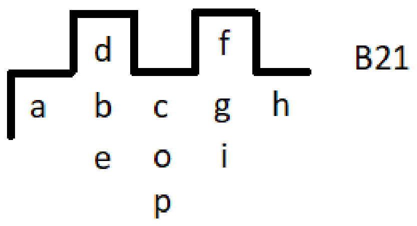
B121

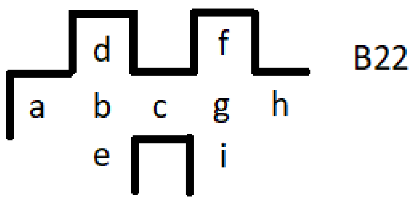


B122

But in B121, the d leaf is removable, as part of the tetromino dabc, and in B122, the d leaf is removable, as part of the tromino dab. This contradicts the initial assumption that there is no removable leaf, so we know that the fork B12 is impossible, as is the fork B1.

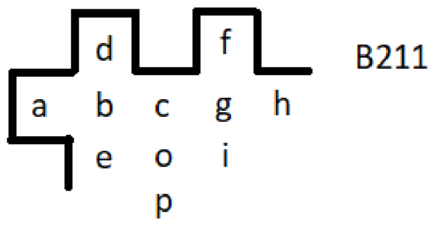
There remains the fork B2. Consider what happens under c:

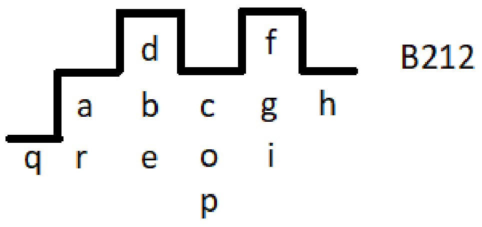




(In B21, there cannot be a boundary below the o because o would be of the wrong colour.)

Consider the fork B21 and what happens under a:

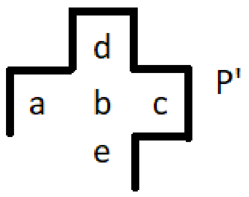




But in B211, the d leaf is removable, as part of the tromino dab, and in B212, the d leaf is removable, as part of the domino db. This contradicts the initial assumption that there is no removable leaf, so we know that the fork B21 is impossible.

There remains only B22, which necessitates more work.

Remember that P has no removable leaves. Form P’ by removing from P all that lies to the right of c.



Clearly P’ is a PUP, and has no removable leaves except possibly c. Rotate P’ clockwise 90 degrees, and consider its leftmost top leaf. Clearly all the previous considerations apply, and the top leaf must correspond to d in B22. Apply the symbols a’, b’, c’, etc to each of the cells.

Dividing P’ at the cell c’ will form two polyominoes, one of which will contain c. Assume that this is the right hand polyomino; if it is not, we can reflect P’ in the vertical axis so that it is.

Form P’’ by removing from P’ all that lies to the right of c’. Clearly P’’ is a PUP, and has no removable leaves except possibile c’. The original c is not a problem as we have eliminated it.

We can now repeat the above process an infinite number of times, which is of course impossible for a finite polyomino. Therefore we have eliminated B22, and so B2, and so we have shown that the original assumption leads to a contradiction.

**Conclusion**.

With this algorithm it has been possible to extend A245660 to over 40 terms. Unfortunately, the same mechanism does not seem to apply to A245620 – unitary polyominoes with or without holes.

**Post Scriptum**.

1. I thank Craig Knecht for pointing out a couple of typo's.
2. Joerg Arndt suggested performing the count by creating necklaces. I tried this and obtained a significant, though not game-changing performance improvement. However, maybe someone else with a fresh approach could make a much better algorithm.

The necklace approach.

Follow the external border of a PUP in the clockwise direction. Put a black bead on the necklace for each right turn, and a white bead for each left turn.

The construction methods discussed above become string substitutions in the necklace. Perform these string substitutions recursively starting from the smallest necklace (BBBB) and so generate all possible PUPs.

M2: substitute BWWB with WBBW.

M3: substitute BWB with WBBWBBW.

M4: substitute BB with WBBWBBWBBW.

In the algorithm that was implemented:

1. After each substitution, it becomes necessary to check that the resulting necklace still represents a PUP. That is, there is no intersection between the new edges and existing edges.
2. Any necklace is considered equivalent to the necklace produced by inverting the sequence of beads.
3. Different sequences of Mx may generate the same necklace. It is therefore necessary to test for uniqueness.

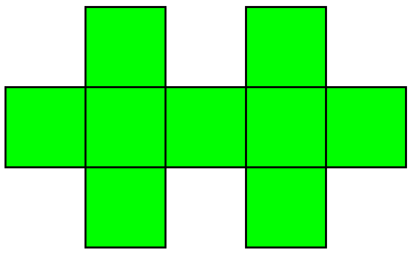
With this new approach, it was possible to reach a(50) (and confirm the previous results!)

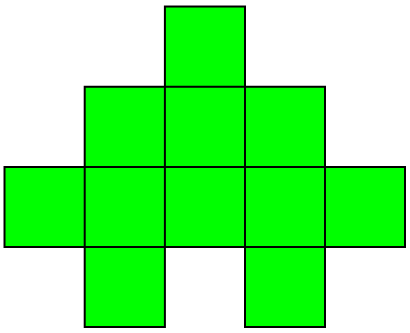
**Question**.

The following 2 PUP's have necklaces of the same length: BBWB**BWWB**BWBBWBBWWBBW and BBWB**WBBW**BWBBWBBWWBBW. The before and after parts of M2 are shown in bold.

The necklaces are the same length, and clearly the number of black beads in each is the same, as is the number of white beads (it is always true that num(B) = num(W) + 4). Yet the two PUPs have an area difference of 2.

Is there an easy way, just by looking at the sequences of beads, to calculate the area of the PUP?





**Answer (revised in V3)**.

Perform recursively the inverse string substitutions discussed above, counting 2, 3, or 4 according to the construction method M2, M3 or M4 that has been undone. When the result is the BBBB necklace, stop and add 1. The result is the area of the PUP.

This procedure will often work but I cannot guarantee that it will always work. Inverting M3 must be avoided if necklace length <= 12. Inverting M2 may damage the necklace so that it no longer corresponds to an intact polyomino; it should therefore be used only if the other inversions cannot be applied.

As George Sicherman pointed out to me, a much faster way of calculating the area is to use the Surveyor’s method.

John Mason, 10 May 2023