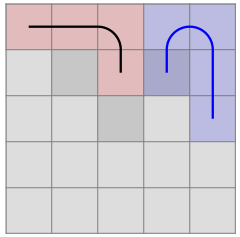
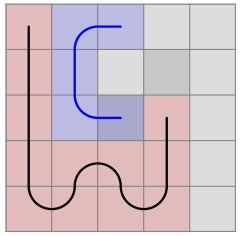
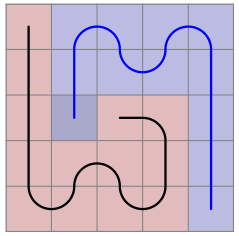
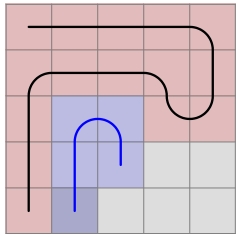


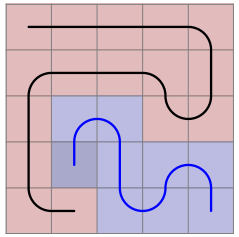
- A 2 x 2 empty area next to the live end that is walled by three sides (2-2-2 long) will have a future line going through along the walls. In this example, the far end is already extended by one step as it had only one option to move.

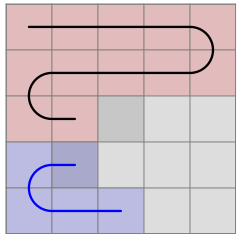




- Future line extension when we step on a future line: The far can be extended if it was 2 distance away from the near end. It can now fill the C-shape.





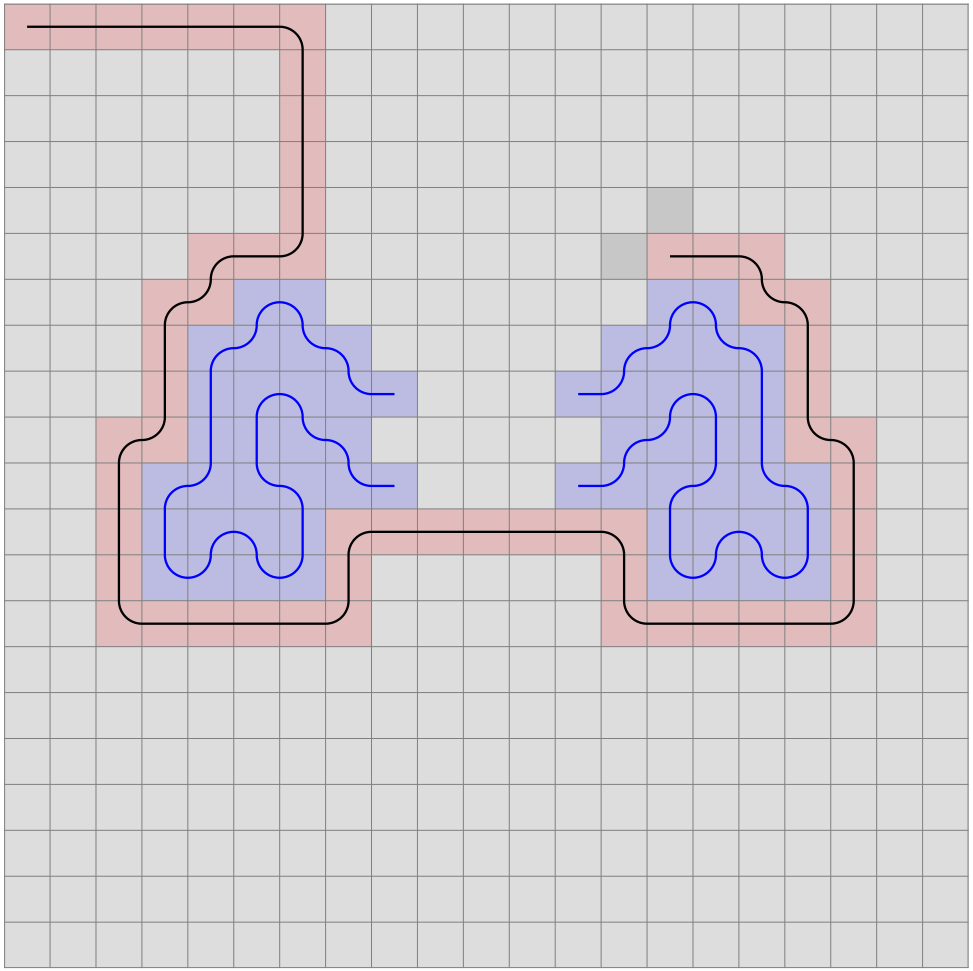


The same goes with 1 x- and y-distance. A C-Shape is not always created in this case.

The one-way labyrinth algorithm

This research aims to solve the following problem:
"Draw a line that goes through an n x n grid (where n is an odd number), passing through each field once. The line has to start from the field at the upper left corner (1,1) and end at (n,n). At any time it is allowed to move left, right, up or down, and it has to randomly choose between the available fields."

At first sight it may look easy. But look at the following example:

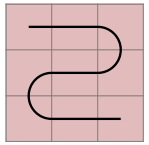


Based on the black line's movement, blue fragments were drawn to indicate a path we have to go through in the future in order to fill the board.
Do you see why the situation is impossible from now on?

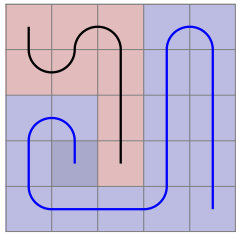
The question is, is there a single rule or a set of rules that will guarantee you can draw a labyrinth of any size? Or do the rules get infinitely complex?

To assist with the research, I have written a computer program. In the beginning, I let it run on a 21 x 21 field, and whenever I noticed a trouble, I coded the solution into it. While you can discover many patterns this way, they will be random and do not help in gaining a fundemental understanding. At one point you will find things get too complex, and you are still far from solving the 21 x 21 board.
That's where a gradual approach comes in.

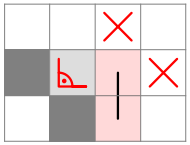
A 3 x 3 area can only be filled in two ways, like this and mirrored:

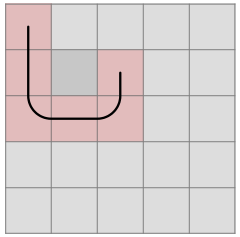


The 5 x 5 requires much more consideration. Whenever it is possible to draw future lines, the program has to be able to do it. The future lines can not only extend at each step but connect too.

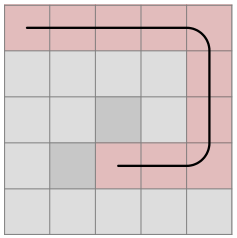


By August 21, 2023 all 5 x 5 scenarios were discovered. The number of walkthroughs are 104.
Here are the things to consider on a grid of this size:

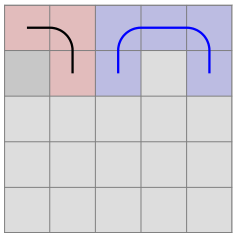




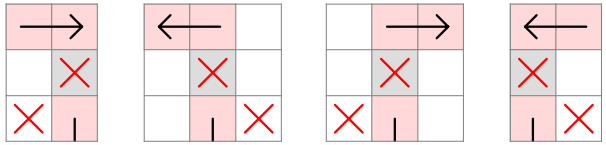
- A single field next to the live end that is walled from two other sides (either by the border or the line) needs to be filled in the next step. I call it C-shape. The pattern is both mirrored and rotated, so that the empty field is straight ahead. To qualify for this rule, the empty field cannot be the end corner. If there is a C-shape, we don't need to check other rules.



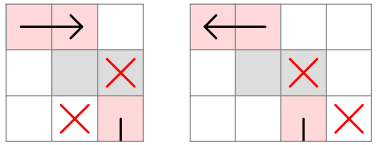
- Movement near the edge: In the example, we cannot step left (3,5), since the (2,5) field is empty.



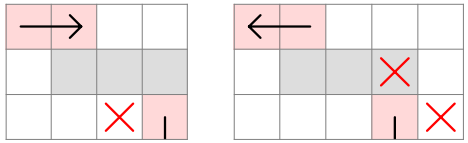
- A 2 x 3 empty area next to the live end that is walled by three sides (2-3-2 long) will have a future line going through along the walls. At the wall next to the main line, its direction is the opposite of the main line, meaning it will go from (3,2) upwards whereas the main line just took a step downwards. How the middle field will be filled is not yet known. Either the near end (the one the main line will go through first) or the far end can fill it.



The gray square means empty field. When the field 2 to straight is taken, its left or right side will be taken too.

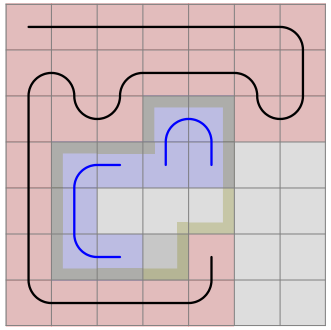


These will only be checked if one of the above 4 situations were not present. (They have to be mirrored, too.)

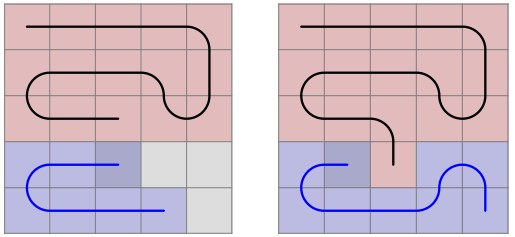


Likewise, these will be not be checked if the previous rules were true.

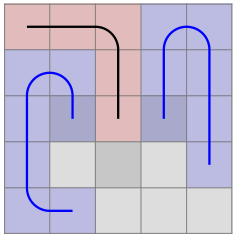
And when none of the 1-distance situations are valid, we check for 2-distance.



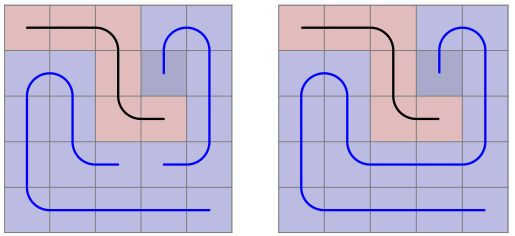
Impair areas can now happen inside the grid, not just on the edge, and the following rules have to be applied:



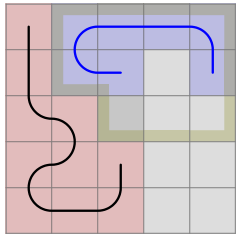
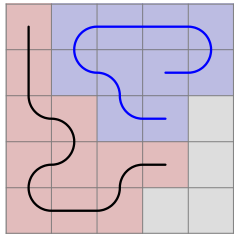
If the far end was near the end corner, it has to choose the other empty field.



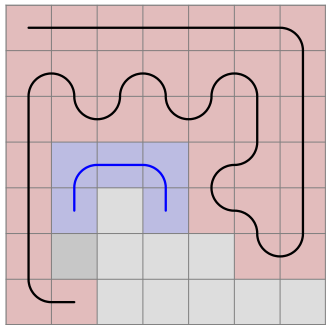
- Future line extension when stepping away: If there was a near end where the main line was in the previous step, it now may have only one choice to move, so it can be extended.



- Future line connection: In this case, the line being stepped on extends until the far end has two options. (When the end corner is one of them, it has to be removed.) Then, the line on the left extends and now has no other option than to connect to the line on the right.

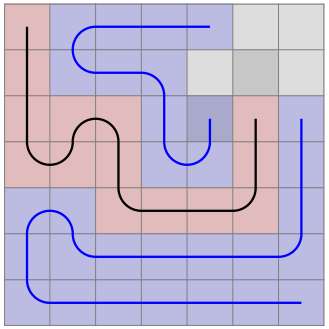
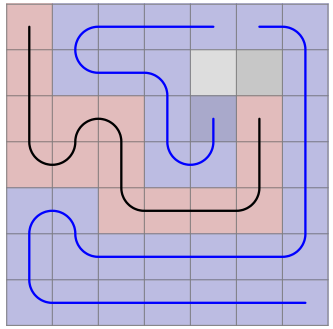



- When we are two distance away from the edge, we need to check if stepping towards it is possible.
It is because if we do so, an enclosed area is created, with one way to go out of it. If that area has an impair amount of cells, it cannot be filled, so we cannot take that step.
The explanation is simple: Imagine if the table was a chess board. In order to step from white to black, you would need to take an impair amount of steps - the color changes at every step. Here, the entry of the area would be (4,3) and the exit (5,3). An impair amount of steps means pair amount of cells.
In the example, you can also say that we cannot step right, because there is a future line start 2 to straight and an end 2 to straight and 2 to right. On 7 x 7, there will be examples where this is the rule we have to apply, because area counting is not getting triggered:

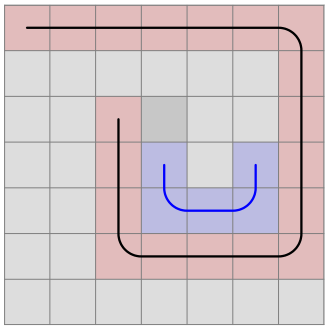


But let's start with the simpler rules:

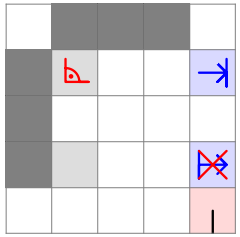
- Future line extension: When a near end is at 2 distance left or right from the live end, it will fill the field between them if the live end steps elsewhere. That's what happened in the 5 x 5 example above before the line failed.

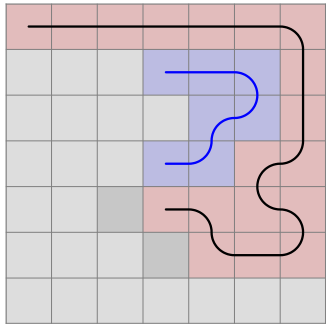



In other situations, there is a 1-thin future line next to the live end that can be extended if its far end is at the corner. Though disabling this rule does not affect the total amount of walkthroughs on a 7 x 7 grid, I chose to include it in the project on the basis that if a future line can be extended, we should do it. It can make a considerable difference. The left picture is without the rule, the right is with it.



- Just like moving near the edge, we need to disable some fields if we are approaching an older section of the main line. In order to determine on which side the enclosed area is created, we need to examine the direction of the line at the connection point.





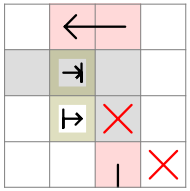
- And these are the remaining size-specific rules. Future 2 x 2 Start End, Future 2 x 3 Start End and Future 3 x 3 Start End.

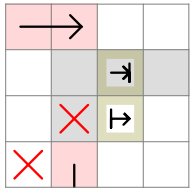
The program, in fast mode, can run through approximately 100 cases per second, depending on your computer speed. This enables us to discover all 7 x 7 walkthroughs, which is 111 712.
It is equal to what is described in the Online Encyclopedia of Integer Series (Number of simple Hamiltonian paths connecting opposite corners of a 2n+1 x 2n+1 grid).

As the sizes grow, it will be impossible to run through all cases with one computer in a reasonable time. In order to discover the patterns, we need to run the program randomly.

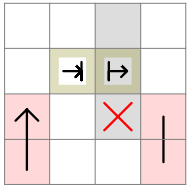
Is it possible to develop an algorythm that works for all sizes? The edge-related and area-counting rules are universal, but the size-specific rules get more and more complex. Can you define them with one statement?

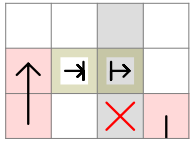
I have made statistics about how many random walkthroughs you can complete on different grids using the 7 x 7-specific and the universal rules before running into an error. Based on 1000 attempts, here are the results:
9: 19.5
11: 5.7
13: 2.6
15: 1.2
17: 0.7
19: 0.4
21: 0.2



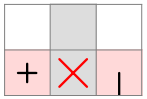


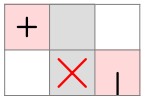
The procedure is similar to the the straight 2-distance rule. The only difference is that we count the area starting and ending at the marked fields. In the first, the direction of the circle is left, in the second right.
Besides mirroring them, we also have to rotate them both counter-clockwise and clockwise.
But we do not need 12 of such rules. Taking the first, the live end cannot come from the left, because the area parity was already checked in the previous step, and now we just added 2 fields to it. It can come from the right, and then there is naturally only one field we might have to disable.
Here are the representations of the two scenarios for the left side:

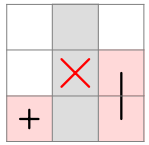




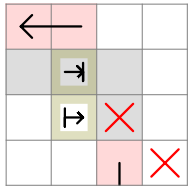
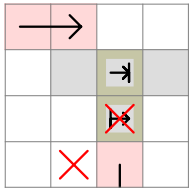
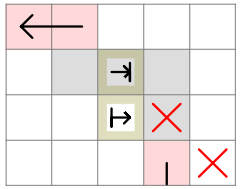
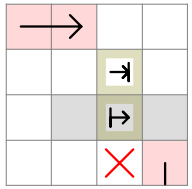
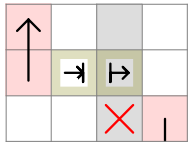
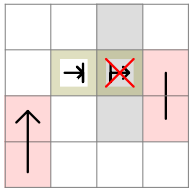
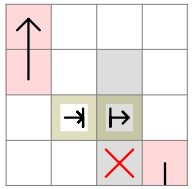
Similarly to the straight rules, these will only apply if there is no wall 2 distance to the left or right. Let's construct these preconditions.



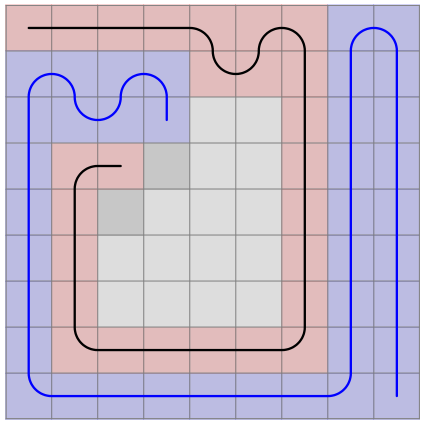
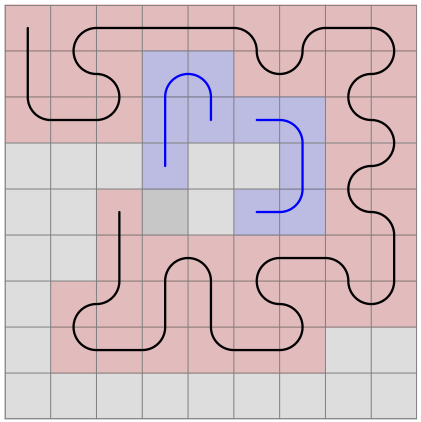




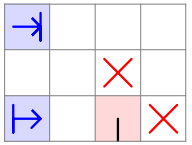
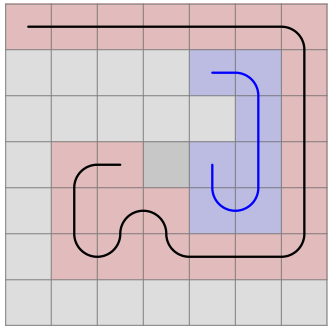
We are not finished. Did you notice the example above is not covered by these rules? We have to move the taken fields 1 and 2 steps to the side, both in straight and side direction.

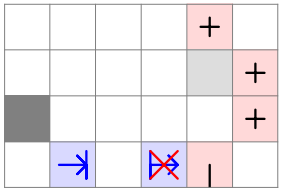
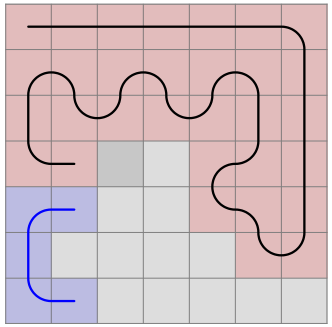
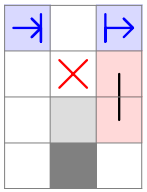
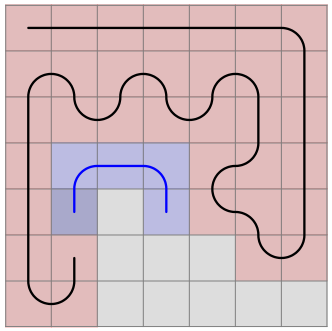
When any of the straight 2-distance rules are present, we don't need to check the side rules or the area created with the border. This is not entirely proven, but take these 9 x 9 examples:

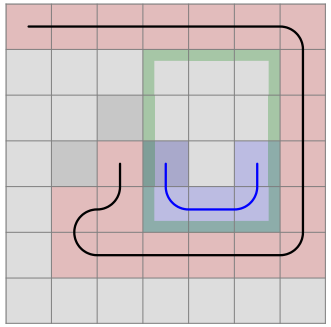



And these are the rest of the rules:

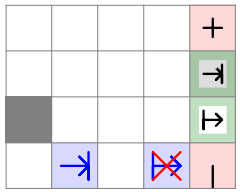
- This is what I started the 7 x 7 introduction with. I will call it Future L.



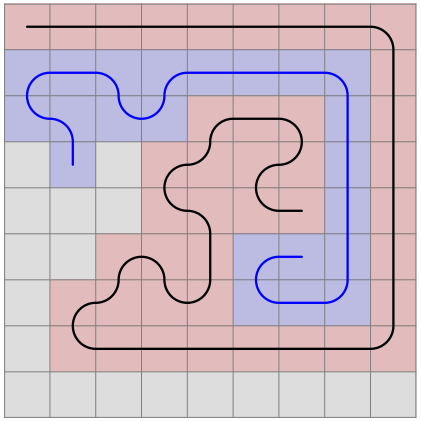
It would be a mistake to disable the right field.

So we need to check if an enclosed has been created on that side, but counting the area is unnecessary. Nevertheless, we can represent the rule this way, setting the circle direction to right:

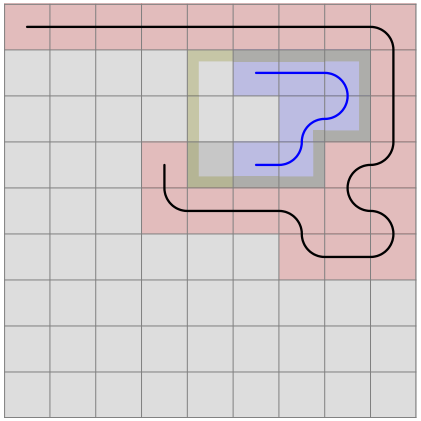


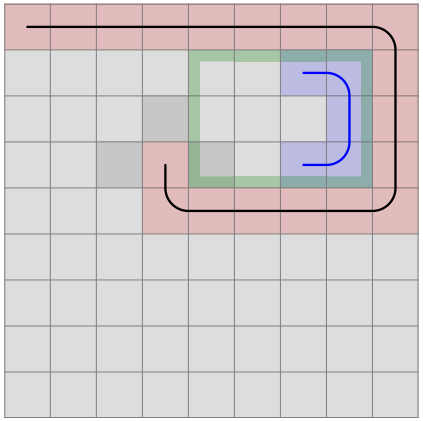
The code generator will examine if the count area start and end fields are 1 or 2 distance apart. In the first case, it will only determine in which direction the taken field straight ahead is going to, and if it is right, the forbidden field will take effect.
You may ask, why that field is "taken", not "taken or border". From what I found through some examples, if that field is border, the enclosed area on the right is impair, so the line cannot step in the other direction anyway. But it needs further examination.

To discover 9-specific patterns, I run the program keeping it left as long as the time to get to the first error is too big. After that, I will run it randomly. The first 13 826 walkthroughs are completed before we encounter a situation. It is similar to the last one we discovered on 7 x 7:

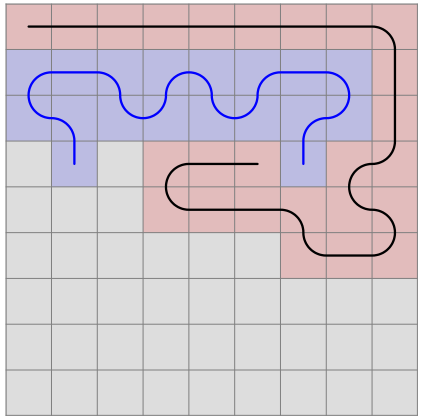


Let's simplify the pattern. Which will be impossible to fill?

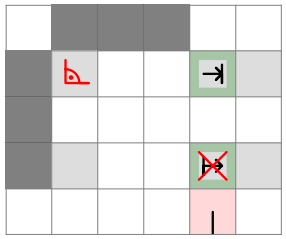




It is the picture on the left. Since the yellow-bordered area is impair, adding the (4,2) (4,3) (4,4) fields will be pair. We enter the area at (4,4), so we will exit at (4,3). Now we enter the 3 x 3 area in the top left corner at its side, (3,3) and will exit at (2,4). The results is two C-shapes on each side:



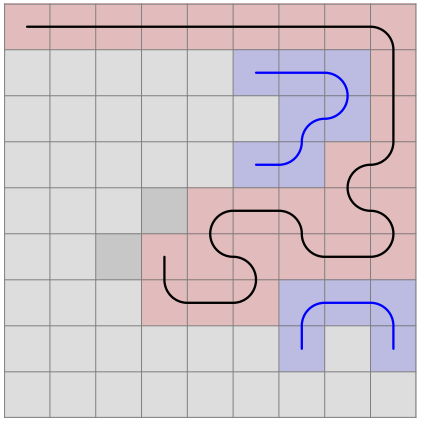
We can define a rule by marking the following fields and counting the area from the fields in front of the main line to the right:



Start_1 field is (4,3) and Start_2 field is (4,4) in the actual example. End field is (4,2). Direction of the circle: right (counter-clockwise). If the area is pair, we cannot step straight.

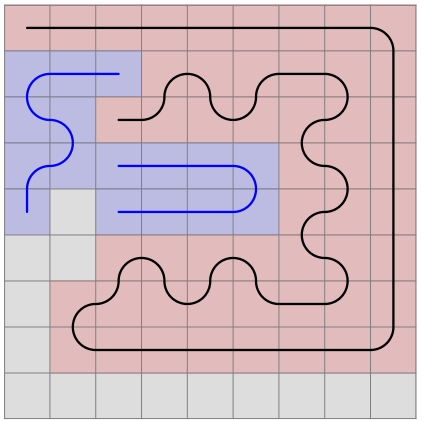
When generating code from the drawing, we have to check on which side the enclosed area was created. Here, we want it to be on the right side, so there are two cases to look at:
- The taken or border field beyond the end field is a taken field. In this case, if the field to its left is taken, its index must be lower. If the field to the right is taken, its index must be higher.
- It is the border. Add together the x- and y-coordinates to get a value. A higher value is closer to the end corner. Here, we compare the border field straight ahead and on its left, and we want the first-mentioned to be the smaller.

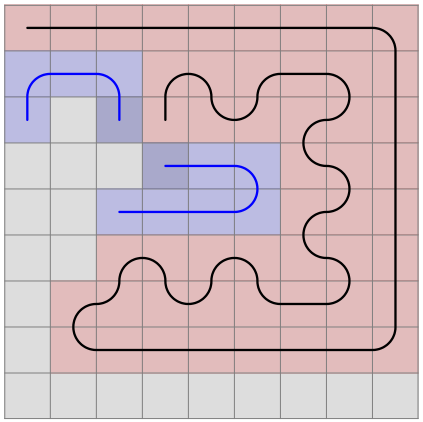
I have applied this rule rotated clockwise (besides mirroring it, of course), so that the live end can both come from the bottom and the right. But it can also come from the left in this example:



This will probably be another rule, because in this case it is not necessary to have an empty 3 x 3 field on the left.

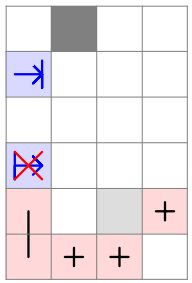
Now let's run the program further up to number 13 992:

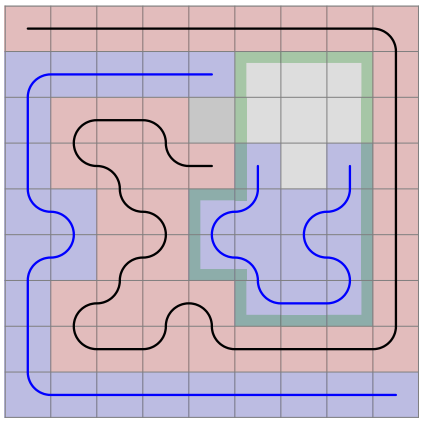




It is also just like the 7 x 7 rule, just with the extension of the area on the opposite side of the future line ends. But we can't simply remove the two taken fields on that side, because the line might continue in that direction, as it is the case here:

One certain situation reveals the incorrectness of the 7-rules when it comes to a 9-grid. In the following example, when I apply a rule rotated, it will disable a field that would otherwise be viable.

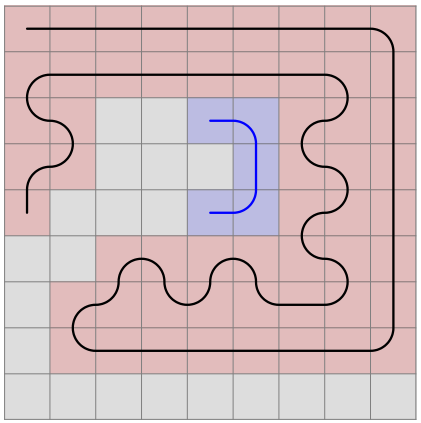




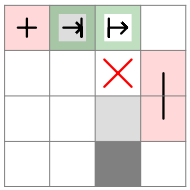
Rotating was not necessary to start with on 7 x 7, because no such situation occurred.

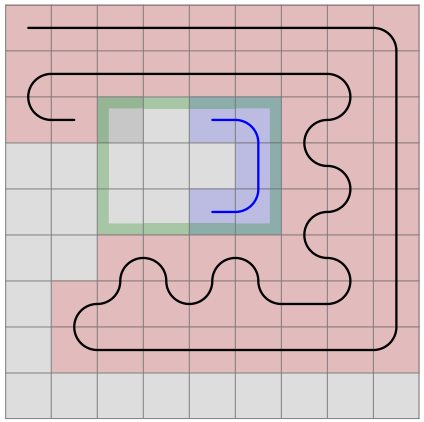
We can see that defining a rule with future line starts and ends does not tell us on which side the future line was created. That is the side that contains the enclosed area. We need to therefore replace such rules with area counting, which we actually already did, with the exception of Future L. Here the future line couldn't have been created on the other side, because that's the side the live end is at right now. And area counting is not always possible, like in this situation:

The next error, at 14 004 has something to with how I defined the universal rules of approaching an older section of the line, it needs to be reworked in light of the C-shape the main line can create with the border.

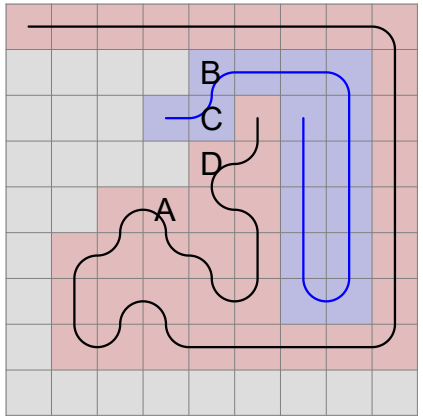


We need to take a few steps back, and then we can create the rule. It is similar to the universal 2-distance rule on the side, it just checks the field 2 behind and 1 to the side too. Even though the area counted is pair, now stepping to the right is disabled.

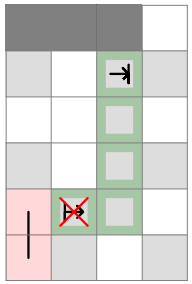


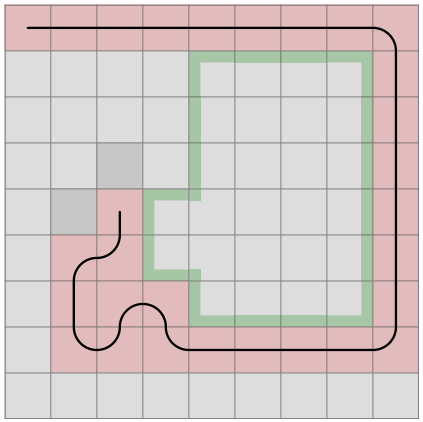


At 55 298, we get this:

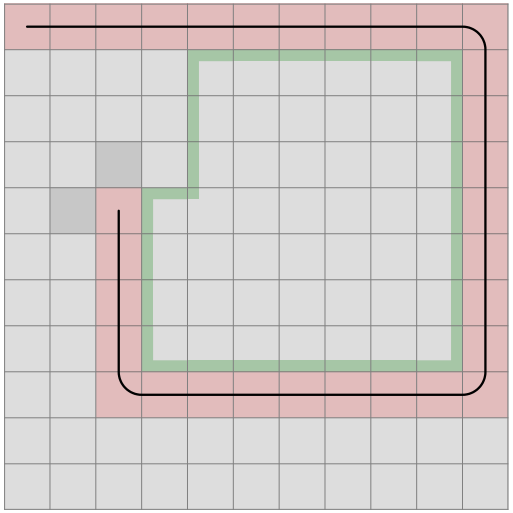


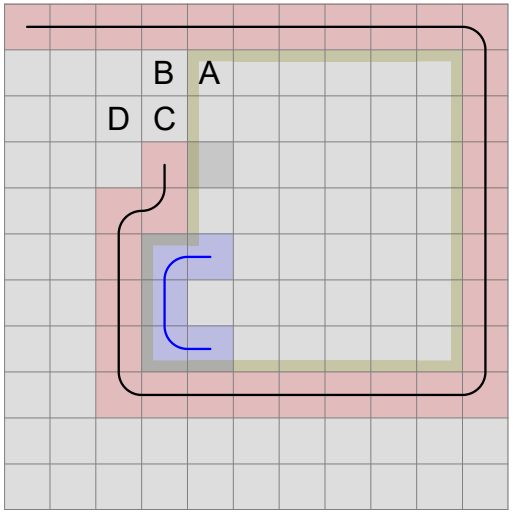
Let's analyze it! A double C-shape is created, because the line occupied the A field, and out of the B, C and D fields it exited the right-side area at C. It means, the area enclosed by the marked fields is pair. In this case, we shouldn't step right and the rule will therefore look like:





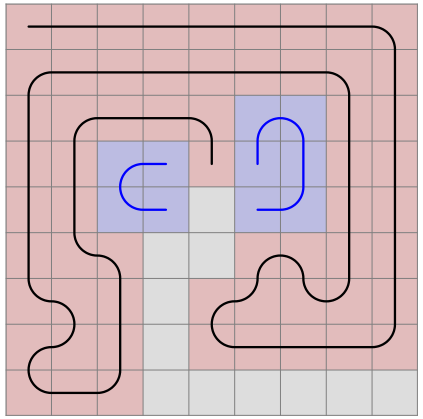
But what if from the A position, we step upwards in another situation? Compare these two on 11 x 11:

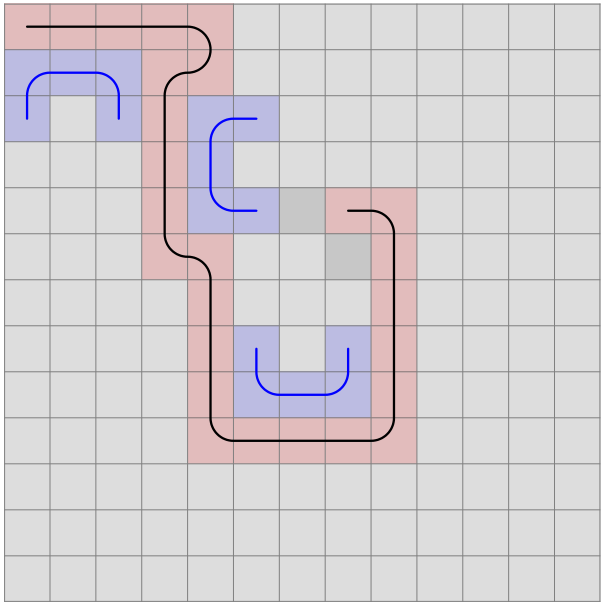




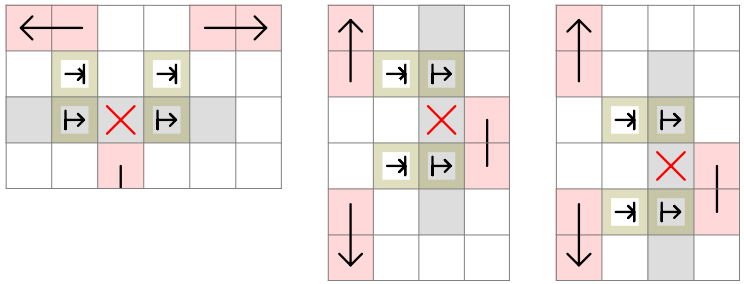
If the area we started with is pair, then the other will be impair. We can only enter the area at the light-gray field and will exit at A. From there we must go through B, C and D, and then a double C-shape is again created.

234 256 has at first sight something to do with future lines.



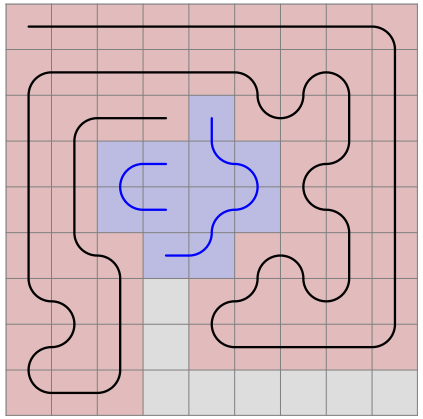


But it is more than that. Notice that enclosed areas has been created on both sides simultaneously. Because of the universal rules for approaching an older section of the line, now we have no option to move. The areas can be filled individually, but we cannot step to left and right at the same time.
We have to create 2-distance rules, which take both sides into account.

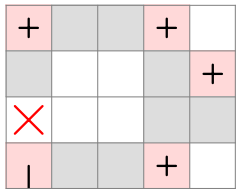


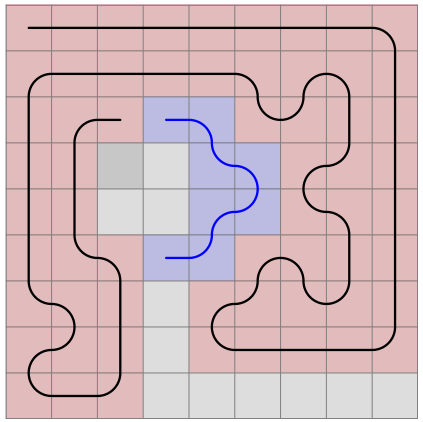
These are just a few of the possible combinations.
Any of the far straight rules (straight, mid across and across as I call them, depending on the horizontal distance of the obstacle) on the left side can be combined with any of those on the right side when the enclosed area is going to the same direction - left for left side and right for right side.
And the same is true when the pattern is rotated to the left or right side.
As far as porgramming concerned, it just needed a rework of the universal rules, we didn't need to make completely new ones.

As we run the program further, we will discover this at 227 200:

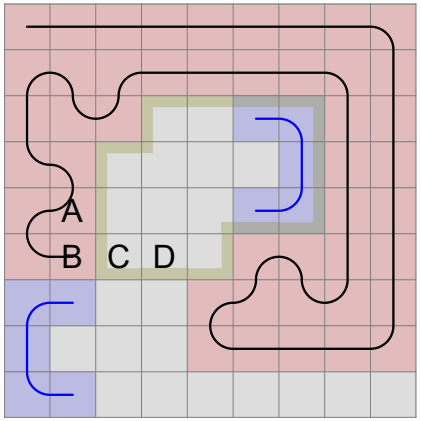


Intuitively, we can draw up the square, and let's mark the exit as well. There can be loops on the upper, lower and right side, they have no importance when tracing it back to the live end. There is only one way to go through.

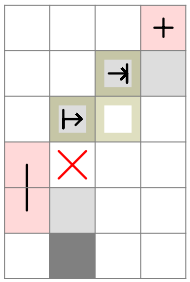


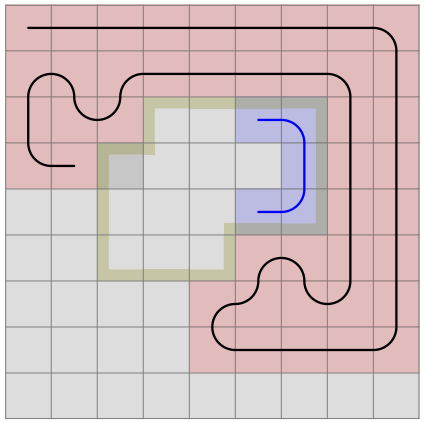


233 810 will look like:

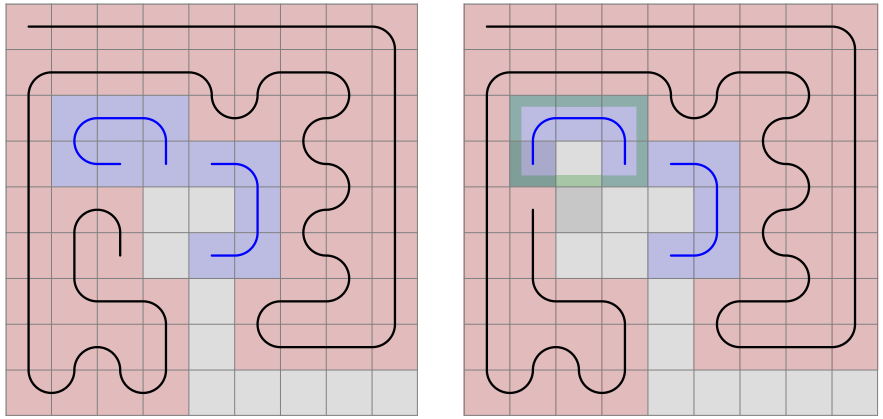


Once we step to A, it is unavoidable to get to B before entering the outlined area. It is because we can only reach B from the left or the bottom.
The area is impair, therefore we cannot complete it starting in C and ending in D.
As with many of the previous rules, the C-shape created with the border is to blame and therefore we need to represent it.

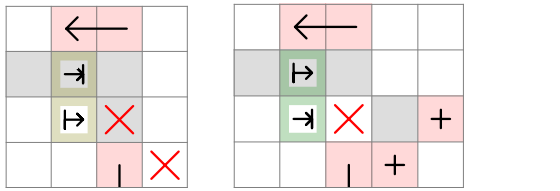


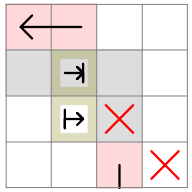


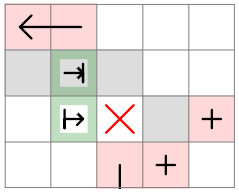
The same concept we encounter at 635 301, only the C-shape is created when we enter an area, on the other side of it.

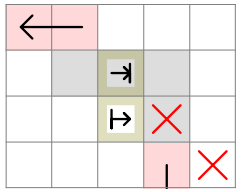


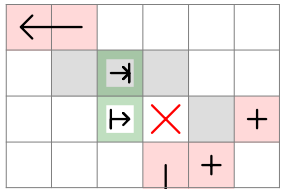
We have seen this in the third 9 x 9 rule. There the taken field next to the exit was in middle across position, and now it is across. And we also need to think about an obstacle straight ahead. Here are the original universal rules and their modifications.
Straight, circle direction left:



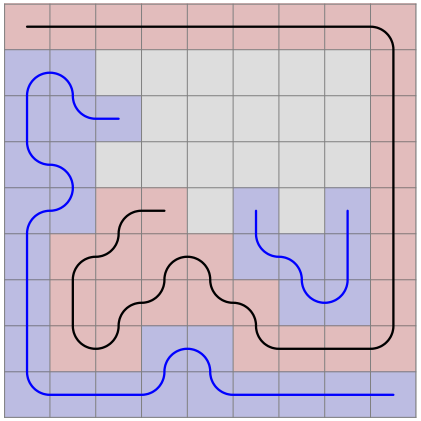




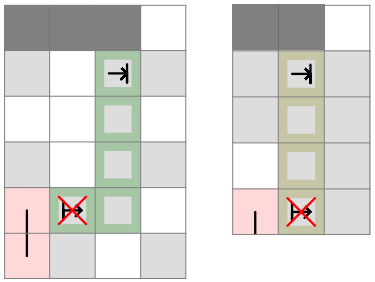




At 349 215, we find this:

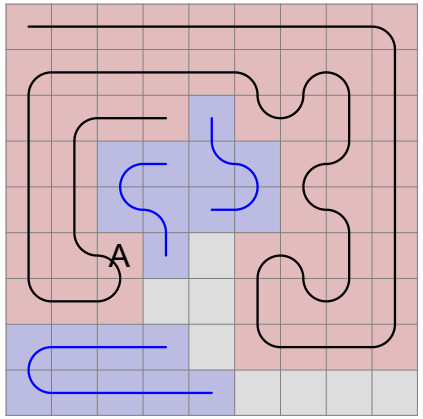


Though a double C-shape has been created in backwards direction, it indicates that the area on the right cannot be filled either.
We have made a similar rule previously. Now we need to simplify it.

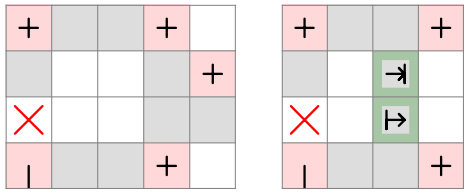


The area now has to be impair for the right direction to be forbidden.
Essentially, we just added the three extra fields to the pair area.

478 361 is similar to what we have seen before, only now there is a 2-wide path to exit the area:

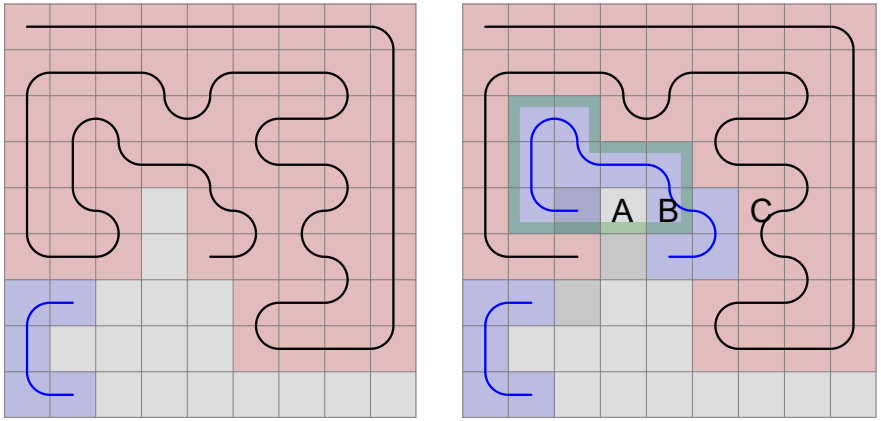


We have to mark where the area has been created in another way.

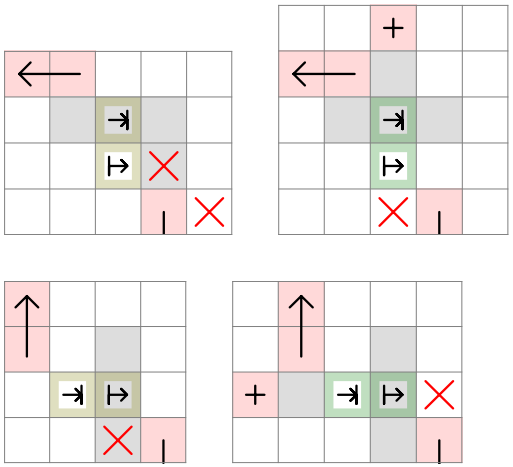


The taken field in the upper right corner is now checked for direction, but it is not enough. It can go upwards, and the exit of the area can still be on the bottom edge, just look at the example and imagine the live end was at A with the pattern already drawn. (On 11 x 11, it is possible to draw it.)
In order to establish an enclosed area, we must not encounter the bottom-right corner of the grid when walking along the edge of it.

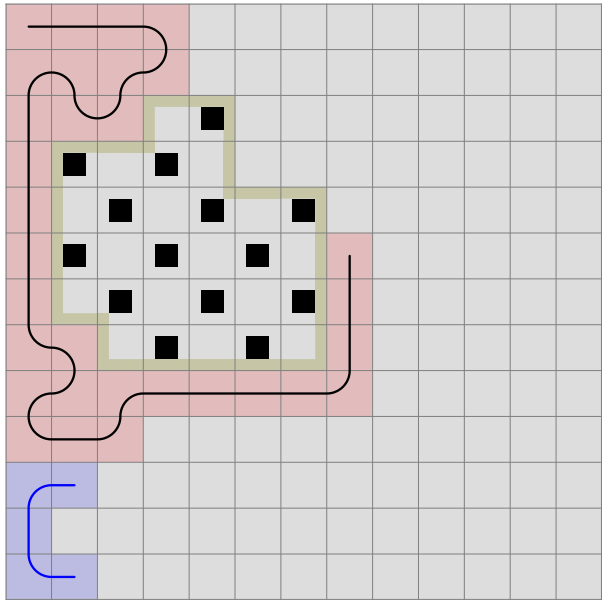
626 071 is:



With the marked area being pair, if we enter the area by stepping left, we will exit at A. But we can only get there from B; if we entered from the top, nothing would fill B, and we cannot enter and exit it after we left the area - subtracting 1 from the area would make it impair, so then we couldn't have exited at A.
The taken field C creates a C-shape, which we need to step into from B.
The universal far across rule have to be extended. By default, we disable the option to step straight or right if the counted area is impair. When it is pair, we need to disable the left field.

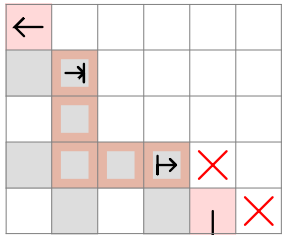


Let's mark the original example as a checkerboard.

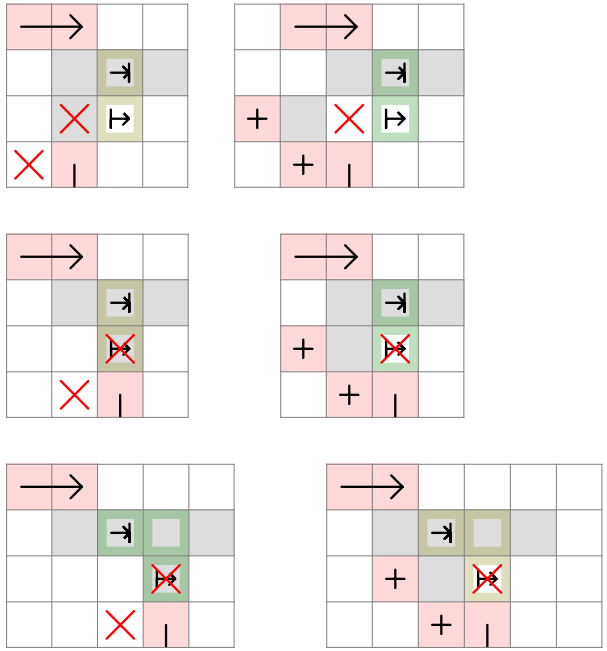


We enter at a black field and exit at black too, so the number of black fields should be one more than the number of white fields.
Here there are 14 black fields and 15 white. That's why the area cannot be filled. The up and right directions need to be disabled, so we can only step left.

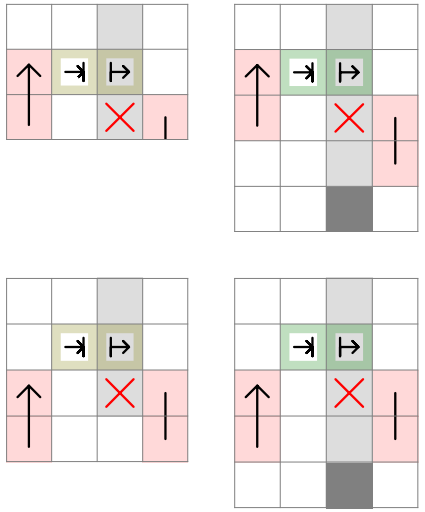
This is the rule representation. The reddish arealine now means the arealine is impair, and we know that the entry and exit points are the arealine start and end fields.

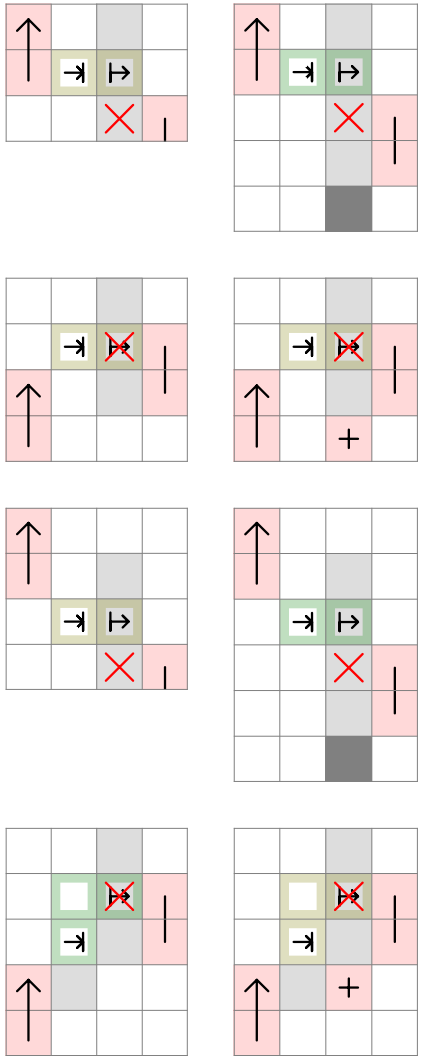


Circle direction right:



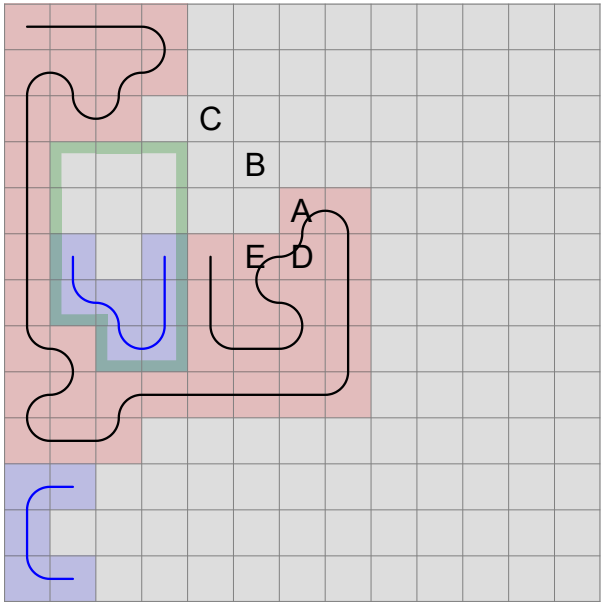
Side, with taken fields above and below:





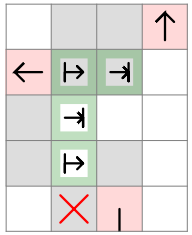
I have made some changes by adding some empty fields in side positions, so they are the same as the straight rules, just rotated.
Also, I have added the side across down rule and changed the straight across rule accordingly. Not only fields next to each other can define an area, they can be across too. In that case, if the area originally was marked impair, now it has to be pair.
Notice that in side rules, when the taken field that would create the C-shape is below the obstacle creating the area, it can be a border field too. We have seen an example of that previously.

Now what if both the start and end C-conditions are true? We can construct this on 13 x 13:

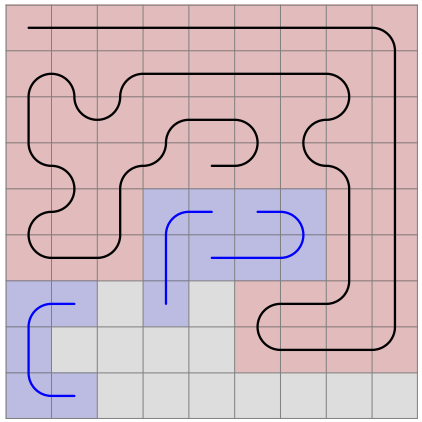


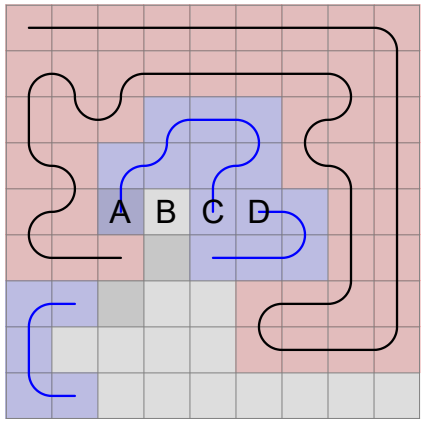
Several walkthrough attempts will leave you thinking why you cannot fill the area once obstacle responsible for the start C-shape is created (A). The area enclosed by A, B and C is pair. So when you enter it at A or B (obviously C is not a possibility), in order to exit at C, you need to leave out an impair amount of fields from the area. In case of entering at B, you cannot leave out A, but when you enter at A, you can leave out B, and no more than that. Now the area will be impair.
The minimal area would be stepping left from A, left again, up and up to get to C. You have covered 5 fields.
In order to make a walkthroughable area, you would need to extend it by pairs of fields next to each other, like D and E. One will be filled at a pair amount of steps, the other at an impair amount.

The rule will now look like this:

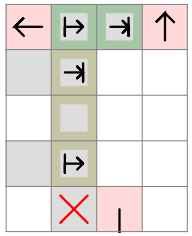


740 039 is a slight modification.

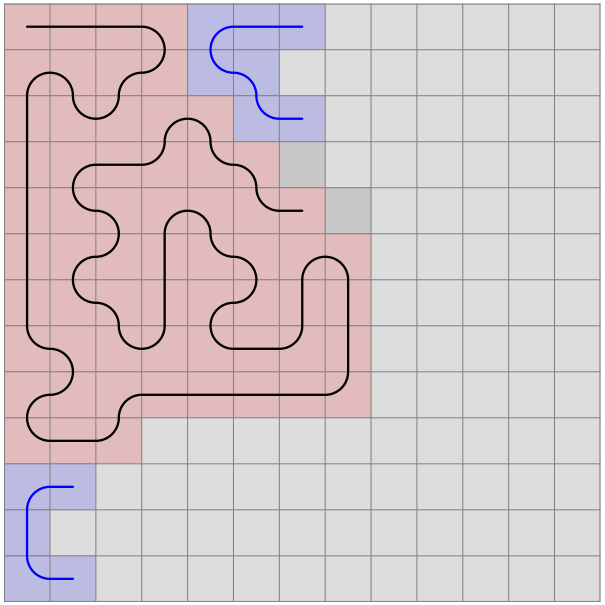




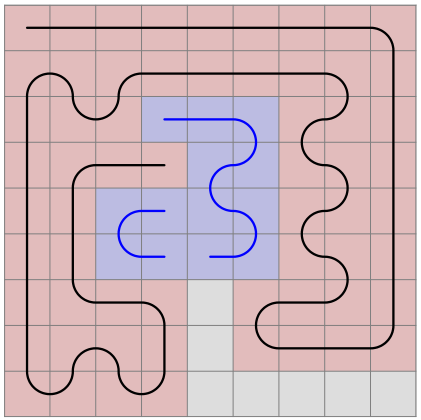
The only difference is, that the obstacle is 3-distance away. With the area being impair, if we enter at A, we must exit at C.
What if we omit D from the area? Then the area will be pair, so we must exit at B, and the only way to get there is from C. And if D is included, we can only step to C from there. Either way, we step away from the area beyond D, so the rule will be:

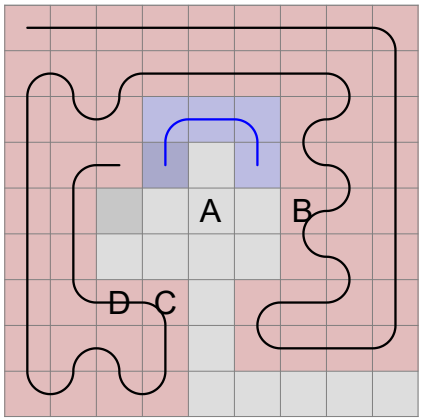


And now the walkthrough is possible.



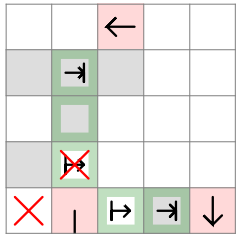
Continuing the 9 x 9 program, we get this at 641 019:



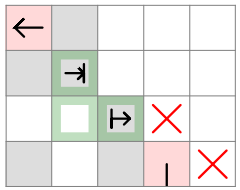


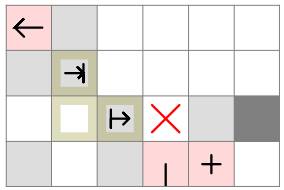
If you enter the pair area straight ahead, you will exit at A, and you need to turn towards B because of the C-shape. Now you cannot go in and out of the area enclosed by C, and the situation would be the same if that obstacle was in D.

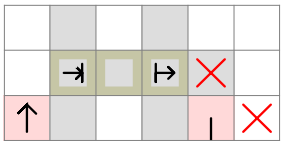
To mark the two areas, each one has to be given a directional obstacle next to the count area end field. In this case, it represents a taken field, but we don't go wrong if we include the border as well.

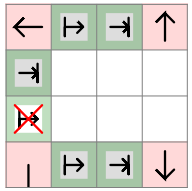


And with this marking system, we can correct the rules prevously made. All rules featuring future line start and end fields have to be rewritten to start with.
So we get the 2-distance across rules, the straight 3-distance rule to prevent a double C-shape, and the square constellation with 3 areas. All of them are rotated clockwise or counter-clockwise.

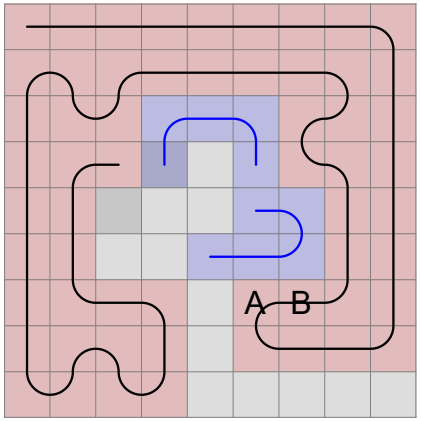




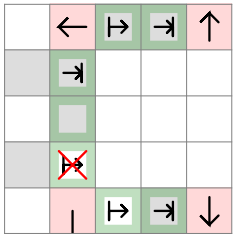




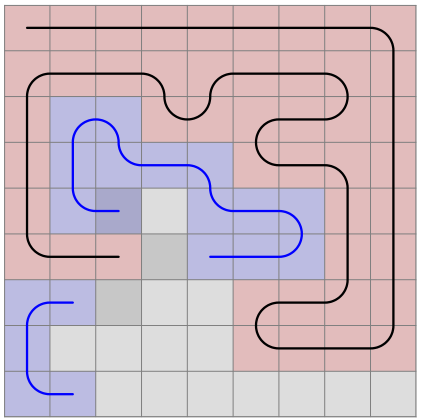
Let's return to the last example and make a modification:



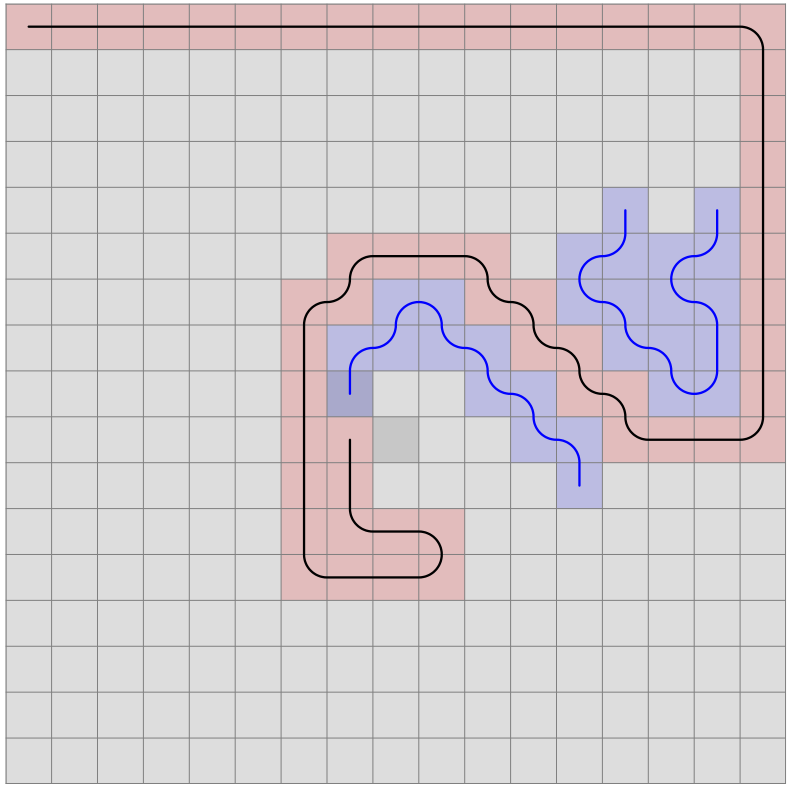
The field previously marked with B is now empty. But we still need to step in that direction, due to the area enclosed by A, which obstacle could as well be in B.
The rule will be now symmetrical. It is similar to the square obstacle pattern.



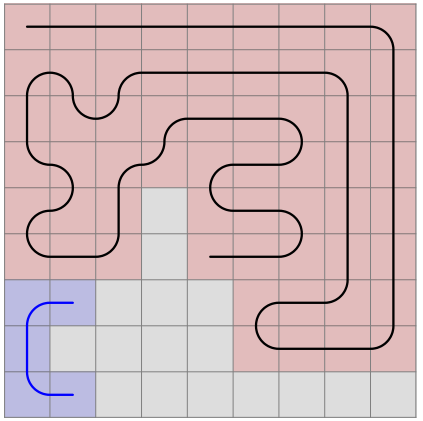
The same concept we encounter at 725 325. We have seen this prevously, just with C-shape, not an area.

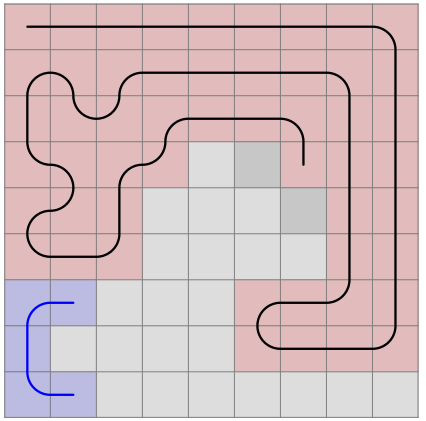


On 17 x 17, we can construct a situation where the obstacle across the stair is 2 behind and 2 to right. As the table size increases, the stair-obstacle narrowing can move infinite distance away from the live end. That's why it is important to group these rules as one.

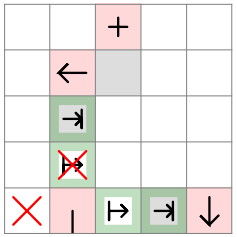


811 808:

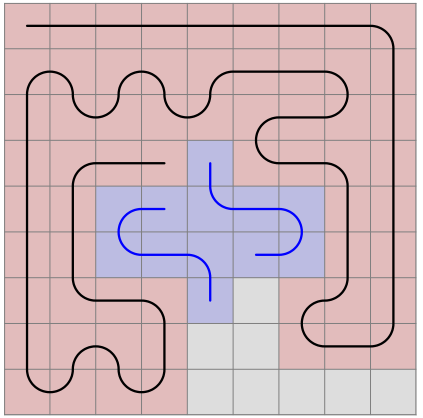


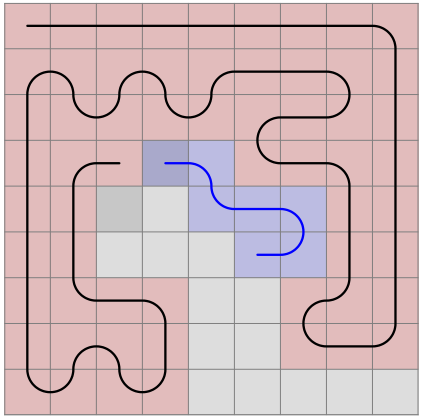


Recognize it is a variation of the square obstacle pattern where instead of an area, there is a C-shape at the rule's upper edge.

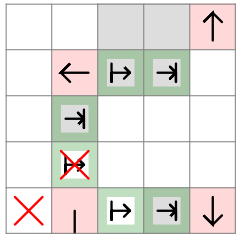


1 261 580:

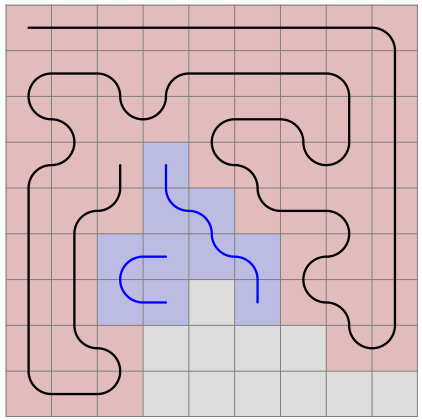


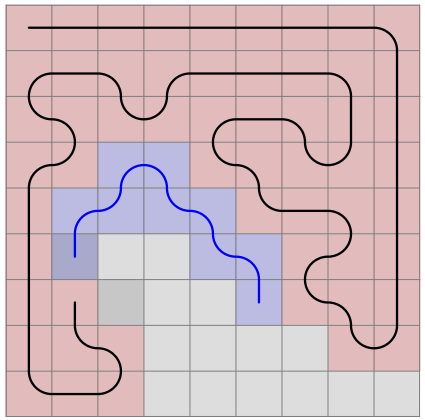


Again, same pattern with area. The upper obstacle is now moved, but it will satisfy the previous examples too. The rule replaces the old one.

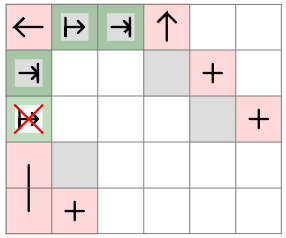


2 022 337 is getting stuck because of the stair-shaped walls that force the future line along them. Therefore, an area is created with only one field to go in and out of it. What is the solution?

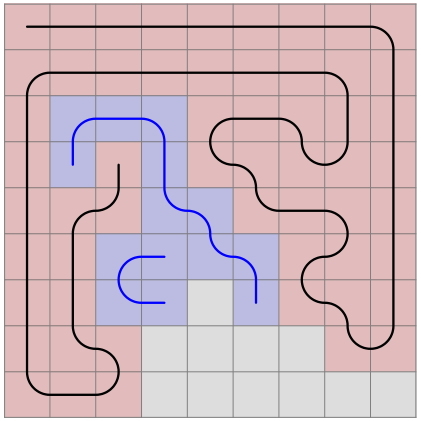


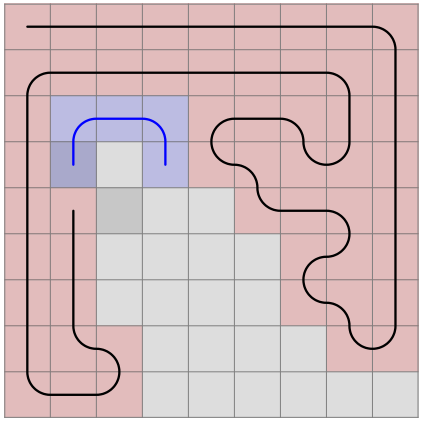


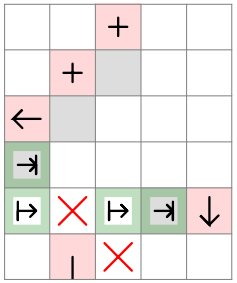
Though not as universal as we want it to be, this will solve this specific situation:



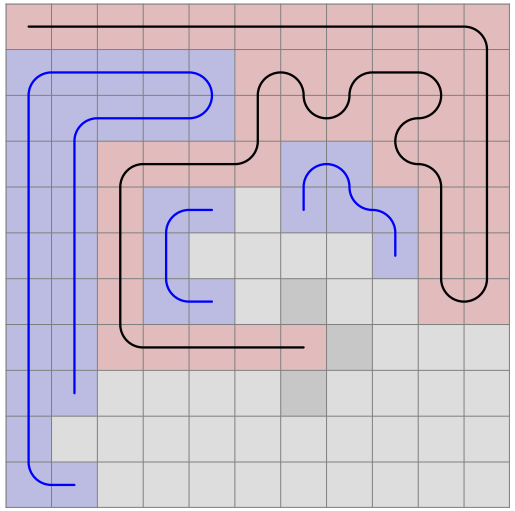
And soon, at 2 022 773 we encounter a similar one:

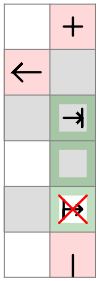




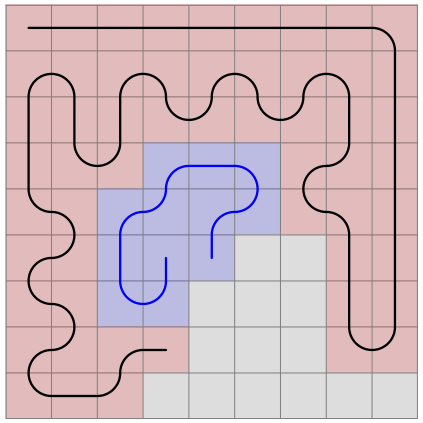


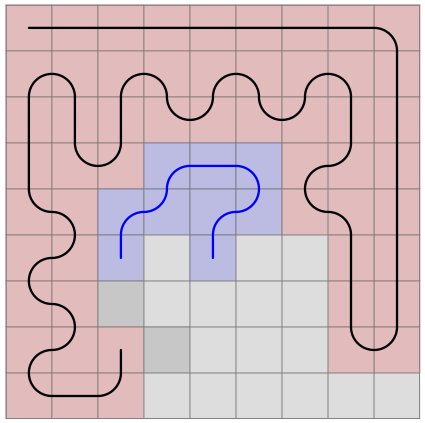
From our experience, the area can be substituted with C-shape.

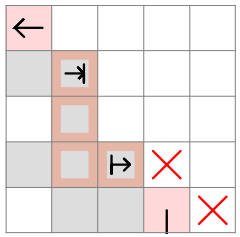




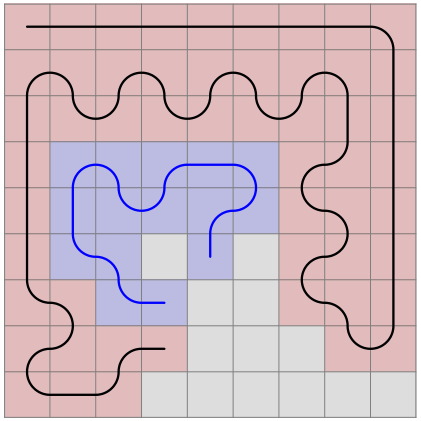
3 225 432 is a variation of the impair area imbalance rules we have seen before.

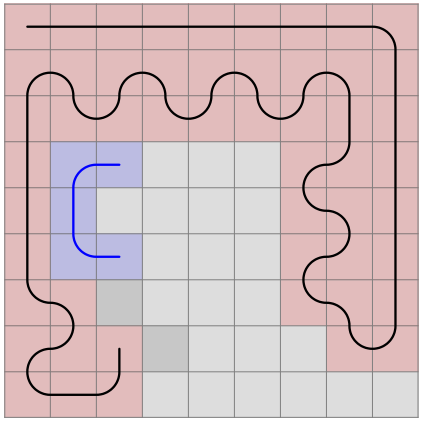




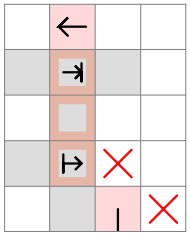


We have all the tools to handle 2 034 575.

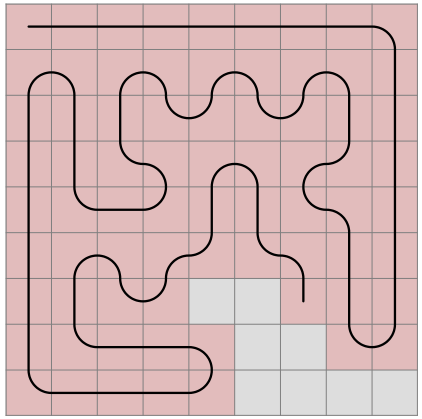


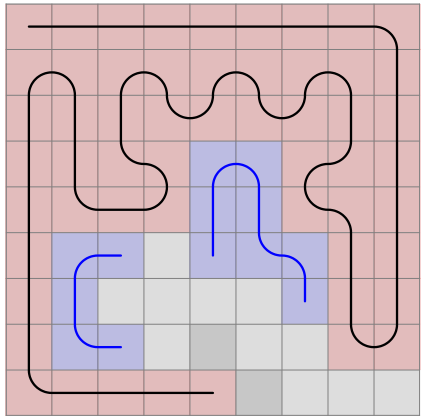


It is an impair area where the number of the starting field's color is less than the other color.

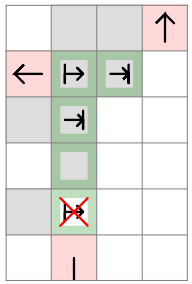


Next stop is at 3 224 847.



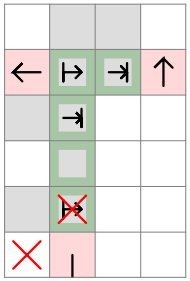


A pair area is created with the obstacle 3 distance away, so if we step into it, we will exit at the middle, but because of an area, we cannot step there.

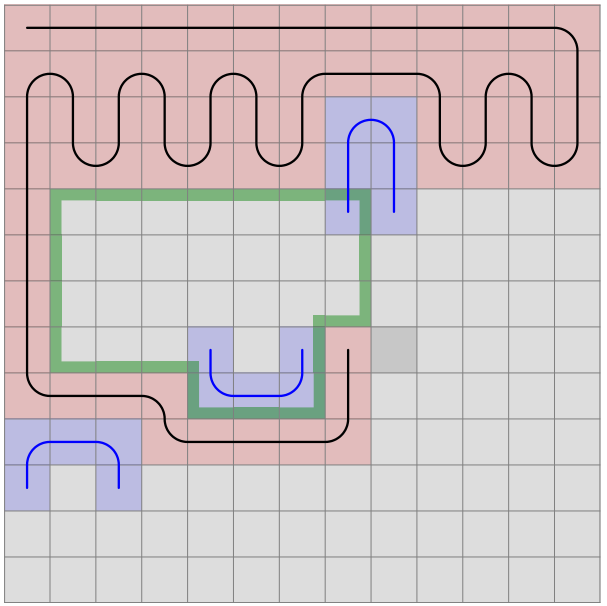


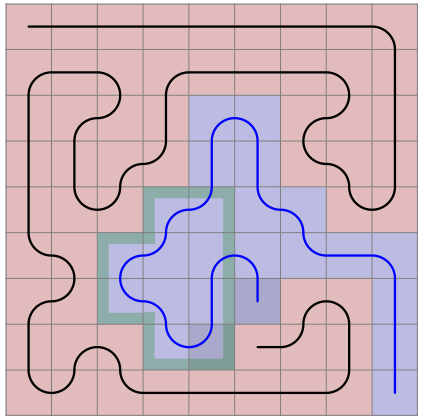
Beware of disabling the left field. If the count area end field is excluded from the area, the area will be impair, thus we will exit at the count area start field, coming from the middle.

But if the obstacle in the upper right corner is moved down, even this will be impossible.



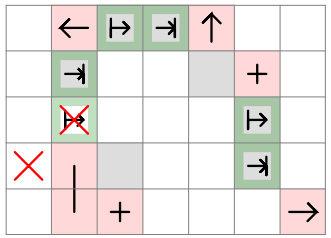
We can recreate this example on 13 x 13.



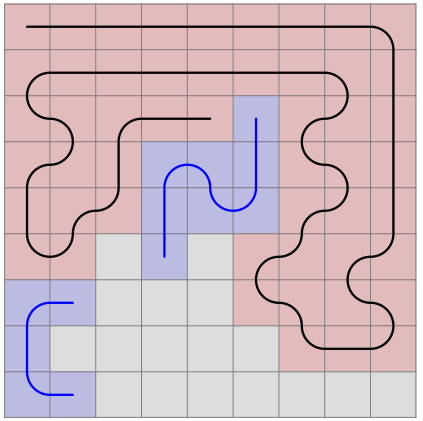


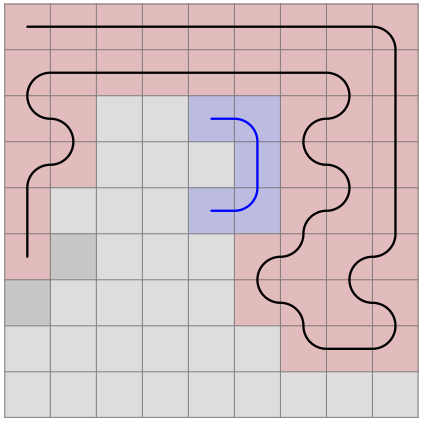
It is not the only thing. So far, when we entered a future line, the program just followed it without checking the possibilities for the next step.
This behaviour needs to be changed too. Future lines are no longer needed, and we should restart the 9 x 9 walkthroughs.

For now, here is the solution to this and the next cases:

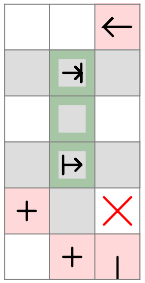


8 076 012 builds upon the existing rule where C-shapes are created on both sides if we enter an impair area.

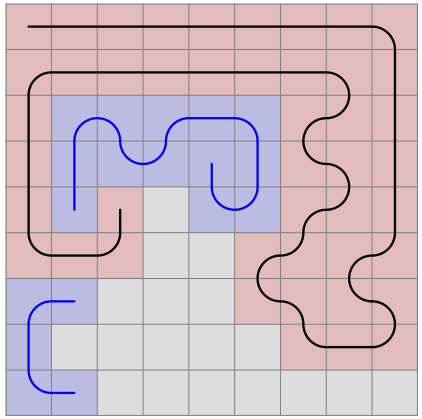


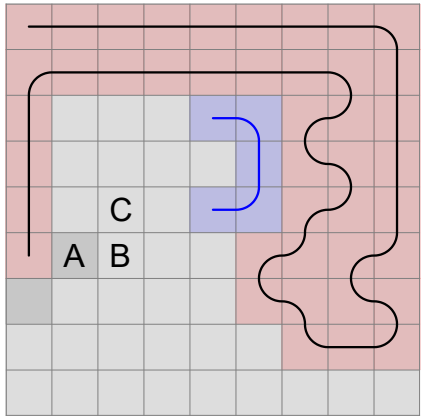


Here, a C-shape at the start would force the line to enter the area.

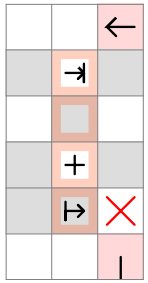


Soon we get a similar situation, only here it is the imbalance of pair and impair fields that is to blame.

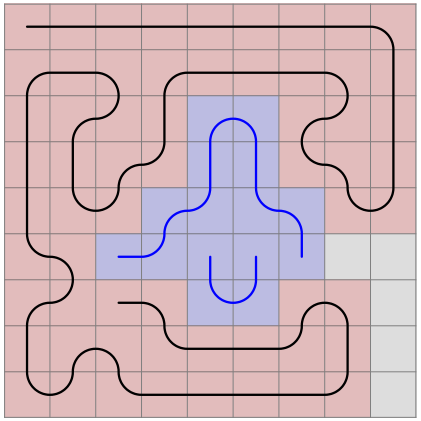


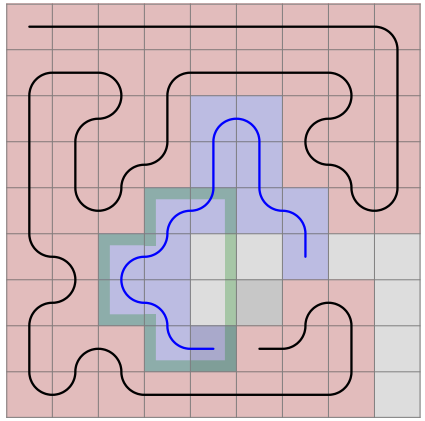


If we step to A, we cannot step left and therefure must continue to B (or right). From B, the only possibility is C, but the 5 x 3 area is not just impair, there is less of the C-parity field than the other.
In the rule, I introduced a new field that indicates the entry point; this has always been the start field until now.

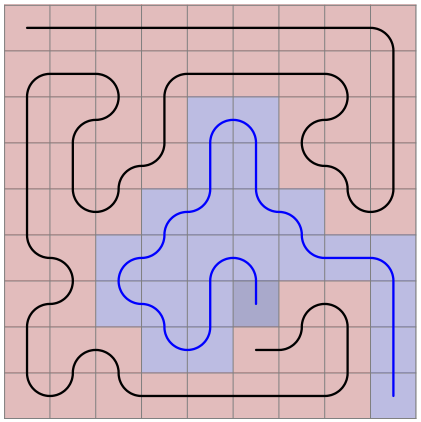


At 19 717 655 the program stops.



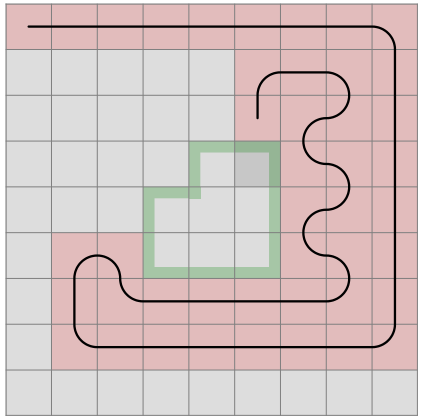


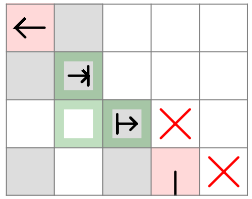
Obvously, we cannot step straight, but had we extended the future line until the end corner, the situation would not have occurred and we would have just got this:



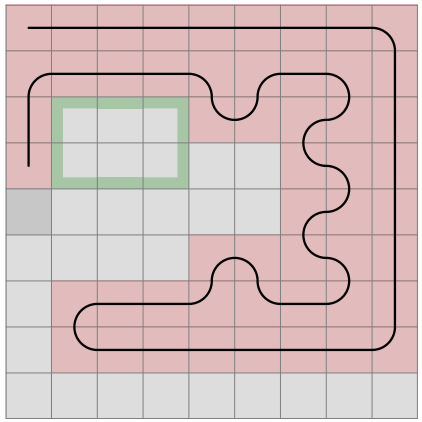
Though the algorithm including the reliance on the future lines is just as solvable, we miss patterns and therefore narrow the spectrum of the discoverable rules. We would eventually discover the patterns as we increase the table, but why not gain the most out of the 9 x 9 study? From now on, future lines are treated as a visible aid, but they do not play a role in deciding which fields are available for the next move. When a possible field is within the body of a future line, the program should stop.

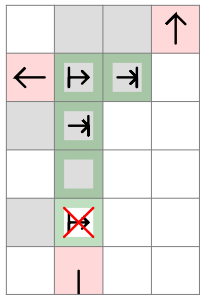
9 121, Count Area 2 Across



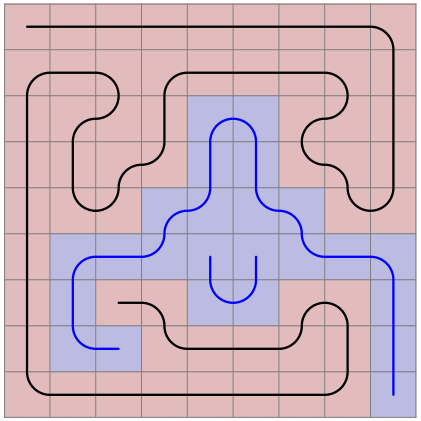


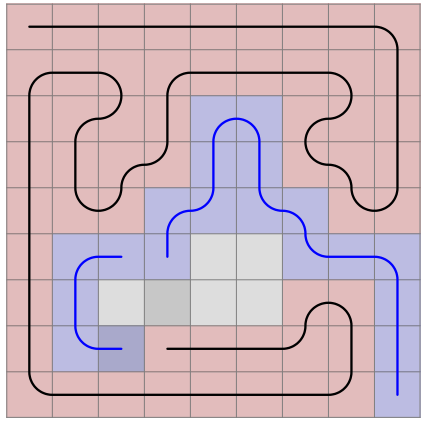
22 328, Straight Mid Across 3 End Area

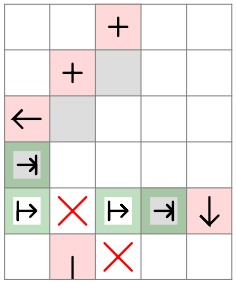


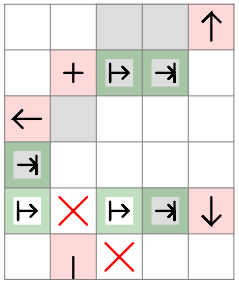


19 718 148 is a slight modification of 2 022 773 where there is an area instead of a C-shape straight ahead.

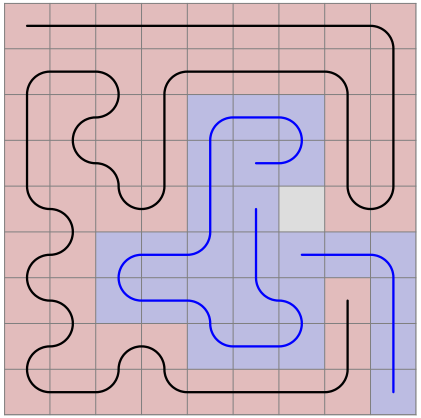


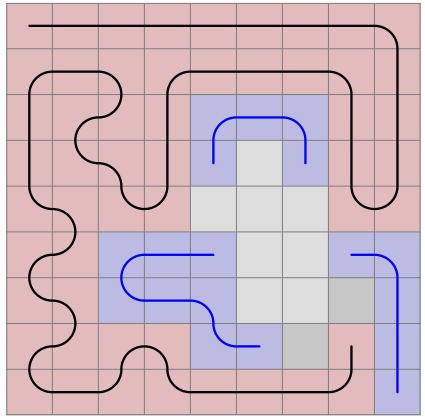




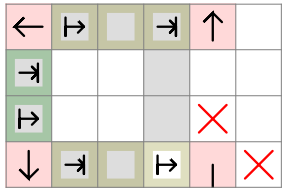


We encounter a new constellation of 3 areas in 23 310 321 where the exit is next to the live end.



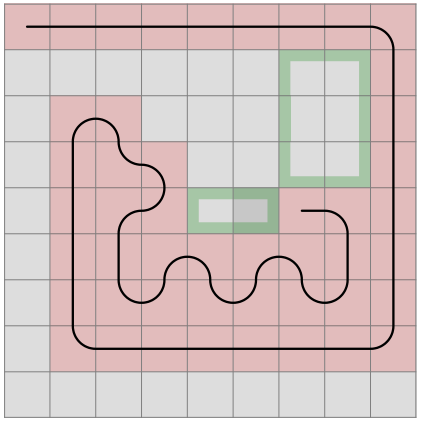


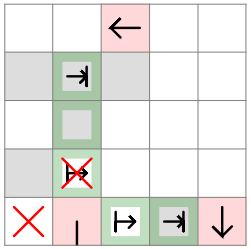
The two empty fields in the middle will not be filled if the don't enter the impair area now.



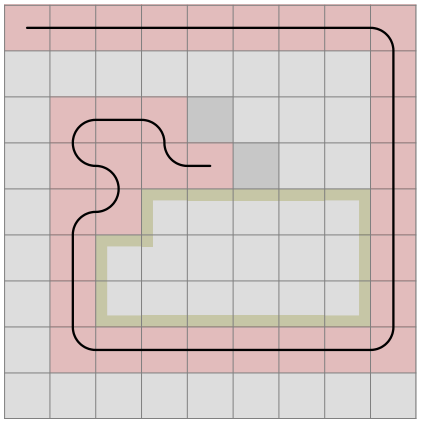
When restarting the run with the new approach, we will find that some walkthroughs were missed previously. The 10 000 th path is slightly younger than before. It is not because the future lines had been drawn incorrecty, but because the 7 x 7 rules that I used in the beginning were not precise for this size.
In the following section I list the 9 x 9 rules in chronological order. The patterns are not introduced when they are first recognized, but when they are first needed, meaning that they disable fields that the other rules don't. And the disabled fields have to be empty.
Still, the number of completed walkthroughs before the appereance of the rule may not be the same as the number of those before getting stuck in the lack of that rule. If the rule disables a field right to a possible field, the left branch would run through first.

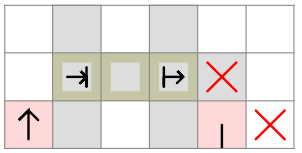
462, Double Area C-Shape



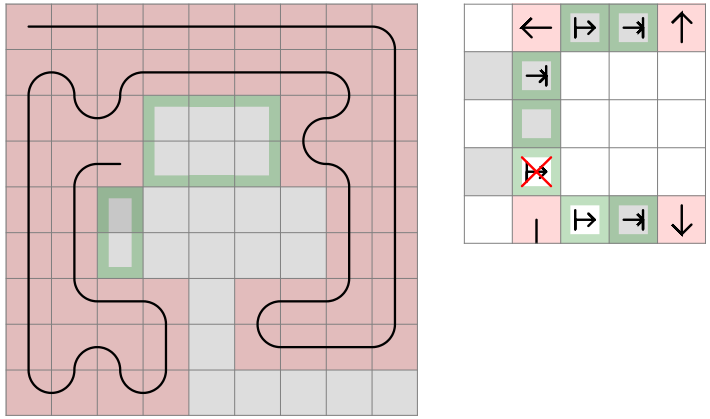


1 861, Double C-Shape

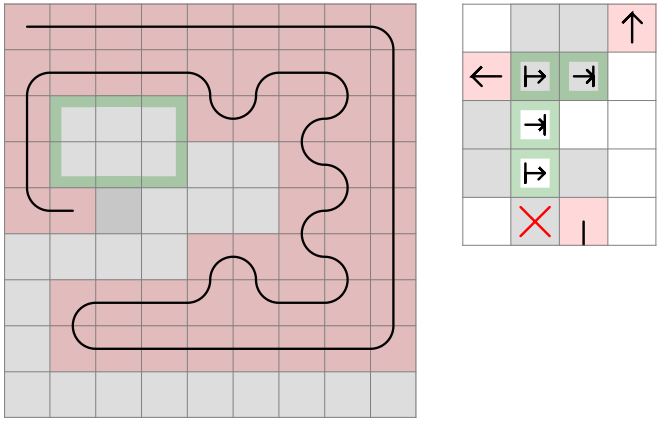




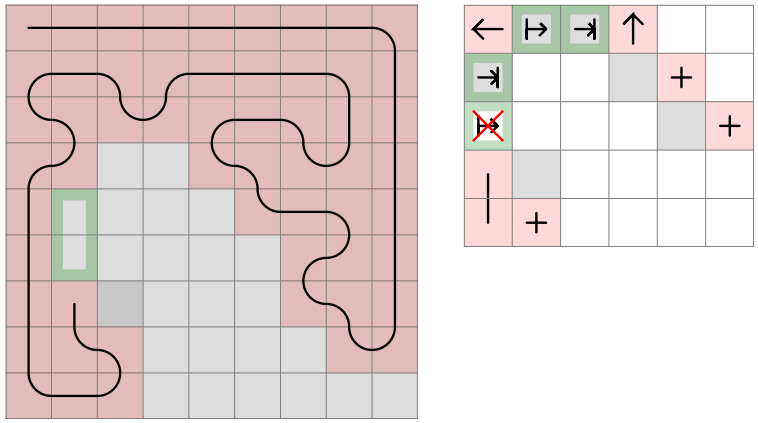
740 363, Triple Area



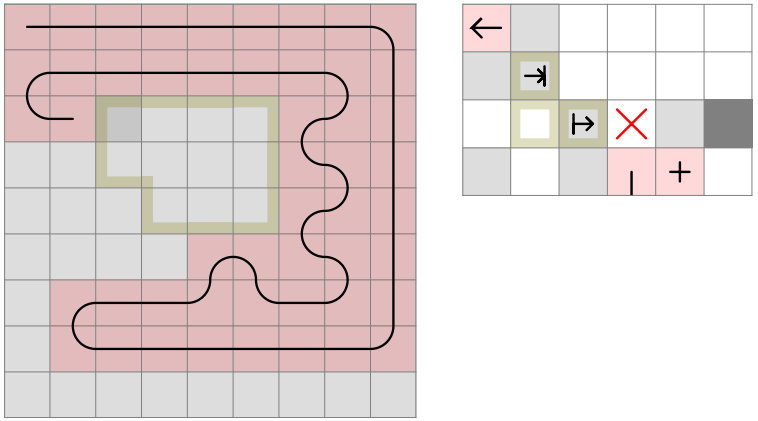
22 328, Straight Across End Area



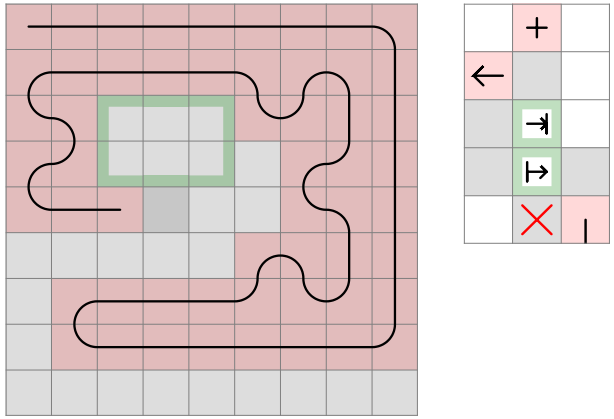
2 022 763, Double Area Stair



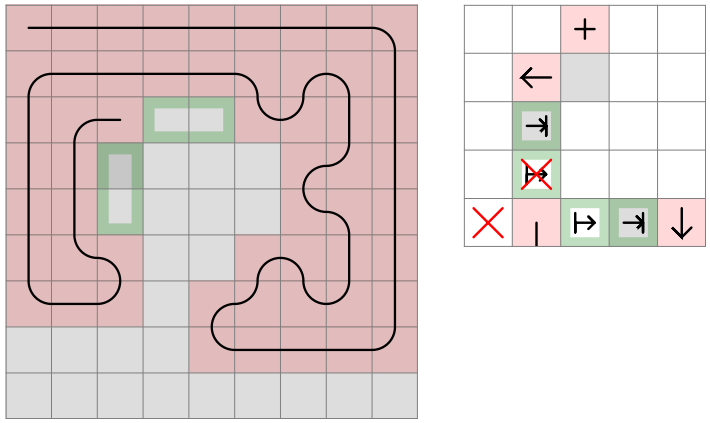
22 362, Count Area 2 Across C



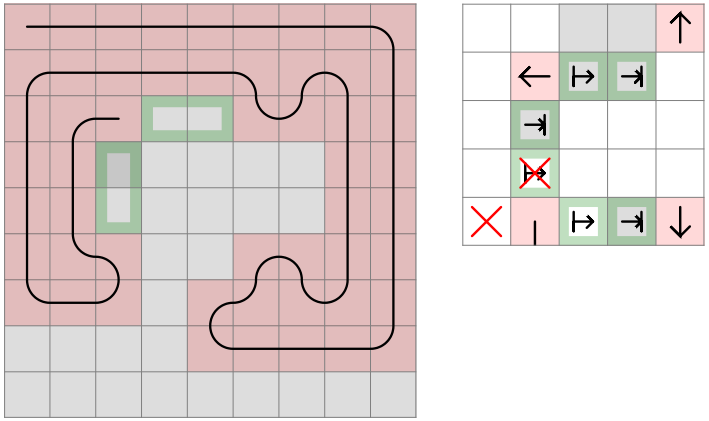
25 153, Straight Across End C



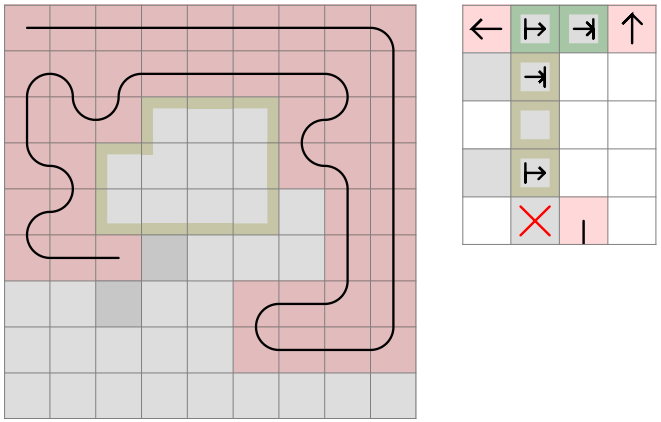
227 130, Square 4 x 2 C-Shape



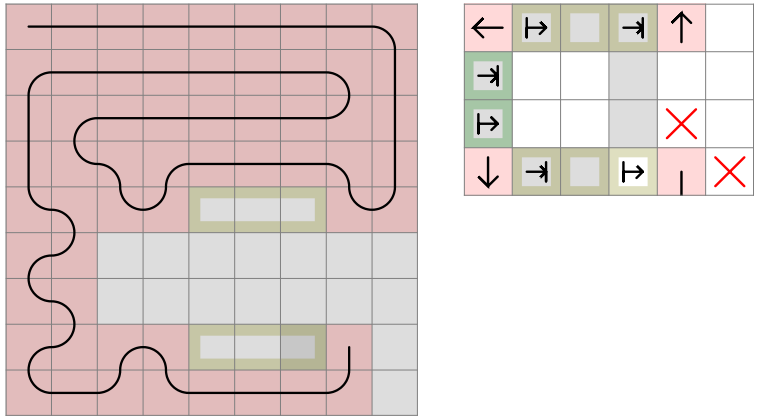
231 960, Square 4 x 2 Area



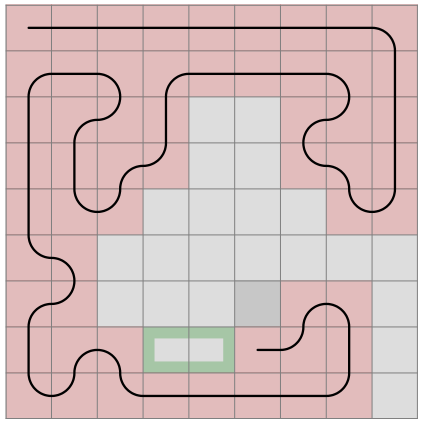
740 129, Straight Across 3 End Area

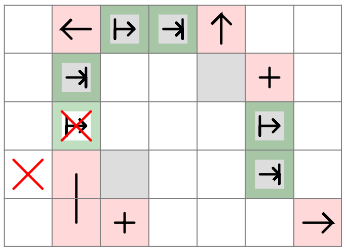


18 665 383, Triple Area Exit Down

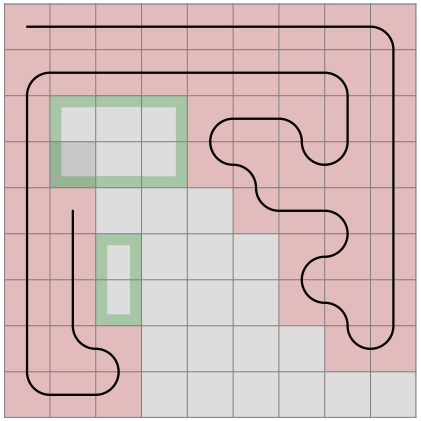


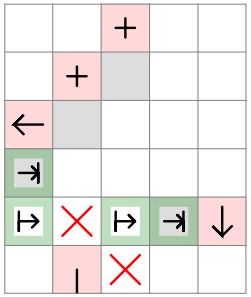
19 720 122, Triple Area Stair



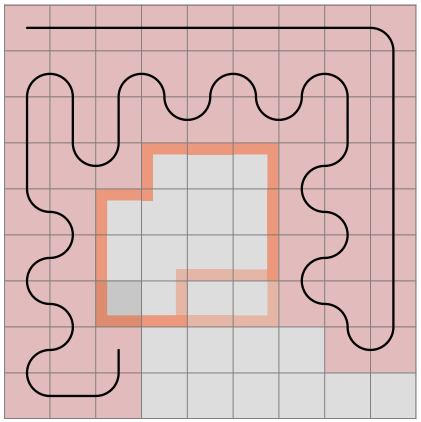


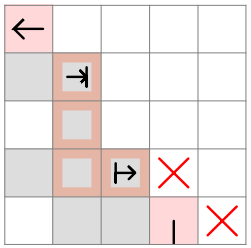
2 023 198, Double Area Stair 2



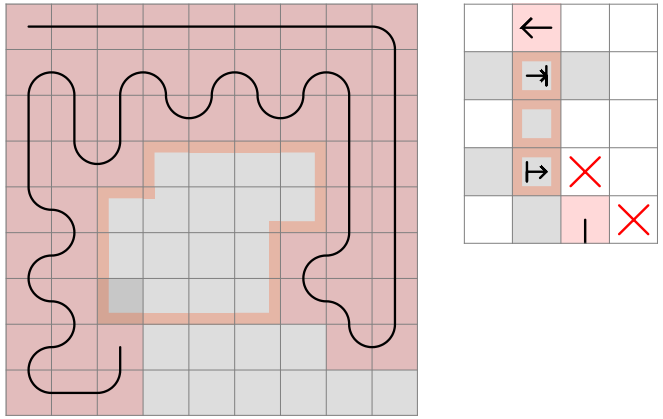


2 034 435, Mid Mid Across 3 Determined (and Mid Across 3 Impair Determined)

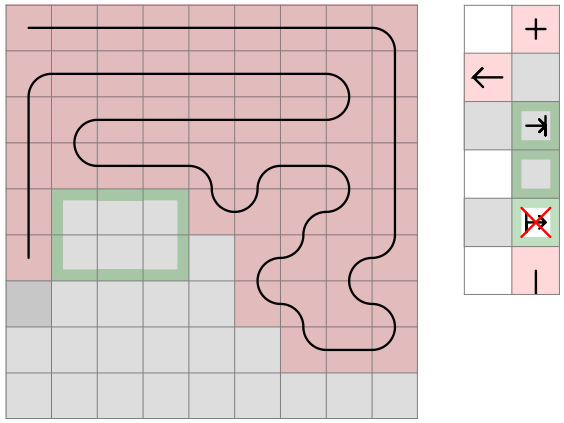




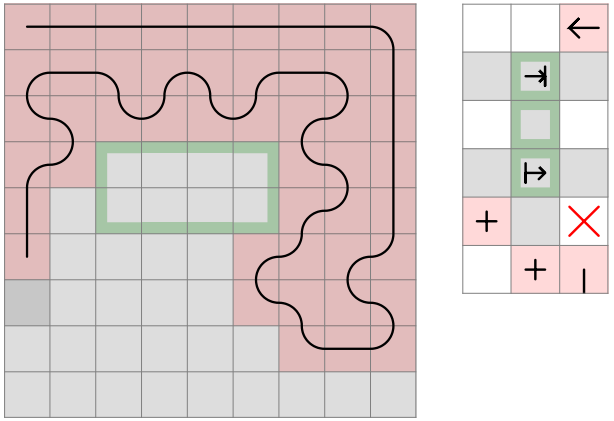
2 059 934, Mid Across 3 Impair Determined



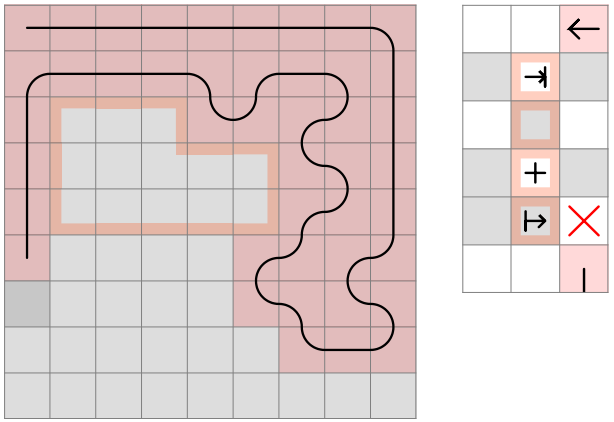
8 076 202, Straight Mid Across 3 End C



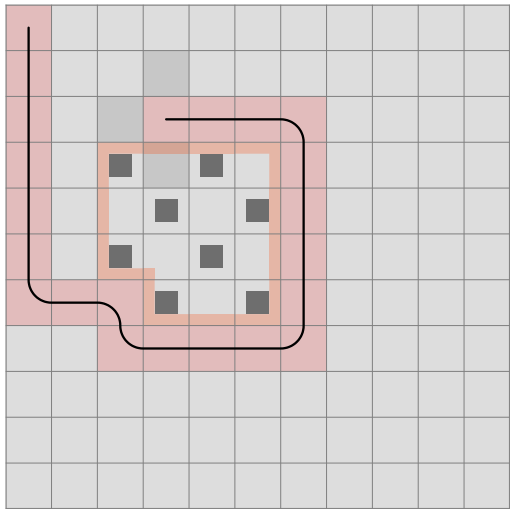
8 076 706, Double C-Shape Start C



8 076 762, Double C-Shape Determined

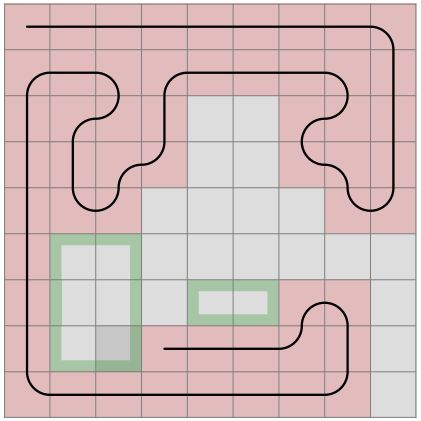


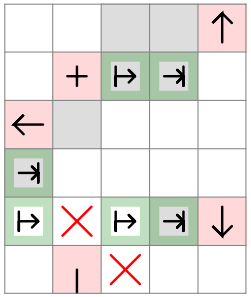
Here, the area is still pair, but there are 12 black fields and 10 white.
To fill it, two lines of an impair length would be needed, each starting and finishing on a black field. Now, it is possible to enter and exit the black field in the upper left corner of the area, but we cannot do it with the field 2 below. And there are no more black fields on the boundary of the area. Filling it is therefore impossible.



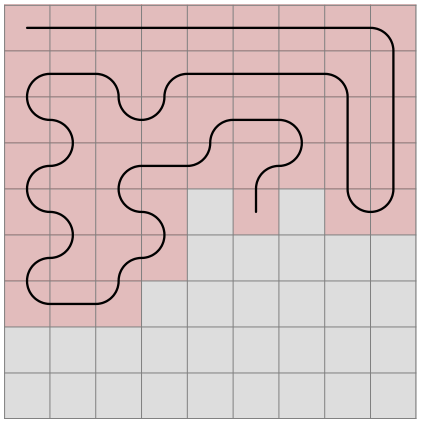
When there are one more black field than white (8 and 7 in this case), the area is impair. If we enter now, we can exit at the count area end (to make a line of pair length), and the corner black can be filled later. We can also enter at the corner black later to exit at the couunt area end.

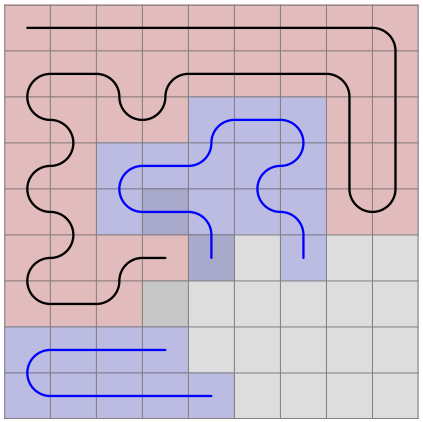
19 720 614, Double Area Stair Area

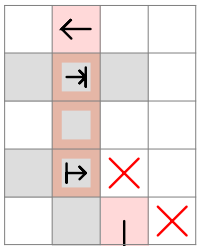


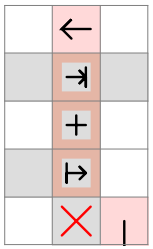


23 350 320 is new, but it shows similarity to the Mid Across 3 Impair Determined rule. As the double C-shape reveals, it is about pair/impair field imbalance.



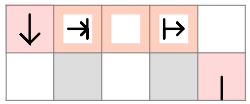


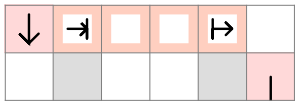




If we step left in the impair area, we can only exit at the count area middle field, but there is one less field of that type than the other.
And no fields can be omitted from the area for entry and exit later.
When the count area start field + middle field is omitted (subtracting a pair amount of cells from the area), the possible exit is the count area end field, which has a different parity than the field to the left.
When the count area middle + end field is omitted, the possible exit is the count area start field, which has again different parity.

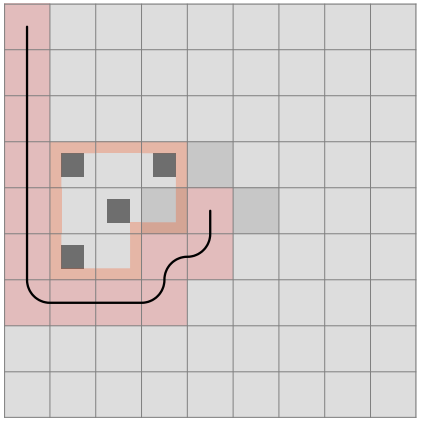
By now, we are able to group some rules and even solve the original 21 x 21 example. Previously, we have covered all of the cases where an obstacle is 2 distance away from the live end. Let's examine distances of 3, 4 and so on in this constellation:



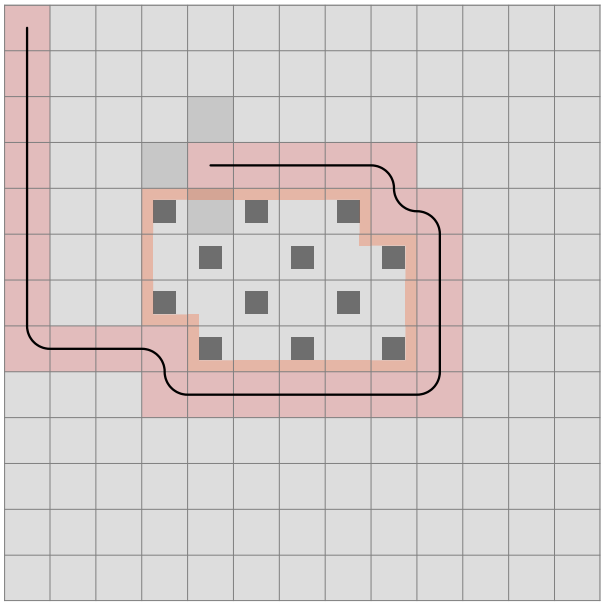


Besides the count area fields being empty, here I also assume that the left field and the field below the count area end is also empty, because the program cannot go through a field twice when creating the arealine. In the future, this might have to be changed and new constellations added.
Knowing the difference between the number of pair and impair fields, we can make a decision in some cases.

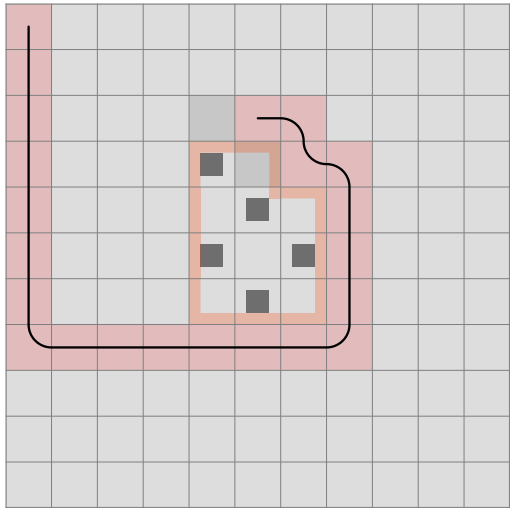
Let's start with this:



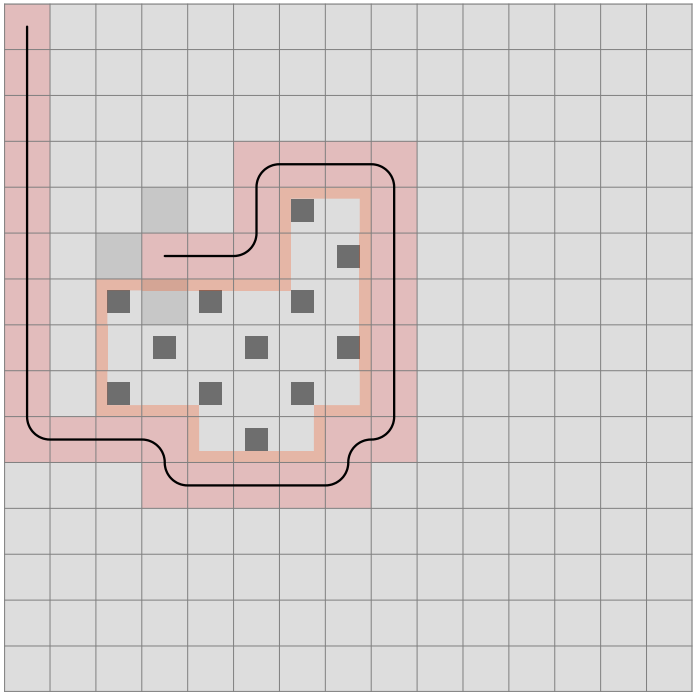
The distance is 3, the area is pair, and the number of pair and impair fields are the same. We can either enter the area now (stepping left) or later (up or right). Either way, we can start and end the area on a different color, so nothing should be disabled.



If black = white - 1:



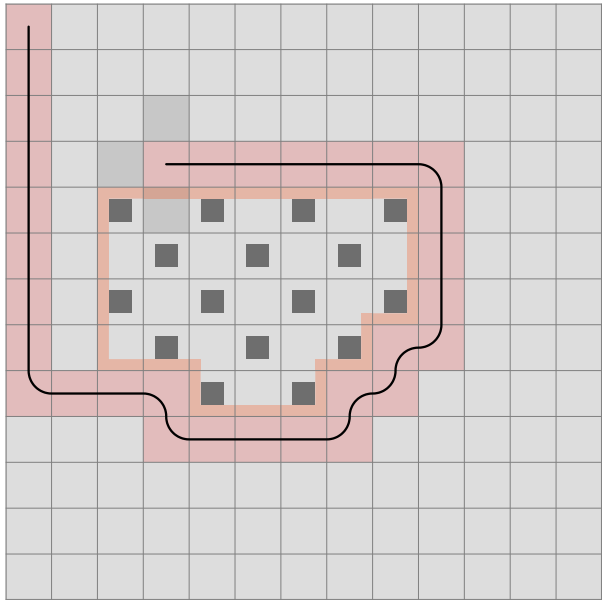
A line that starts and ends on white can be drawn, no matter if we enter now or later. If we enter now, the next field has to be the corner black, and then there will be a line between the two white fields on the boundary.



Here, there are one more white field than black. To fill the area, 1 line of impair length would be needed that starts and ends on white, but if the enter the area now, the corner black could not have been filled. Now there are 2 less black fields available for the rest of the area, but making two lines starting and ending on white is impossible, there are not that many white fields on the boundary.
Needless to say that we cannot enter later either, there is only one white field on the boundary.

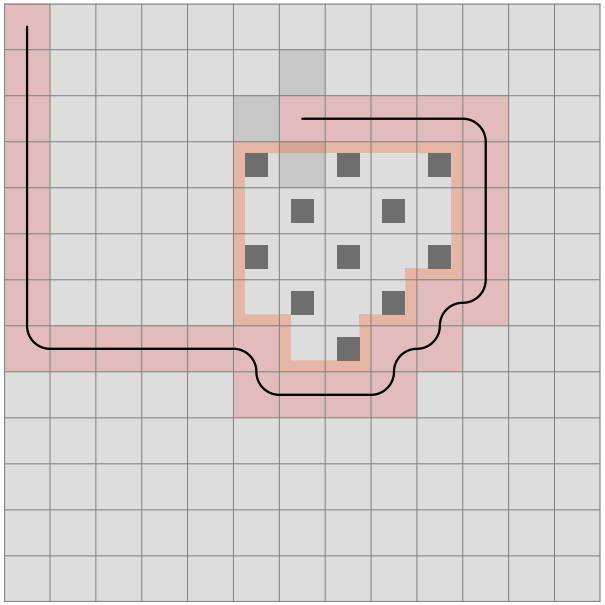
If there are even fewer black fields relative to the white, the situation is the same.

At 4 distance (and in any case), we still don't have a problem filling an area that has equal number of black and white fields.
When the number of black fields are two more than the whites (16 and 14):



Two black to black lines would be needed to fill the area. Apart from the black in the corner, there is only one black field on the boundary.

If black = white + 1:



If we enter now by stepping left, there will have to be one more line even if we exit on black. That line has to go from black to black, so it can only be the corner field. Have we exited at the other black field (the third on the boundary), either the second or the fourth could not have been filled.
This direction therefore has to be disabled.
We can enter later without problems to start at the corner field, then the second, go inside the area and end at the third.

2) Pair distance, impair black and white: 6, 10 etc.

If we enter now and exit at black, (B-1) / 2 black to black lines can be drawn.

If we exit at white, (W+1) / 2 white to white lines can be drawn. (We have to move to the corner black and exit at the first white during the first line)

If we enter later, (B+1) / 2 and (W-1) / 2 black and white lines are possible, respectively.

The rule is double:
- If the number of black fields is greater or equal than the number of whites plus (B+1) / 2, we cannot enter now.
- If the number of white fields is greater or equal than the number of blacks plus (W+1) / 2, we have to enter now.

3) Impair distance, impair black and pair white: 5, 9 etc.

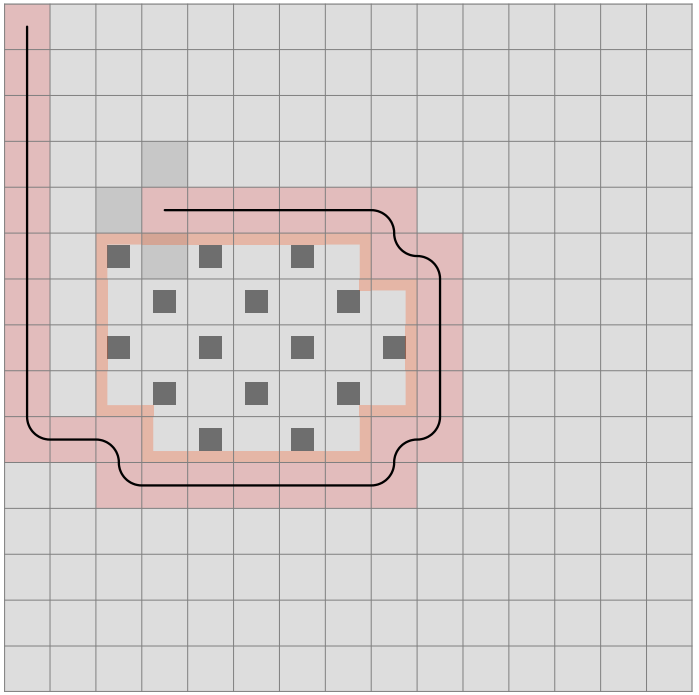
If we enter now and exit at black, (B-1) / 2 black lines are possible.

If we exit at white, W/2 white lines are possible.

If we enter later, the number of black lines can be up to (B+1)/2, the whites W/2

Single rule: When the difference is at least (B+1)/2, we cannot enter now.

If black = white - 2:



Two white to white lines would be needed, but there are only 3 white fields on the boundary, and none is the corner.

Without finding concrete examples, at 5 distance I only draw the boundary and go through the different possibiities.

<u>black = white + 2</u>

If we enter now and finish at black, only two black fields remain. Drawing two black to black lines is not possible, so this direction has to be disabled.
We can enter later, go through the corner and draw another black to black line.

black = white + 3

There is not enough black fields for three black to black lines.

black = white + 1

We can enter now, finish on a black field and draw another black to black line. Or enter later.

black = white - 1

A white to white line is possible in either case.

black = white - 2

We cannot enter now, because even if we end on a white field, only one white remains, and that is not the corner. And cannot enter later either.

The same procedure applies at 6 distance.

black = white + 2

The number of black fields is the same as previously, so entering now is not possible.

black = white + 3 is impossible.

black = white + 1 and black = white - 1 is possible.

black = white - 2

Entering now is possible if we step upwards and exit at the neighbouring white field. Two white fields remains.

But we cannot enter later and make two white to white lines using three white fields.

From now on, we can distinguish between four cases:

1) Pair distance, pair black and white (indicated by B and W): 4, 8, 12 etc. distance

If we enter now and exit on black, B-1 black field remains on the border. B-1 is impair, and the corner alone can make a black to black line. Hence (B-2) / 2 + 1 = B / 2 black to black line would be possible, but...
- if the first line finishes at the corner black, the opportunity for it to make a single line is lost, and the remaining B-1 black fields cannot make B / 2 black lines.
- if it finishes on any other black field, on one side there will be a section that has white fields on both ends (below the greenish fields were taken by the first line)

A black to black line cannot have three white fields on the border (unless more black fields were used up).

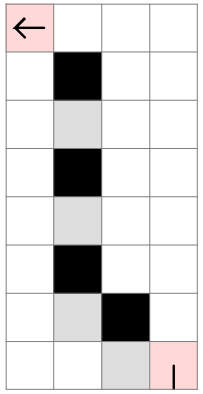
So there can be at most B / 2 - 1 black to black lines.

If we enter now and exit on white, W-1 white fields remain, so 1 + (W-2) / 2 = W / 2 white to white lines is possible.

If we enter later, the number of possible black to black lines is at most B / 2, and the white ones W / 2.

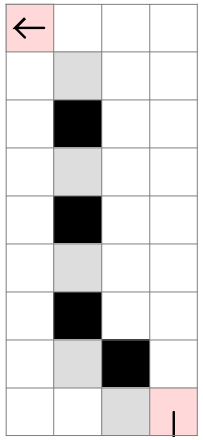
Our rule will look like this: If the number of black fields in the area is greater or equal than the number of white fields plus B / 2, we cannot enter now.

6 distance:



Now: 1W -> 2B
Later: 1W -> 3B
3B: cannot enter now

7 distance:



Now: 2W -> 2B
Later: 2W -> 2B
No rule.

4) Impair distance, pair black and impair white: 3, 7 etc.

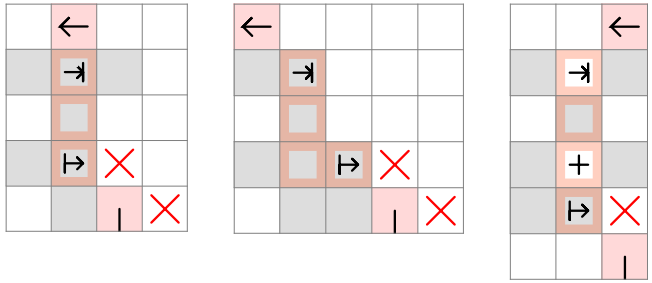
If we enter now and exit at black, B / 2 black lines are possible.

If we exit at white: Similarly to the first of the four cases, a black to black edge will remain on one side. Drawing (W-1) / 2 more white lines is either not possible, or if we do so, the corner black may make up a black to black line, decreasing the difference.

When entering later, B / 2 and (W-1) / 2 are the numbers. Since they match the above, no rule is applied.

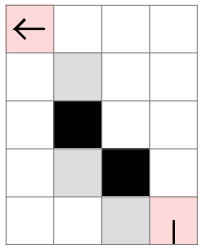
Check the original 21 x 21 example. Two steps back, there will be 9 distance with the wall to the left. The number of black fields on the edge is 5, therefore there cannot be 3 more black fields in the area than white, but counting them, they are 51 and 48.

Does this procedure apply to any of the 9x9 rules? Not exactly, but let's look them through. Here are all of them that deal with black and white field imbalance:



As a reminder, they indicate impair areas, and in the first two case if the count area start field (black) type is not 1 field more in the area, we cannot enter later.
In the third the field marked with + (white) is the type that needs to be 1 more than black, otherwise stepping forward is forbidden.

I will take the second rule under examination as it contains both a horizontal and vertical offset.

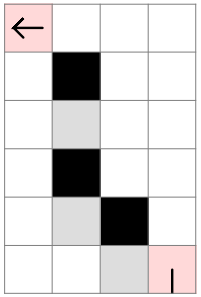


If we enter now:
- We can exit at the end white, so 1 white line is possible.
- We can exit at the black farthest away and then make a line using the black field closest. 1 black line is pssible.
I will mark it like this: 1W -> 1B

If we enter later:
- Having two black fields, 1 black line is possible.
- There is only one white field. It sits in a corner, but the two black fields will give 1 black line. 0 white line is possible.
0W -> 1B

So if there are 1 more white fields in the area than black (1W), we cannot enter later.

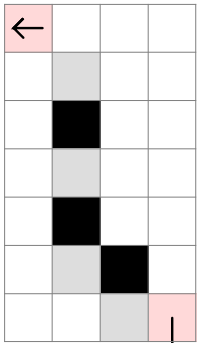
Now let's increase the vertical distance.



If we enter now, 1 white line to 2 black lines are possible. There are 2 black fields on a corner.
1W -> 2B
Later, 0 white line and 2 black lines can be drawn.
0W -> 2B

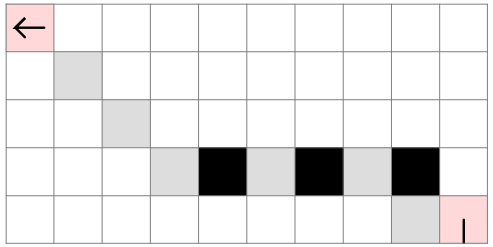
Conclusion: in case of 1W, we cannot enter later.

5 distance:



Now: 2W -> 1B
Later: 1W -> 2B
2W: cannot enter later
2B: cannot enter now

But we are not finished, we still need to examine the distance of 8.



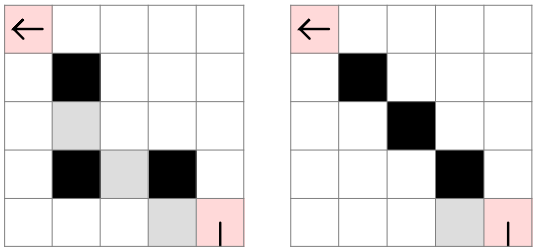
Now: 4W -> 1B
After entry, we need to move up to fill the corner black and exit at the first white field.
At 4 distance, only 2W was possible, but now we can exit at the first white and fill the area when entering at the second and exiting at the third.
Later: 3W -> 1B

After this practice, let's calculate how many white and black lines we can draw when we have an obstacle x and y distance away.

In case of 7, there are 4 fields added to the 3-distance example. We would expect that in case of 2W, we cannot enter later, but now, 2W is possible even when entering later, because one line will be the corner white, and the other can go between the other two white fields, taking up all the blacks along the way.
From now on, increasing the numbers by 1 for every 4 distance increase will work.

The next thing to do is the horizontal increase.

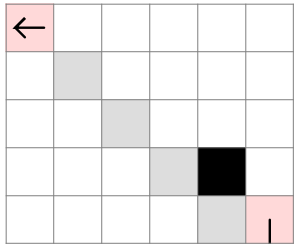
3 distance:



The picture on the left is the representation we have used so far. However, we cannot exit at the black in the middle, and when we exit at one of the whites, the other is only accessible for immediate entry. Therefore, I will add the extra field. If there is a taken field anywhere ahead acting as the obstacle, we can get to it by drawing a straight line and a stair.

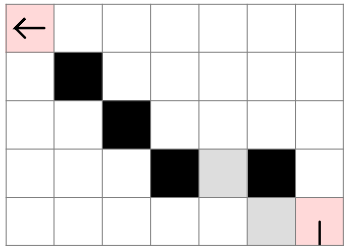
Now: 0W -> 2B
Later: 1B -> 2B
If we used all 3 black corners for a separate line, we would not enter the area.

4 distance:



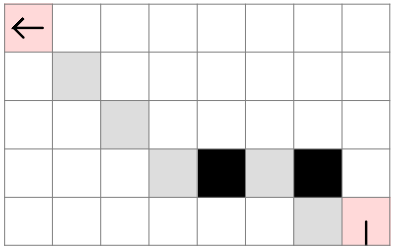
Now: 2W -> 0B
If we exited at the nearest white after entering, we have either not entered the area or not filled the black field.
Later: 2W -> 0B

5 distance:



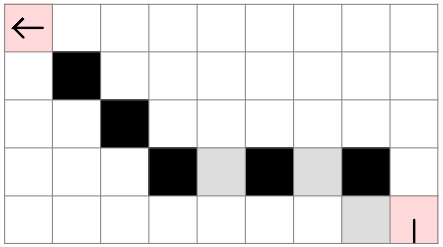
Now: 0W -> 3B
If we reserved the 3 black corners, the only way to end the first line on black is to move downwards after entry.
Later: 0W -> 3B

6 distance:



Now: 3W -> 1B
Similarly, we need to move down after entry in order to finish at the second black field, leaving the first for itself.
Later: 3W -> 1B

7 distance:



Now: 1W -> 3B
Later: 0W -> 4B
There cannot be one white line, because out of the first three black fields only 2 would be filled.

Notice that as we added 4 distance to 3, now both the white and the black end of the ranges have increased by one.