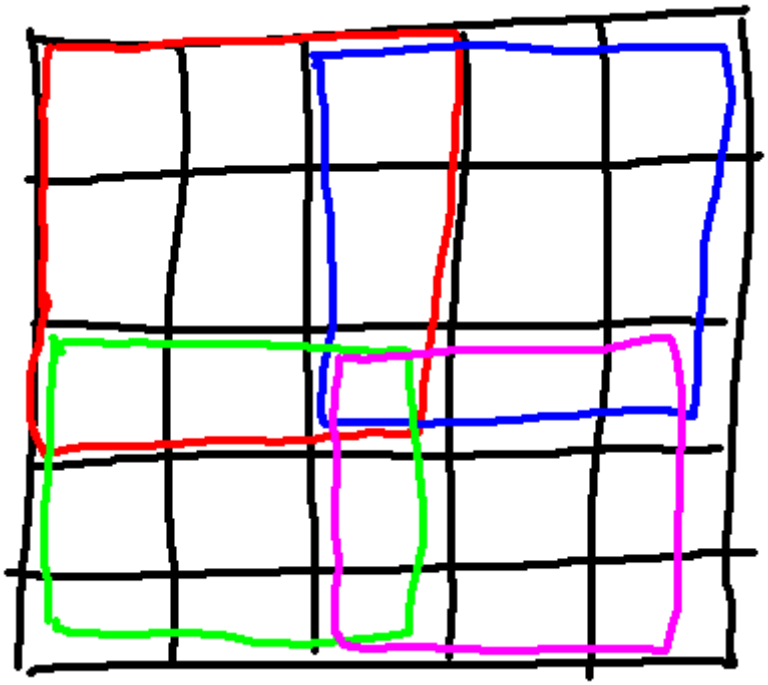
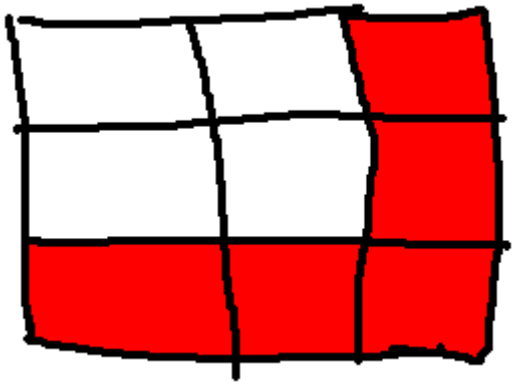
When the grid has size 5×5, we can store the entire grid directly. Let $h = \lfloor \frac{n}{2} \rfloor$.

We store these rows $[0, h), [h, n), [n, n + h), [n + h, 2n), [2n, 2n]$. We can see that $h \leq 10$ when $n \leq 20$, so are able to directly store the entire grid in this fashion.

Now, we want to transfer the 5×5 grid into 3×3 grid.



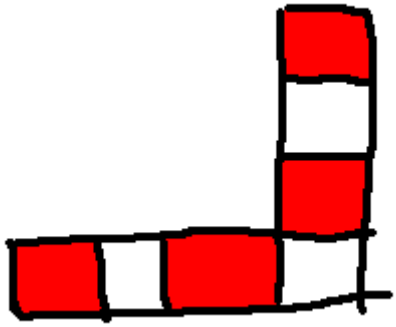
We want to store each $(n + 1) \times (n + 1)$ square in a compact way such that we are able to get all the information we need. Let us focus on the top-left region (the red one).



We only care about the cells in this region and the connectivity between them. There might be islands that are already formed that do no touch the read region. We will count them separately.

Now, when $n = 20$, we need to use 41 bits to store the entire state of the red region. and there can be up to 100 islands in the white region that is not connected to the red region. So we have already used about 50 bits.

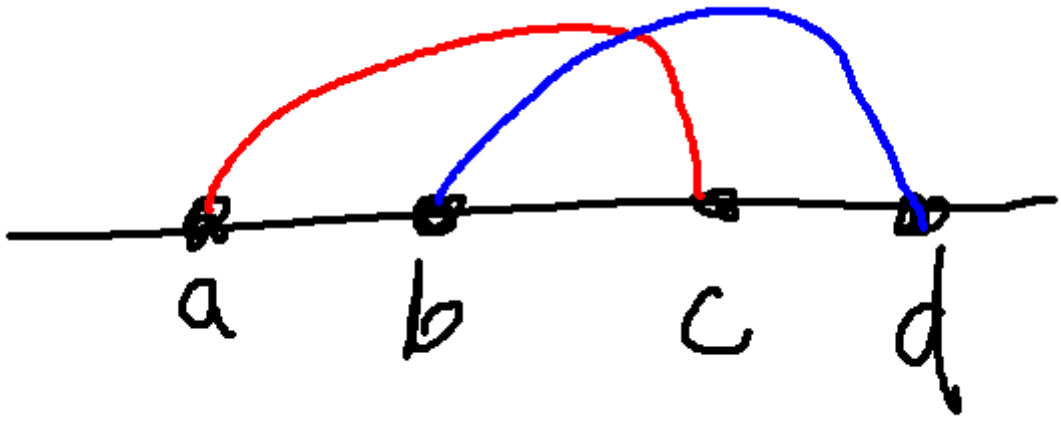
Now, we need to store the connectivity of those lands in the red region using the other bits. We can actually do this in 42 bits only. Firstly, let us focus on the red region only. There are at most 21 connected components. If 2 lands are adjacent in the red region, they belong to the same connected component.



The worst case where the number for connected components for $n = 3$ is shown here.

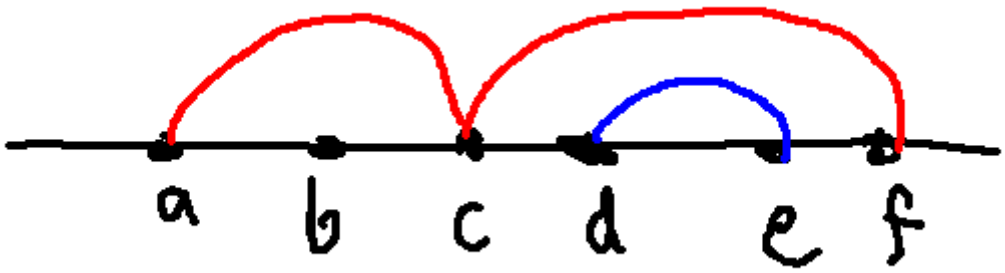
Now, here is the crucial observation. Let us label the connected components in some order of walking along the line.

Suppose $a < b < c < d$. If (a, c) is connected and (b, d) is connected. Then the entire (a, b, c, d) must necessarily be all connected.



There is no way you can not make the red and blue lines intersect.

This means that we can view this as a bracket sequence. That is, if (a, c) are connected and there is no $a < b < c$ such that (a, b) is connected, then we add an edge between a and c. This can be viewed as adding a (to a and) to c.



So if the connectivity looks like this, we will store $(_a)_c(_c(_d)_e)_f$. Notice that for each index, there is at most a single copy of (or). Since we have shown that there are at most 21 connectivity components on the red region, we only have to use 42 additional bits to store the connectivity. Therefore, we can store the important things we need from a corner of the grid in less than 100 bits.