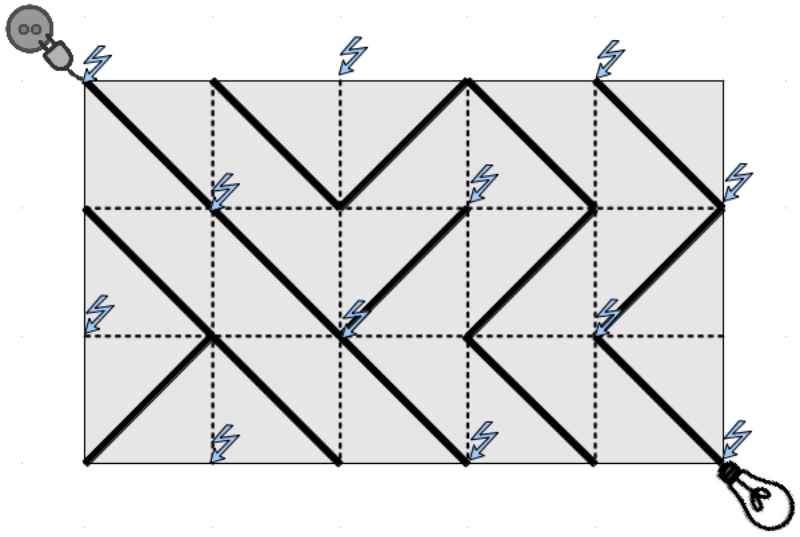
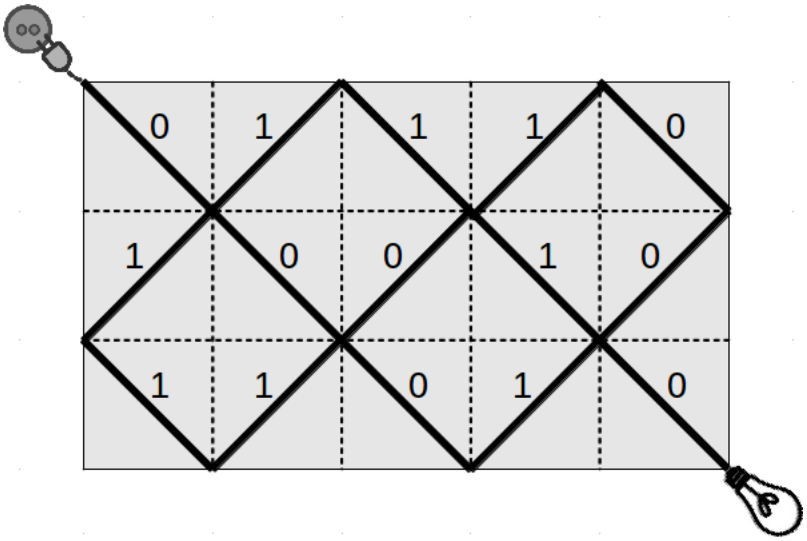
Switch the Lamp On

From the task description we know that the top left vertex of the plate is connected to the power source. The only other vertex in the top left tile that can be connected to the power is the opposite of that one. Observe that all the vertices that are opposite to those that can be connected to the power can also be connected to the power. We can continue in a similar way and identify all the vertices that can be connected to the power. Note that each tile has only two such vertices.



Further in the text we'll only refer to those vertices that can be connected to the power.

Let us construct a weighted graph. Let there be an edge between every pair of vertices belonging to the same tile. The edge has 0 weight if the wire on the tile matches the edge, or 1 otherwise. The weight 1 is chosen to indicate that the tile would have to be turned for the power to flow through its wire.



To find the solution we have to find the shortest path from the power source (vertex s) to the lamp (vertex t) in this graph. We can achieve this in O(NM) time using a double-ended queue. Let $distance_i$ be the currently estimated distance from the power source to the vertex i. We will begin by putting the vertex s to the queue and setting $distance_s$ to 0.

Processing the vertex v from the front of the queue we remove it for the queue and for every node u that has a common edge with v and $distance_v + weigth_{v,u} < distance_u$ or $distance_u$ wasn't set yet, we update $distance_u$ to $distance_v + weigth_{v,u}$. After every such update if $weigth_{v,u} = 0$, we push vertex u to the front of the queue, and if $weigth_{v,u} = 0$ – to the end of the queue.

To speed up the search we don't want to process the same node twice which could happen if we add it to the back of the queue and later to the front. To achieve this we'll put the current $distance_i$ value to the queue together with every vertex we put to it. Here's the pseudocode for finding the shortest path:

```
dist[s] = 0
deque.push_front(s, 0)
while deque.front() != (t, *)
  (v, d) = deque.pop_front()
  if dist[v] == d
    for each v neighbour u
      if dist[u] < dist[v] + weight[v, u] or dist[u] is undefined
        dist[u] = dist[v] + weight[v, u]
        if weight[v, u] = 0
          deque.push_front(u, dist[u])
        else
          deque.push_back(u, dist[u])
```

When N + M is an odd number, the lamp can not be connected to the power.

Task idea by Martynas Budriūnas, spoiler by Martynas Budriūnas and Jūratė Skūpienė.