BlackBox Solver

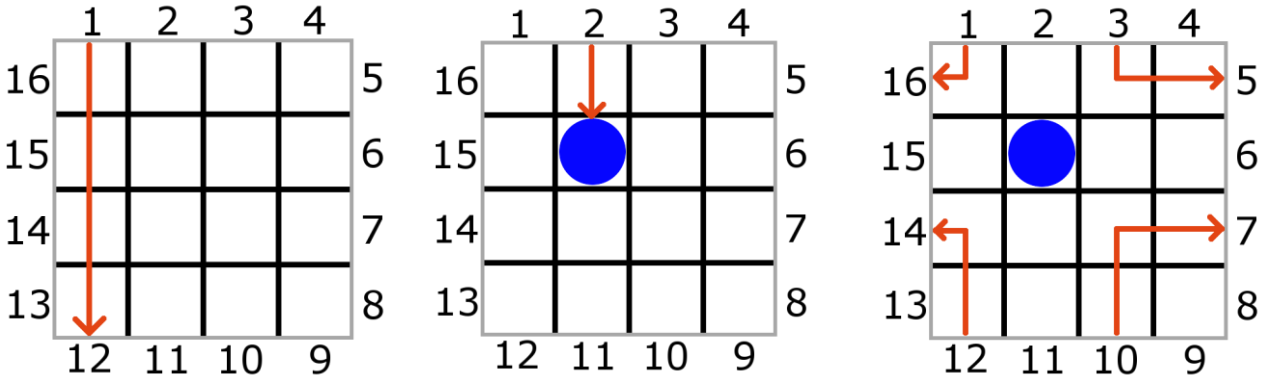
Alexander Janns

7.12.2018

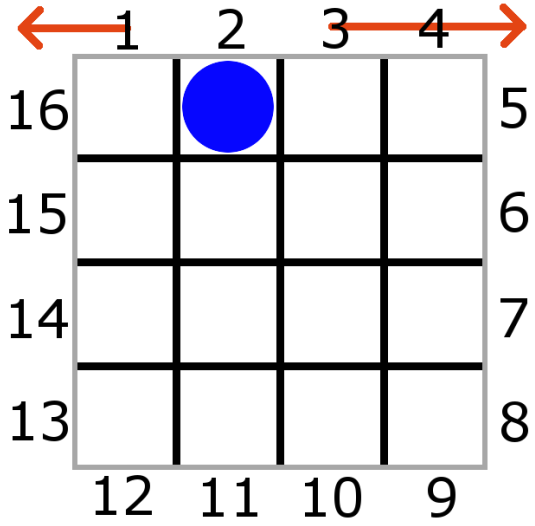
Source:
http://ftp.informatik.rwth-aachen.de/fairspielt/Spiele/blackbox.php

- A game designed by Eric Solomon in the 70‘s, inspired by the CAT scanner
- Objective is to guess the positions of ‘atoms’ based on the feedback of probing ‘rays’
- Possible feedback: Reflection, Hit, Detour, Miss
- Scored on how many rays you needed for your guess (lower is better)
  - Point penalty for wrong guesses

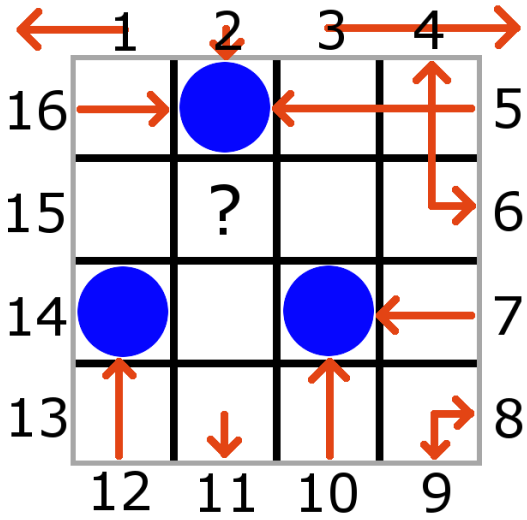
- How does a ray behave once it enters the box?
  - Miss: No interference at all
  - Hit: Heads-on collision with marble
  - Deflection: If a ray enters one of the diagonally adjacent spaces of a marble, it gets turned 90° away from the marble
- All these are not unambiguous, e.g. a ray can be deflected multiple times and leave the board as if it was a Miss



- Peculiarity: Rays can be deflected instantly in such a way that they miss the board. Such rays are missing from the input
- This is not in the standard rules and makes things easier by rendering some otherwise indistuingishable boards distinguishable



- Some boards may have identical feedback
- Mostly because they may have marbles hiding some positions



- Given full set of feedback as input
- Each input line represents a ray: Entry and exit
  - First line is pair of board dimension and marble count
- Goal: Determine placement of atoms – or marbles, in this case

Sample Input

```
5 5
1
2 6
3
4
5 7
8
9
10 11
12
13 14
15
17
18 18
19
```

- In conclusion, the problem is to construct a solution to the game for a known set of full feedback, the game dimension and number of marbles
- Keeping in mind that it‘s very hard to make definite judgments on marble placement

- Simple: Create every possible configuration of marbles on the board and check if it would produce the same input as the given one
  - Checking requires some light "raytracing": use the rules outlined before and follow every ray to where it ends up, record
  - If it doesn't match, try the next configuration
- Not very fast: Scales exponentially with board size AND number of marbles

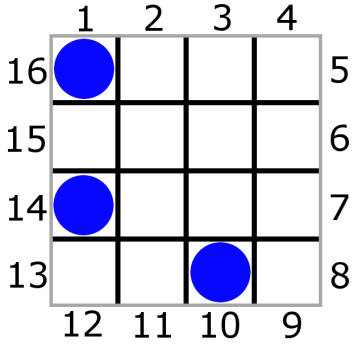
- First small optimization: If the raytracer finds a pair that‘s not in the input, the tested arrangement and the input are already not the same, so we can stop right there

→ Doesn‘t actually save that much time, since we‘re still creating and testing a lot of unnecessary arrangements

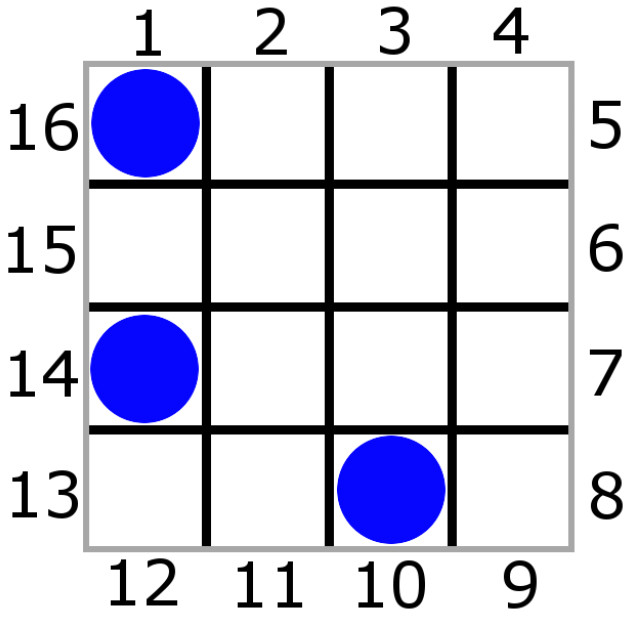
- So, how do we reduce the number of test arrangements?
- Analyse the input! – But how?
- Look at the edges:
  - A ray along an edge can never be reflected inwards
  - This means a Miss is always a true Miss here and a Detour shows where Marbles are
  - Hits are surprisingly the least informative

- When an edge is a Miss, we can treat the next parallel line of cells as an edge as well
- Reduce the board until a trace of a marble is found, then do the same for all other edges
- If a Detour is found, fix a marble at the spot that must have caused it
- If a ray never reaches the board, it means a marble has to be on one side of that entry point, but not the other

  → Introduce "XOR marbles"



- Labeling edges as containing no marbles, fixing some marbles in place and others to only two possible placements drastically reduces complexity
  - Effectively reduces both space and marble count



- That‘s all, folks