Yashi Game

Jacopo Magliani 2040912

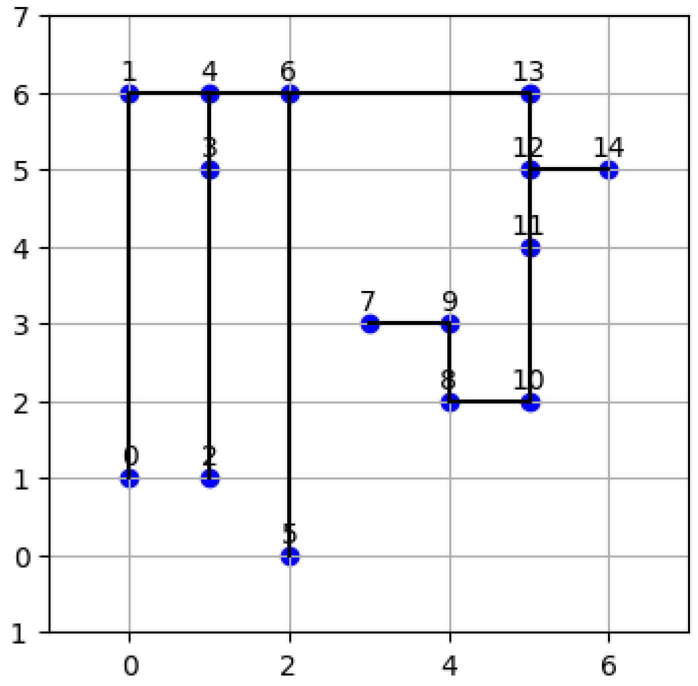
The game

Objective

- connect all the n dots

Constraints:

- no diagonal lines
- all the points must be connected
- no crossing lines
- exactly n-1 lines
- no cycles



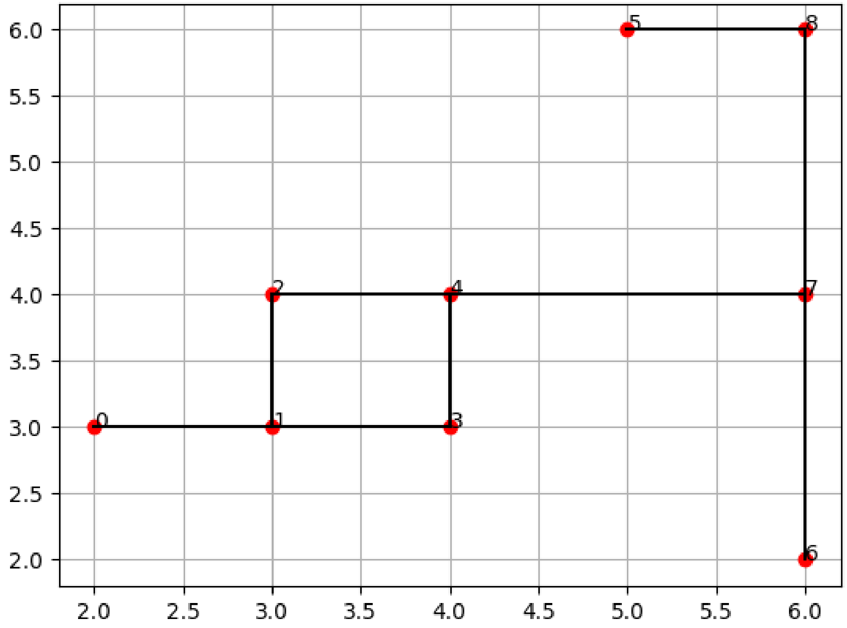
The tasks

The primary tasks to solve are:

- determine if a solution exist
- if a solution exists return it
- if a solution exists return the minimum-length solution

The initial data

ID	X	Y
0	2	3
1	3	3
2	3	4
3	4	3
4	4	4
5	5	6
6	6	2
7	6	4

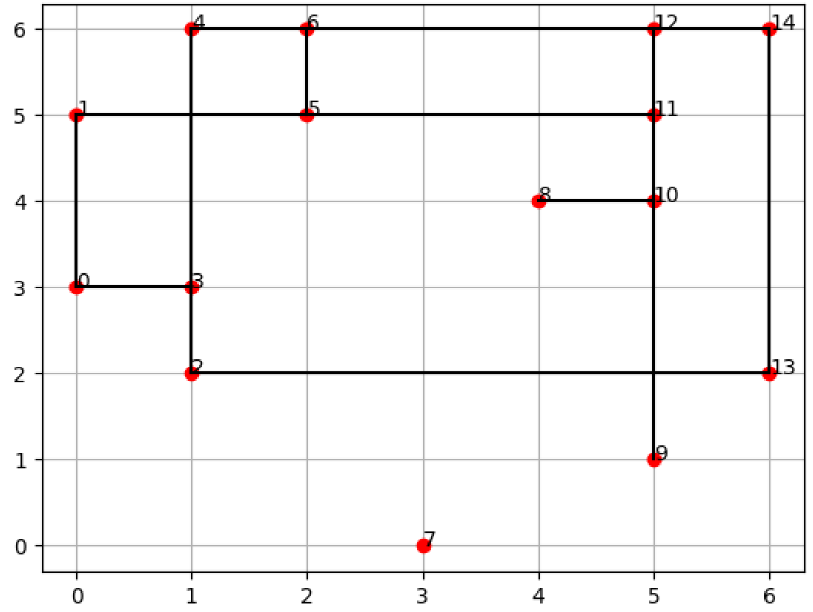


In the initialization we connect each point with its nearest neighbor in the four directions

Connection check

A recursive function visits all the nodes using the orthogonal lines connecting them.

If it does not visit all of them that means they are not all connected and the solvers do not start.



Constraints

The constraints always to respect are the absence of crossing lines, cycles and having n-1 lines

$$\phi = \phi_{no\_crossing} \wedge \phi_{n-1\_lines} \wedge \phi_{no\_cycles}$$

The literals are the identifiers of the single lines e.g. [-1] refers to the line with identifier 1

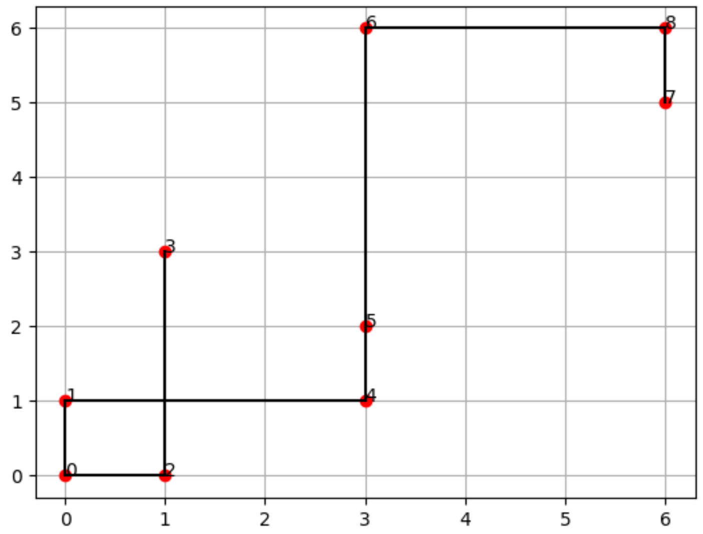
No crossing lines constraint

$$\phi_{no\_crossing} = CNF\left(\bigwedge_{\substack{l_i, l_j, l_i \neq l_j, \\ iscrossing(l_i, l_j)}} \overline{l_i \wedge l_j}\right) = \bigwedge_{\substack{l_i, l_j, l_i \neq l_j, \\ iscrossing(l_i, l_j)}} (\overline{l_i} \vee \overline{l_j})$$

If two lines are crossing I cannot use both of them.

I define a function that determines if two lines are crossing checking the maximum and minimum values of their extreme points.

I do this for every possible couple of lines, then I obtain a clause as [[-1,-2],[-3,-4]]



n–1 lines constraint

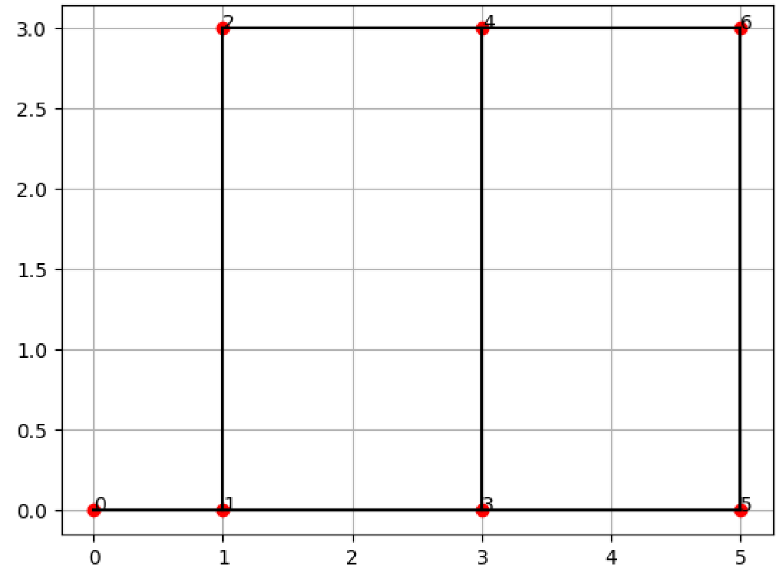
The constraint is the conjunction of “at least k” and “at most k” constraints

$$\phi_{n-1\_lines} = \left(\bigwedge_{\substack{I\subseteq[n] \\ \#I=n-k+1}} \bigvee_{i\in I} l_i\right) \wedge \left(\bigwedge_{\substack{I\subseteq[n] \\ \#I=k+1}} \bigvee_{i\in I} \bar{l_i}\right)$$

where n is the total number of lines and k is the number of points -1

No cycles constraint

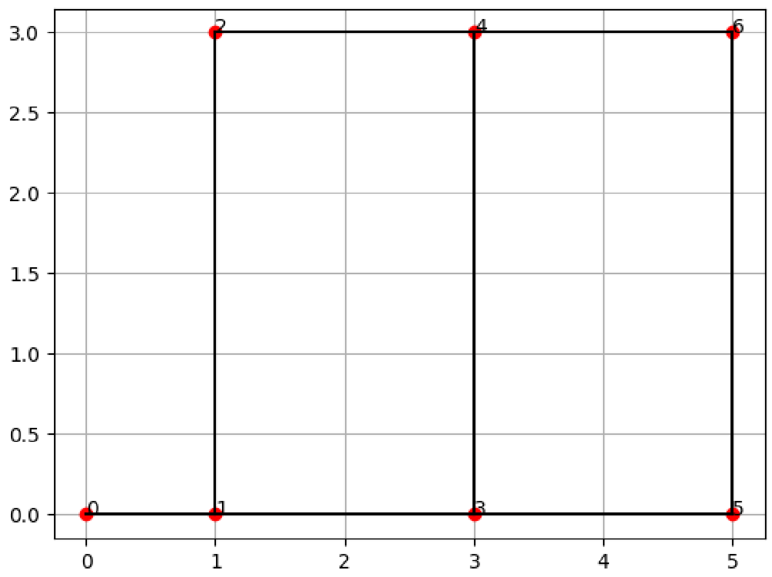
$$\phi_{no\_cycles} = CNF(\bigwedge_{c \in Cycles} \overline{\bigwedge_{l \in c} l}) = \bigwedge_{c \in Cycles} \bigvee_{l \in c} \bar{l}$$



I define a function that finds all the possible clauses of lines that can form a cycle.

In this example it outputs 3 causes because we have 3 possible cycles: 2 cycles of 4 lines sharing the line in the middle and 1 cycle with all the lines of the other cycles but not the line in the middle

No cycles constraint



The function first looks for all the points with more than 1 line and obtains a set of possible dangerous lines.

Then it obtains all the possible subsets of it and for each of these subsets it obtains the points connected to the lines of the subset.

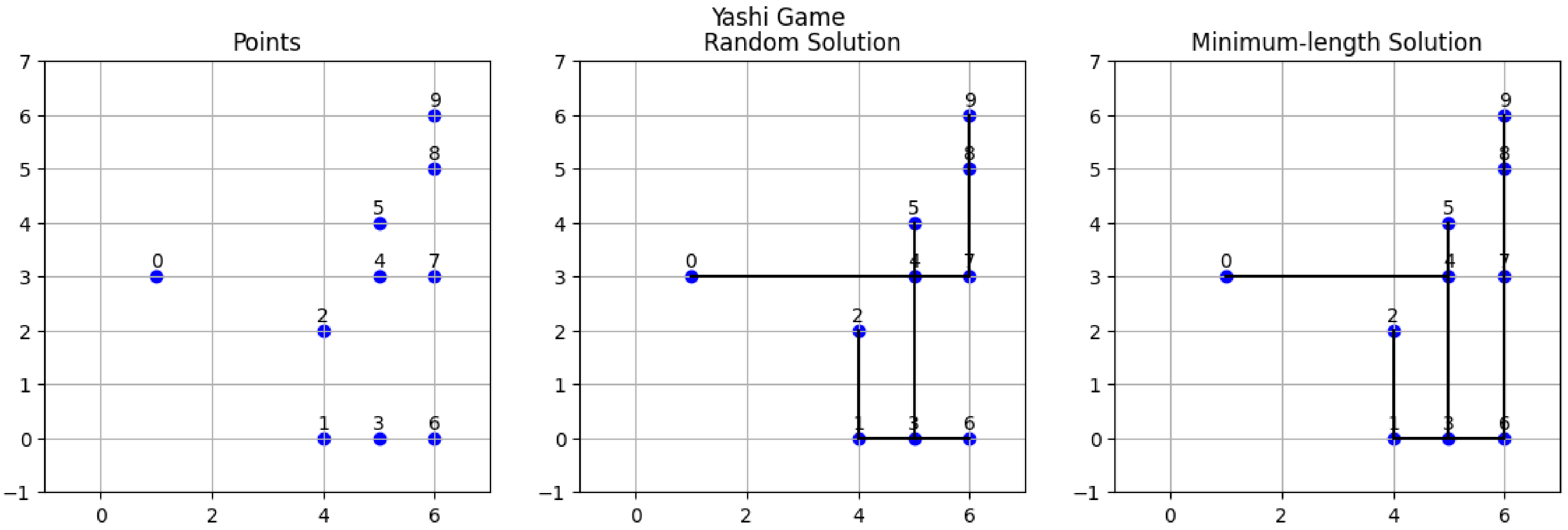
If starting from a point we can iterate through the others with direct connections and return to the starting point, that means the subset of lines is a new clause of lines of a cycle

Solver

The solver:

- FIrst initializes the game with the received dataframe of points
- Determines if the points are all orthogonally connected.
- If so it uses hard constraint to search a first solution
- It counts how many total solutions there are
- It uses soft constraints to determine the minimum-length solution, where each segment of a line has weight 1.
- Returns the plots of the points, of a solution, of the minimum-length solution and how many solutions in total there are
- If the points are not connected or there is no solution it plots just the points.

Solver



Number of solutions: 4 Cost of the solution: 15.0

Thanks

Thank you for the attention

Jacopo Magliani