PROJECT PRESENTATION

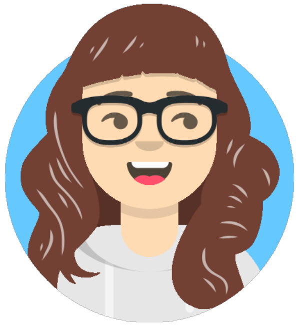
Artificial Intelligence Fundamentals

Academic year 2022/2023

THE SNAKERS TEAM

Giacomo Aru

Gabriele Benanti



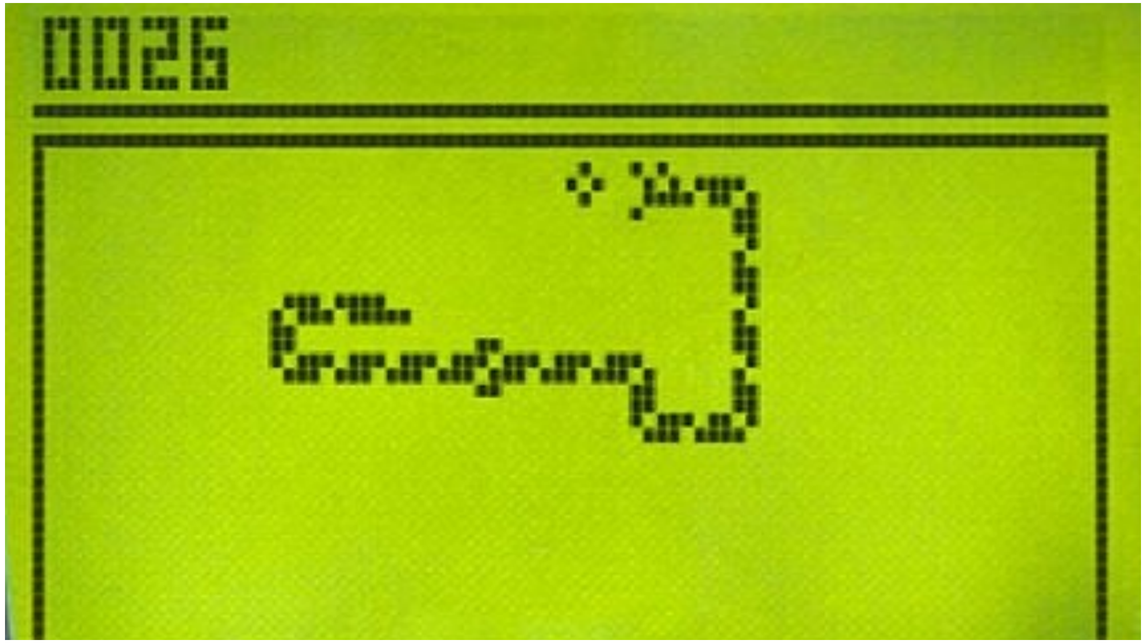
Giulia Ghisolfi

Luca Marini

Irene Testa

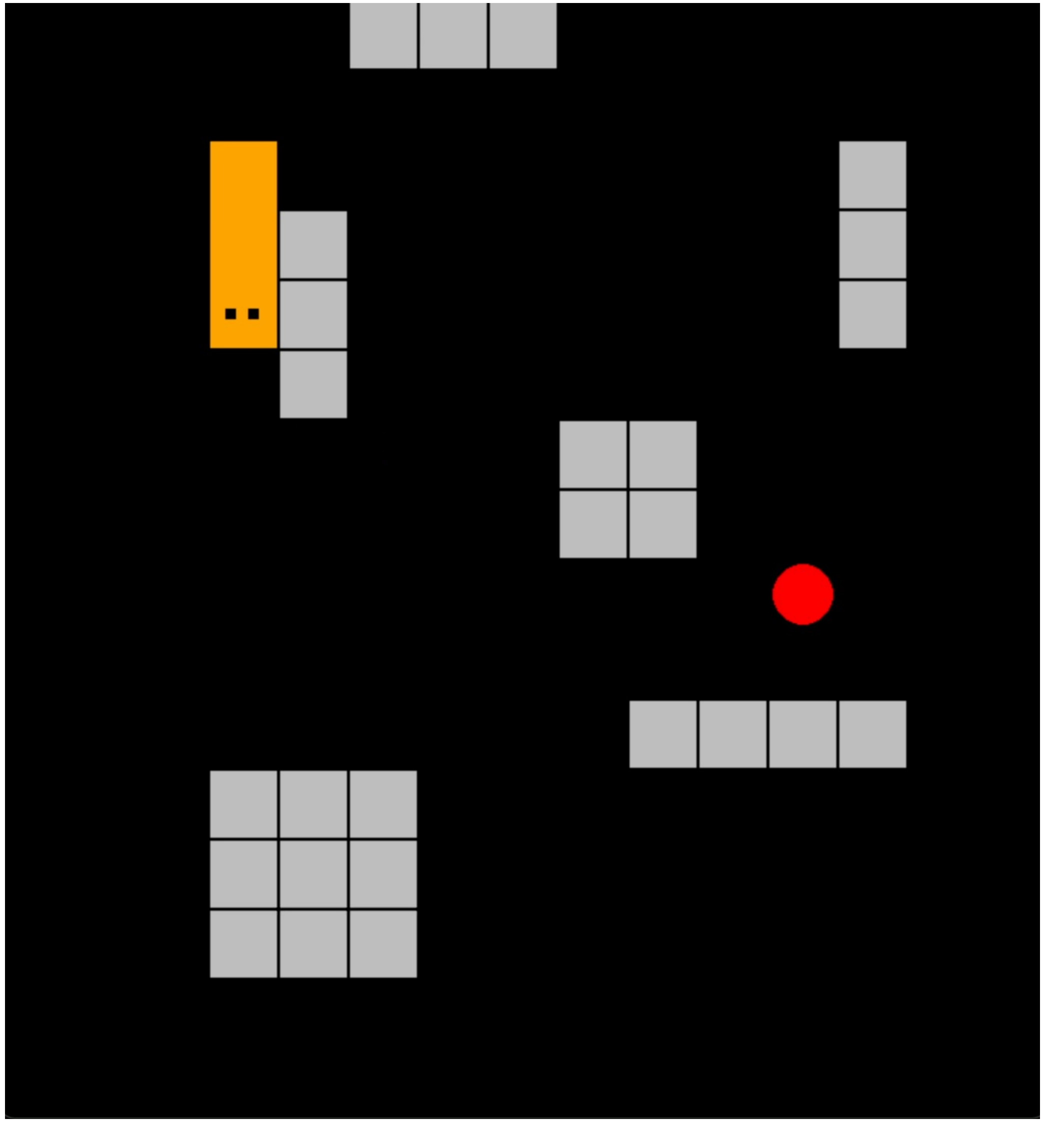
HISTORY OF THE GAME SNAKE

- Snake was inspired by the arcade game *Blockade*, in 1976.
- The first single-player version was introduced in 1982.

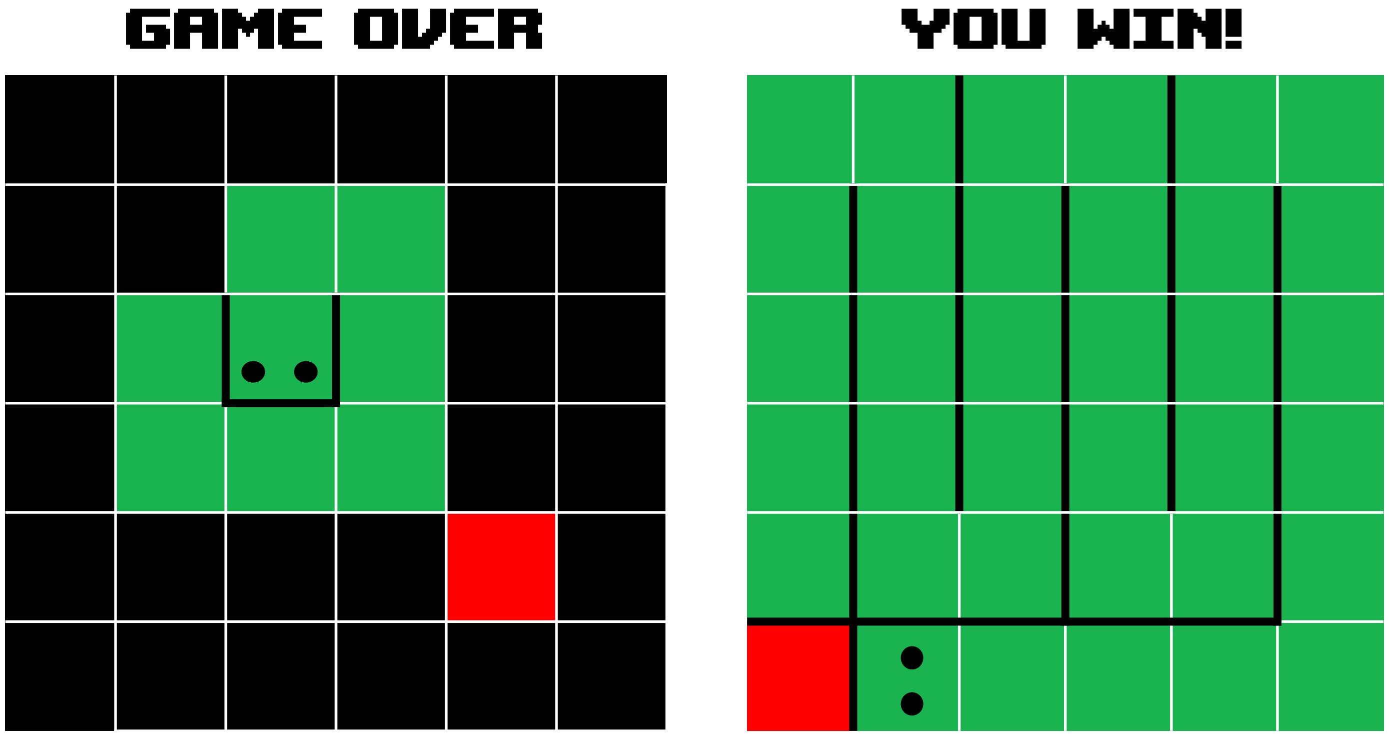


AIM OF THE GAME

- Starting from a prefixed position, try to eat apples as much as possible in order to cover all the grid (when the snake eats an apple its body lengthens by one unit)

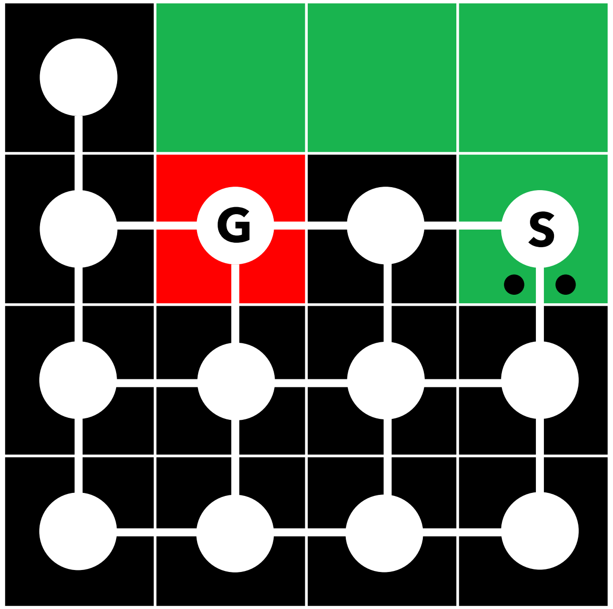


RULES OF THE GAME



- To survive, the snake must not hit neither the obstacles in the grid, the frame of the grid nor its own body

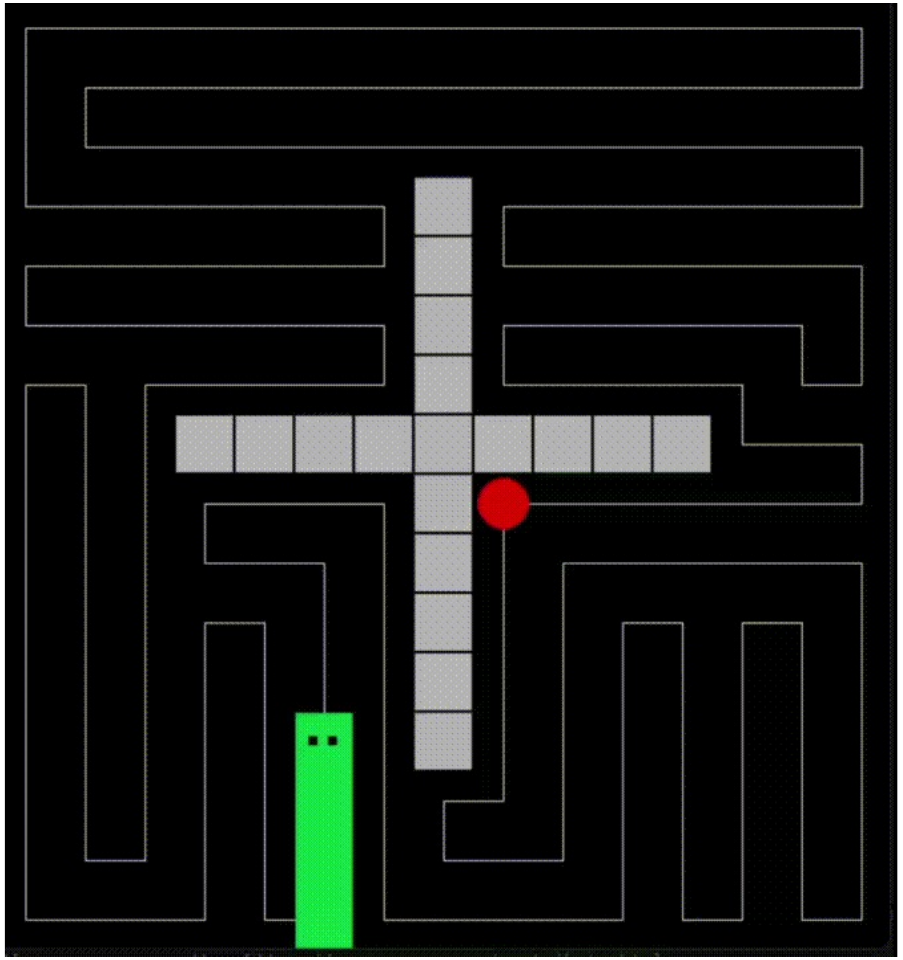
GAME SPACE: A GRAPH



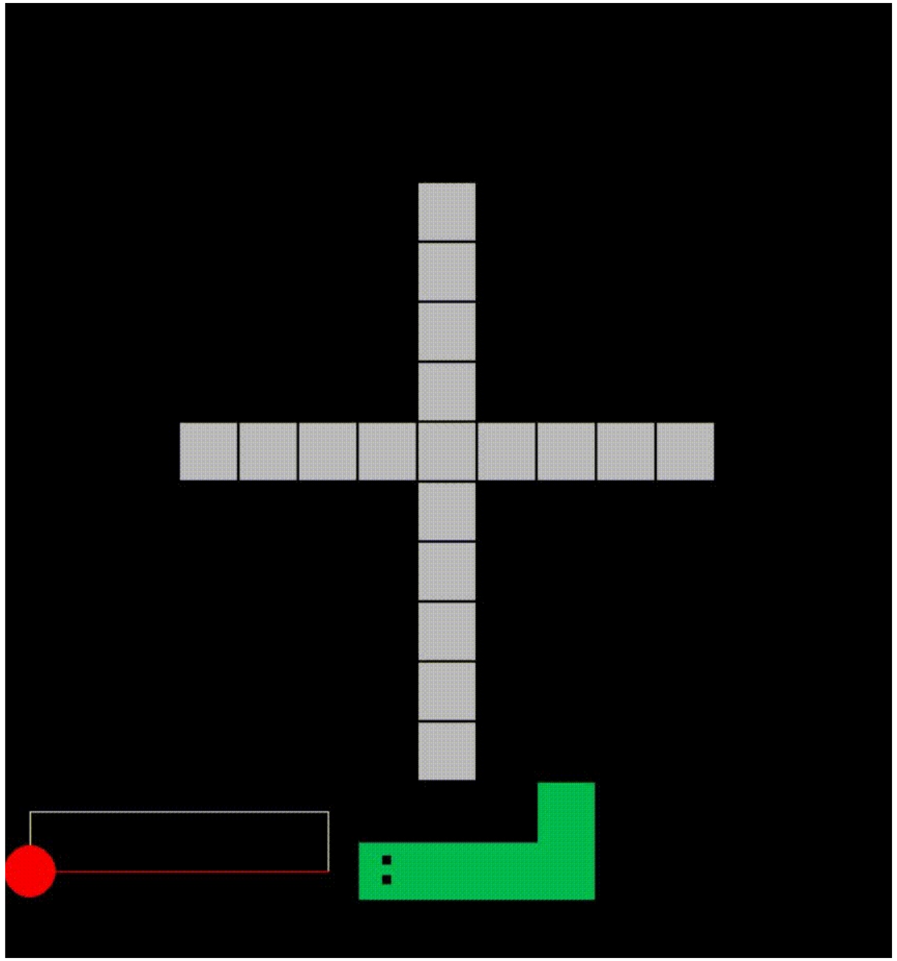
S: start, G: goal

- Self-contained problem where all possible events and outcomes are known. Ideal to apply the AI methodologies we have seen in the course.
- The grid in which the snake moves is actually a solid graph. Each block corresponds to a node in the graph.

STRATEGIES



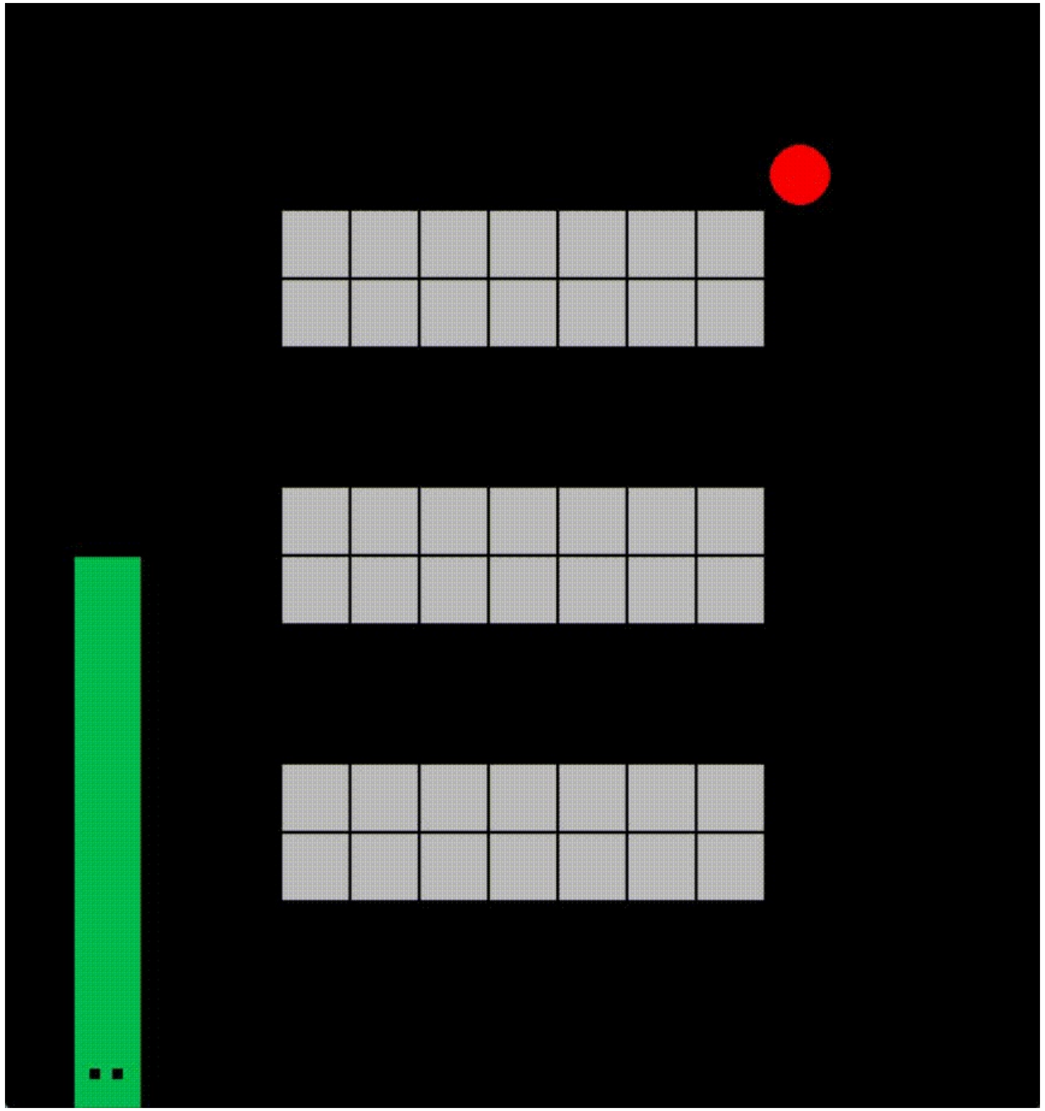
Hamilton strategy



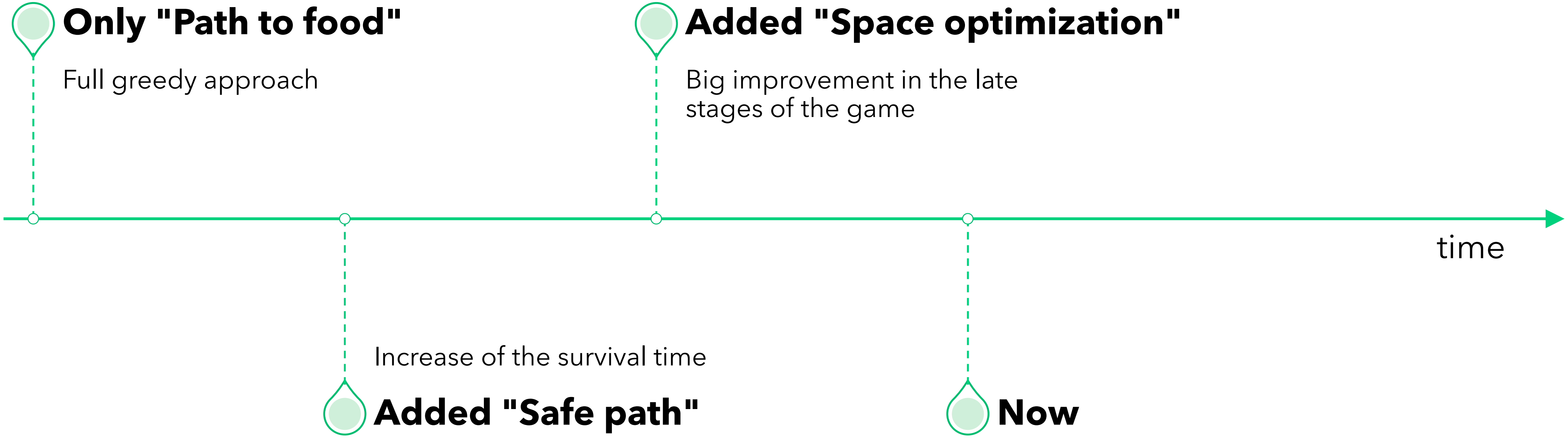
Greedy strategy

GREEDY STRATEGY - WHY?

- the easiest to implement but probably the hardest to perfect
- suitable for any type of grid and obstacle configuration
- requires little computational time
- takes full advantage of search algorithms



3 SUB-STRATEGIES



Only "Path to food"

Full greedy approach

Added "Safe path"

Increase of the survival time

Added "Space optimization"

Big improvement in the late stages of the game

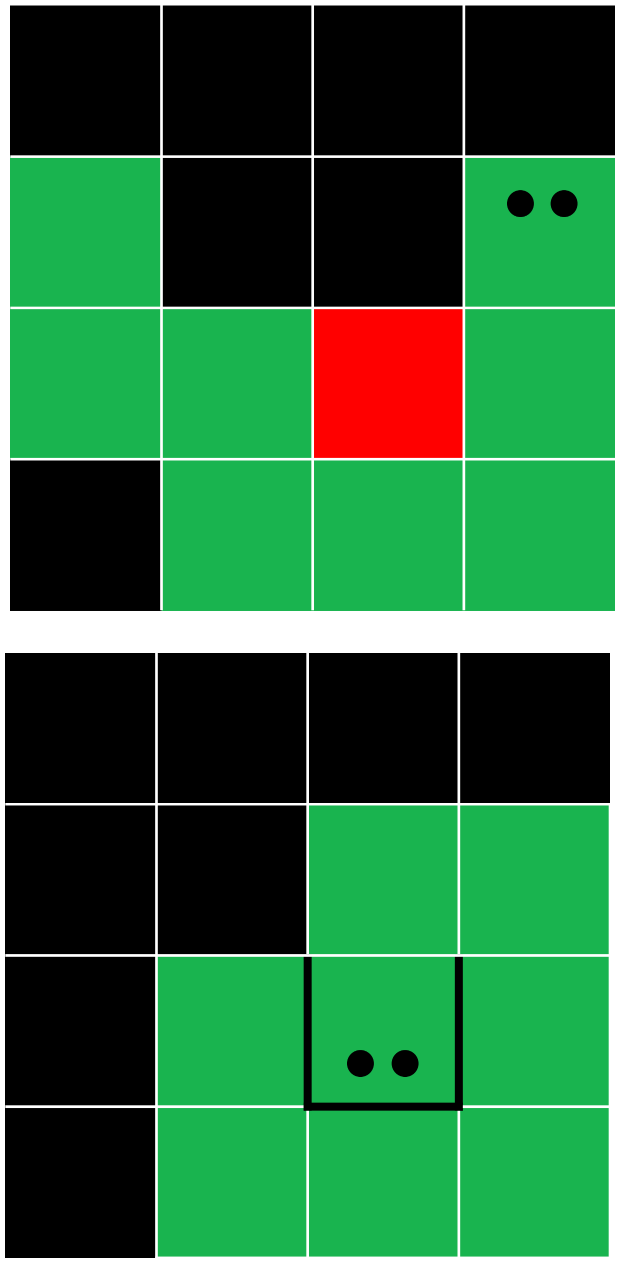
Now

time

PATH TO FOOD

- Every game tick the snake tries to find a path between its head and the food

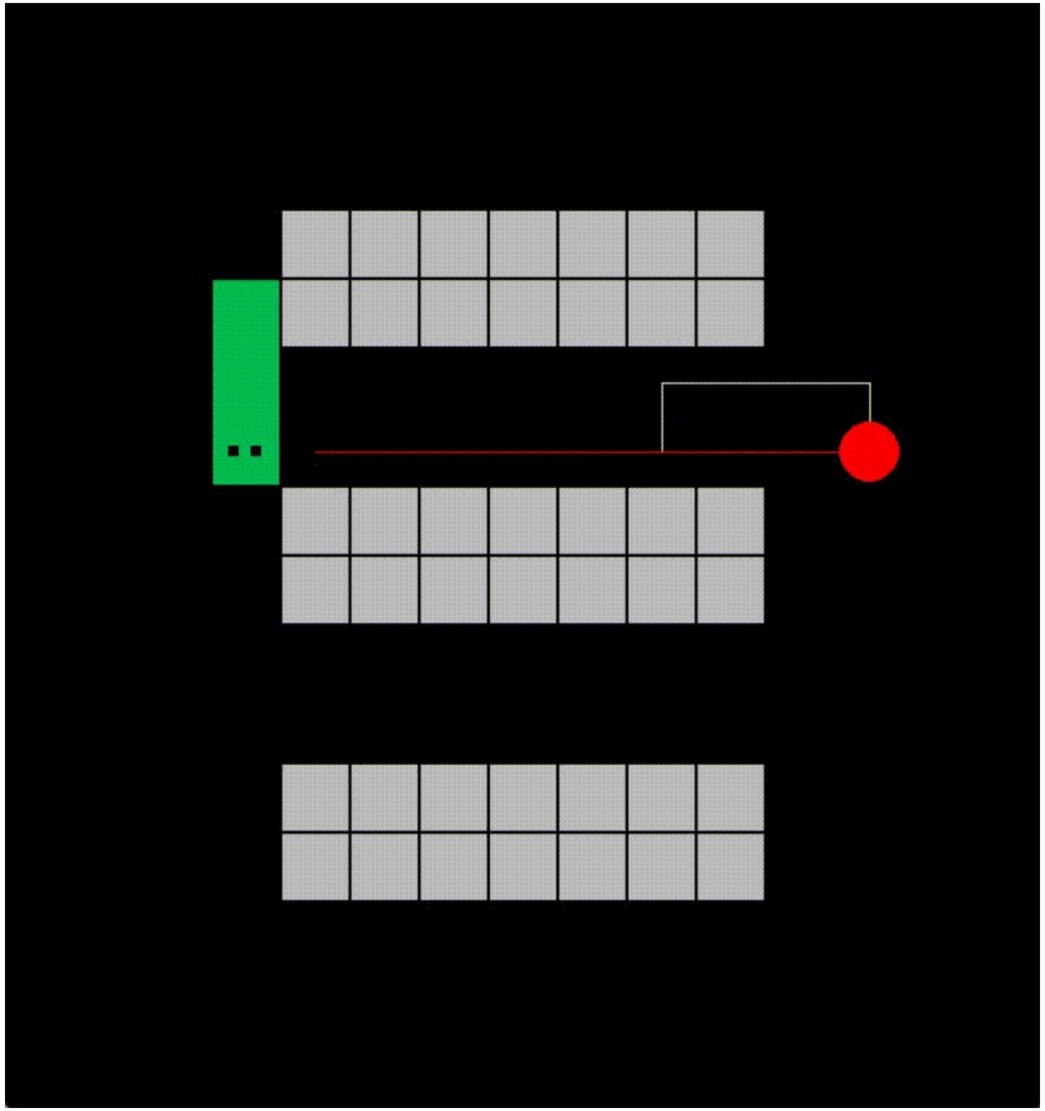
- Obvious problem of early death



SAFE PATH

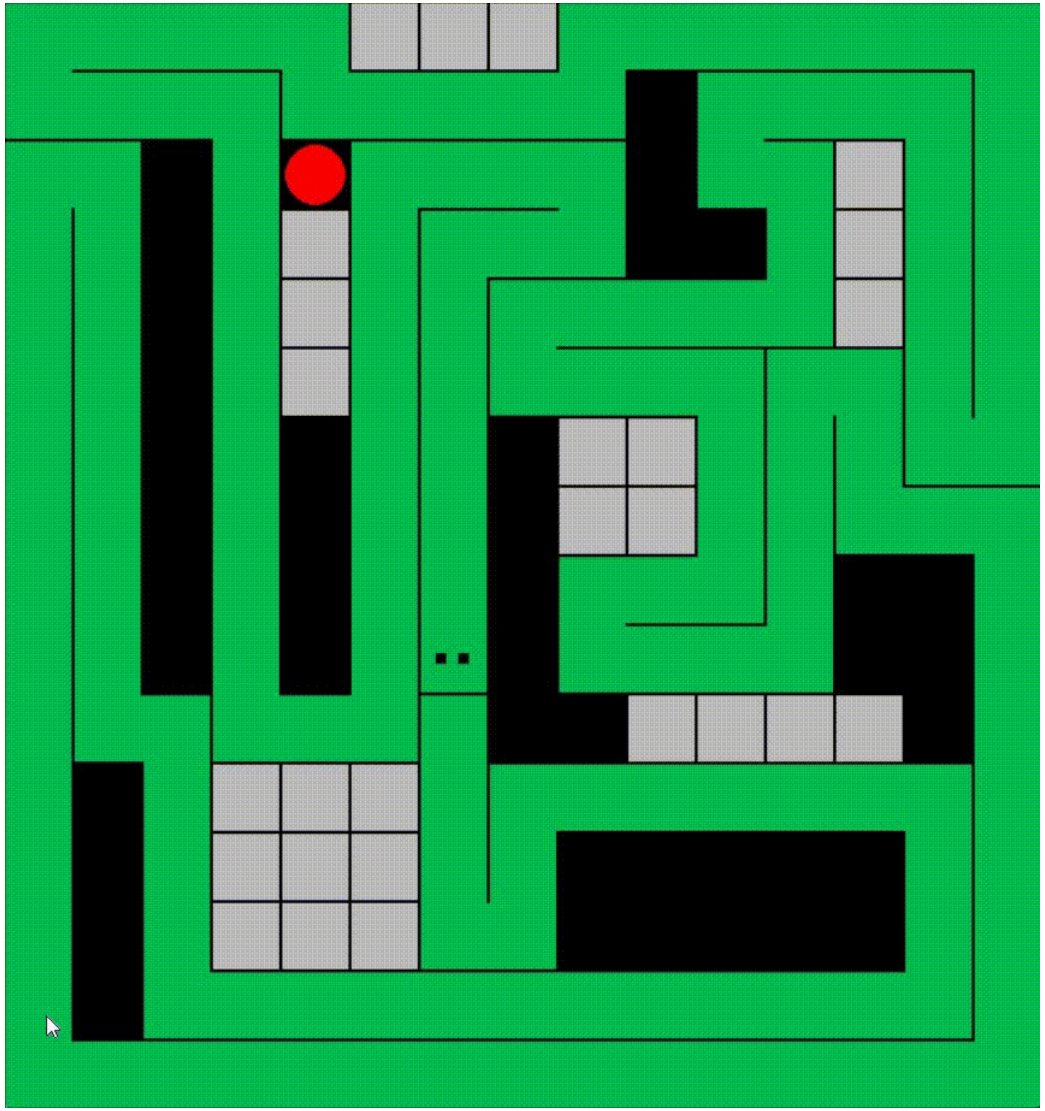
After having computed the path to the food, the algorithm checks if there is a safe cycle

- Red line: "Path to Food"
- Yellow line: "Safe Path"



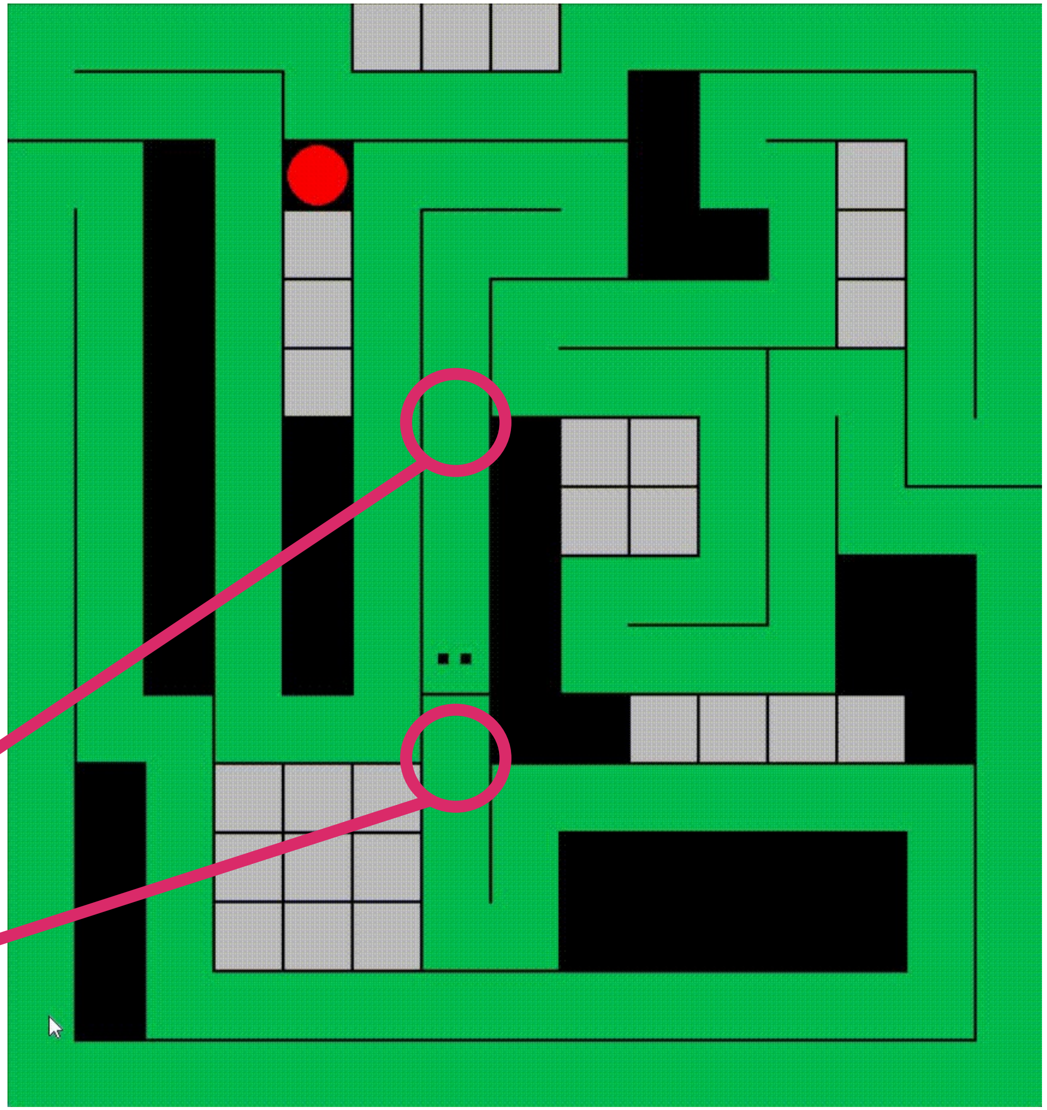
SAFE PATH

- Due to the lack of attention to the snake's position, the grid tends to become full of small scattered empty spaces
- In this situation the bot is stuck in an infinite loop



SPACE OPTIMIZATION

- Every time the bot can't find the shortest path to the food, it tries to optimize the safe path it is following checking for the next optimizable cell and computing the longest path between that cell and the next choke point

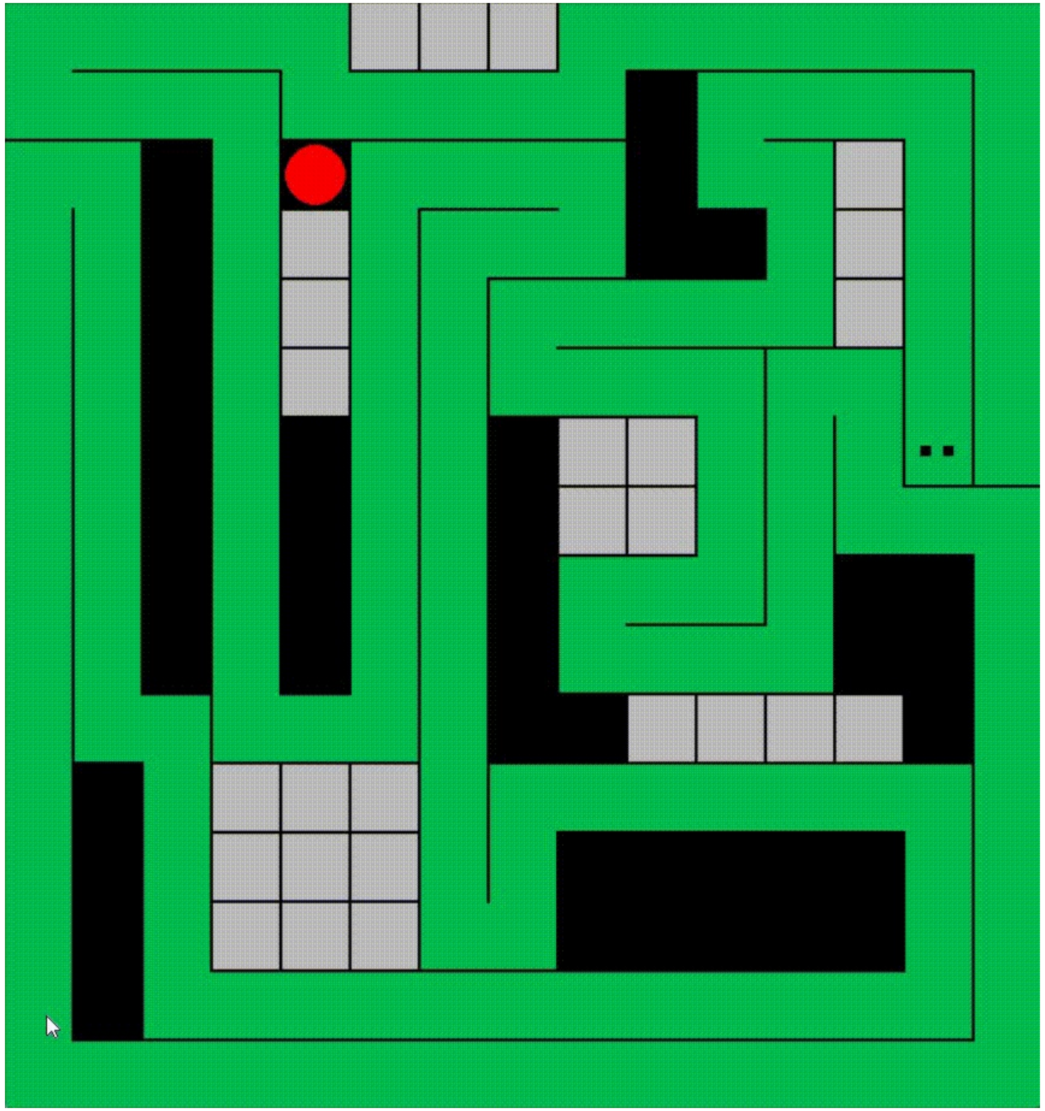


Optimizable cell

Chokepoint

SPACE OPTIMIZATION

- Longest path ≠ Hamiltonian cycle
- With this optimization the bot always reaches "pseudo-terminal" states

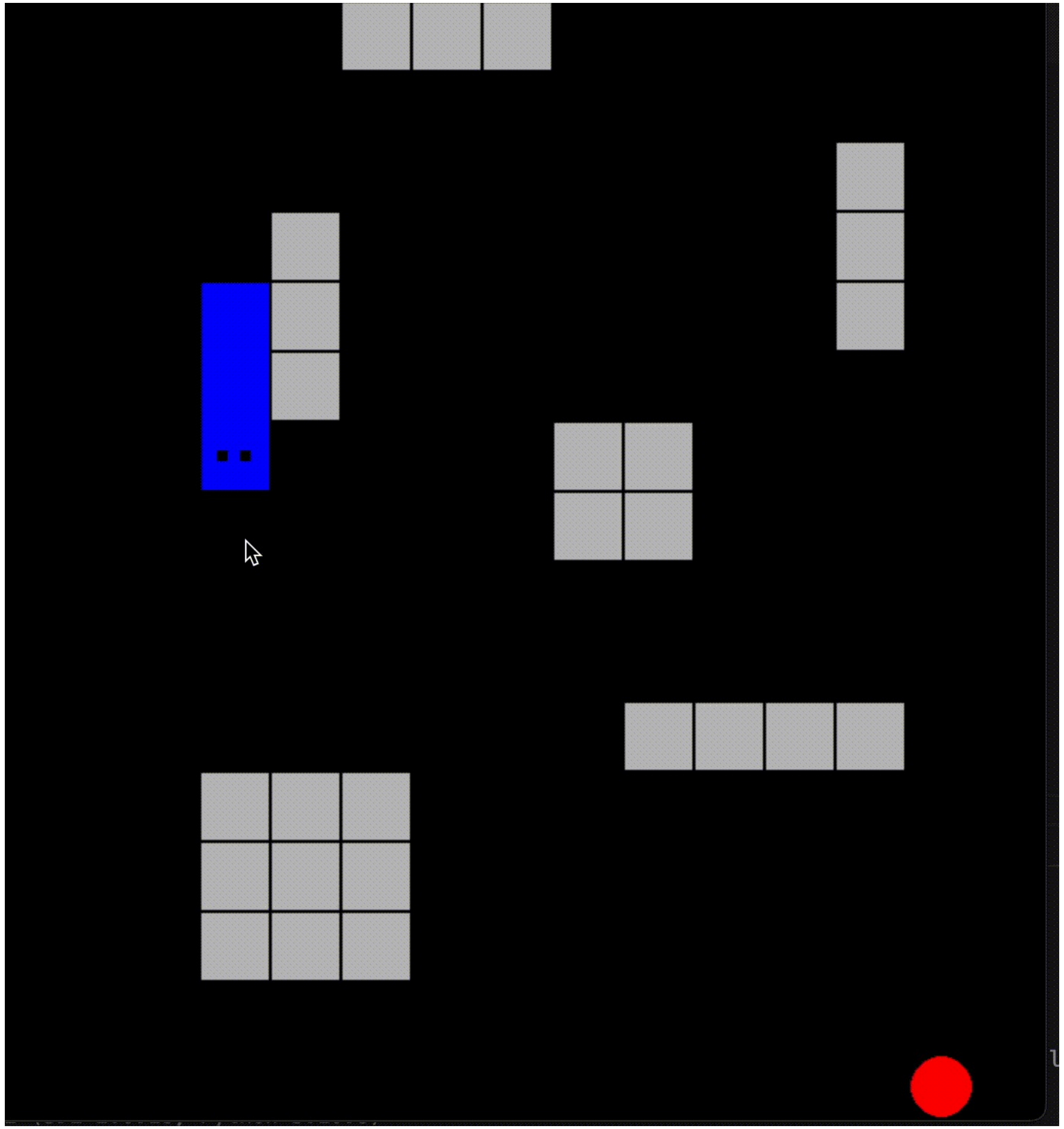


USED ALGORITHMS

- A*
- Longest Path

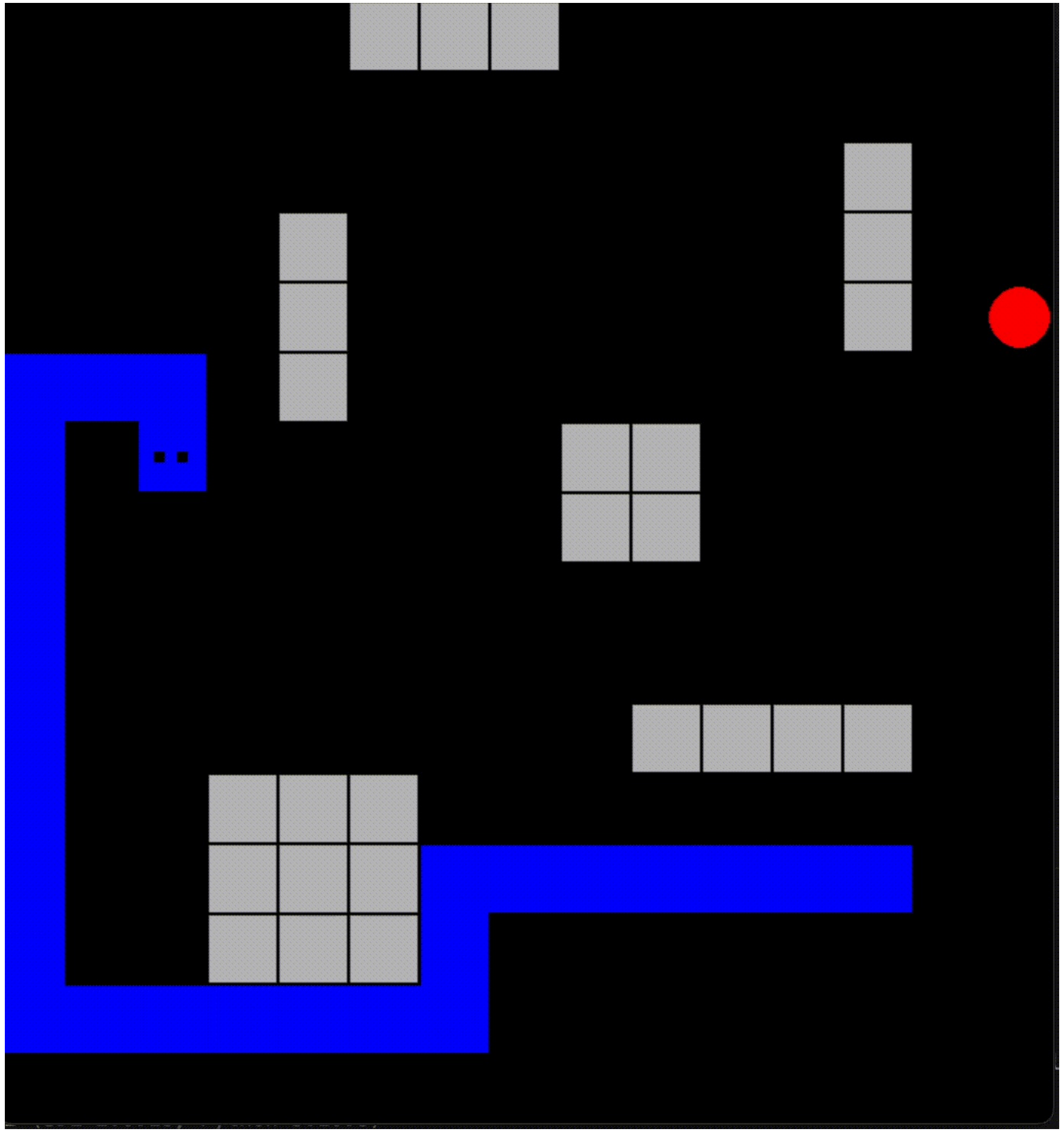
A*

- **Manhattan distance** as the main heuristic (best underestimation + avoids zigzagging paths)
  - **Min-turns:** storing the direction of the snake in each node
  - **Save space:** expanding first the nodes with less neighbors



A*

- **Manhattan distance** as the main heuristic (best underestimation + avoids zigzagging paths)
  - **Min-turns:** storing the direction of the snake in each node
  - **Save space:** expanding first the nodes with less neighbors



LONGEST PATH PROBLEM

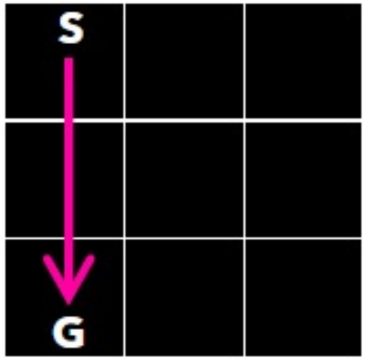
- Find a **simple path** of **maximum lenght**

- NP-complete :(

  ⇨ Longest Path Heuristic algorithm

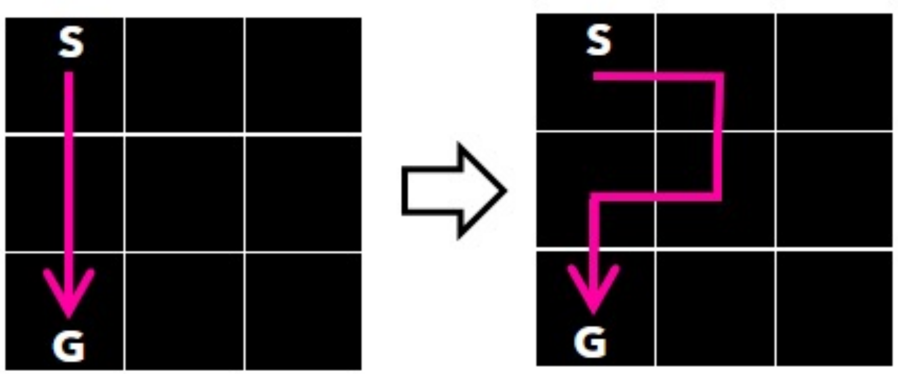
LONGEST PATH HEURISTIC ALGORITHM

1. Find the shortest path (A*)



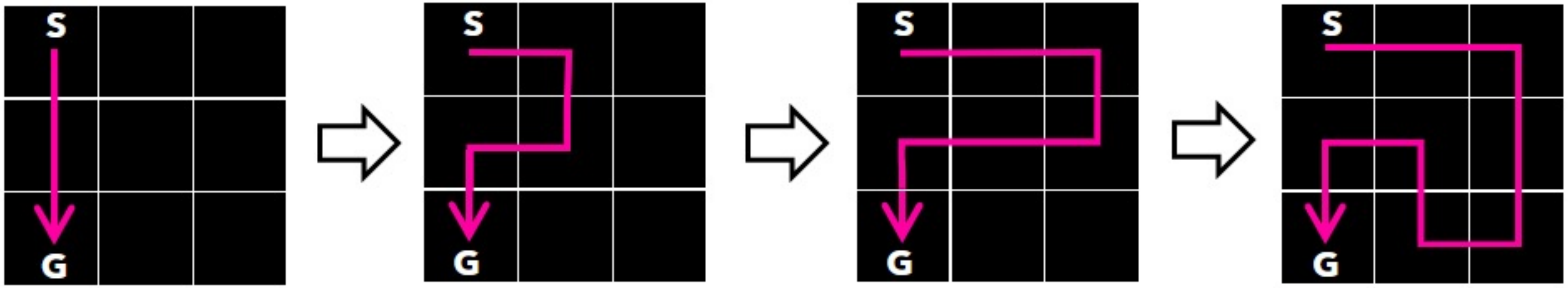
LONGEST PATH HEURISTIC ALGORITHM

1. Find the shortest path (A*)
2. Extend a pair of path pieces with another pair which is not currently included in the path



LONGEST PATH HEURISTIC ALGORITHM

1. Find the shortest path (A*)
2. Extend a pair of path pieces with another pair which is not currently included in the path
3. Iterate until no extensions can be found

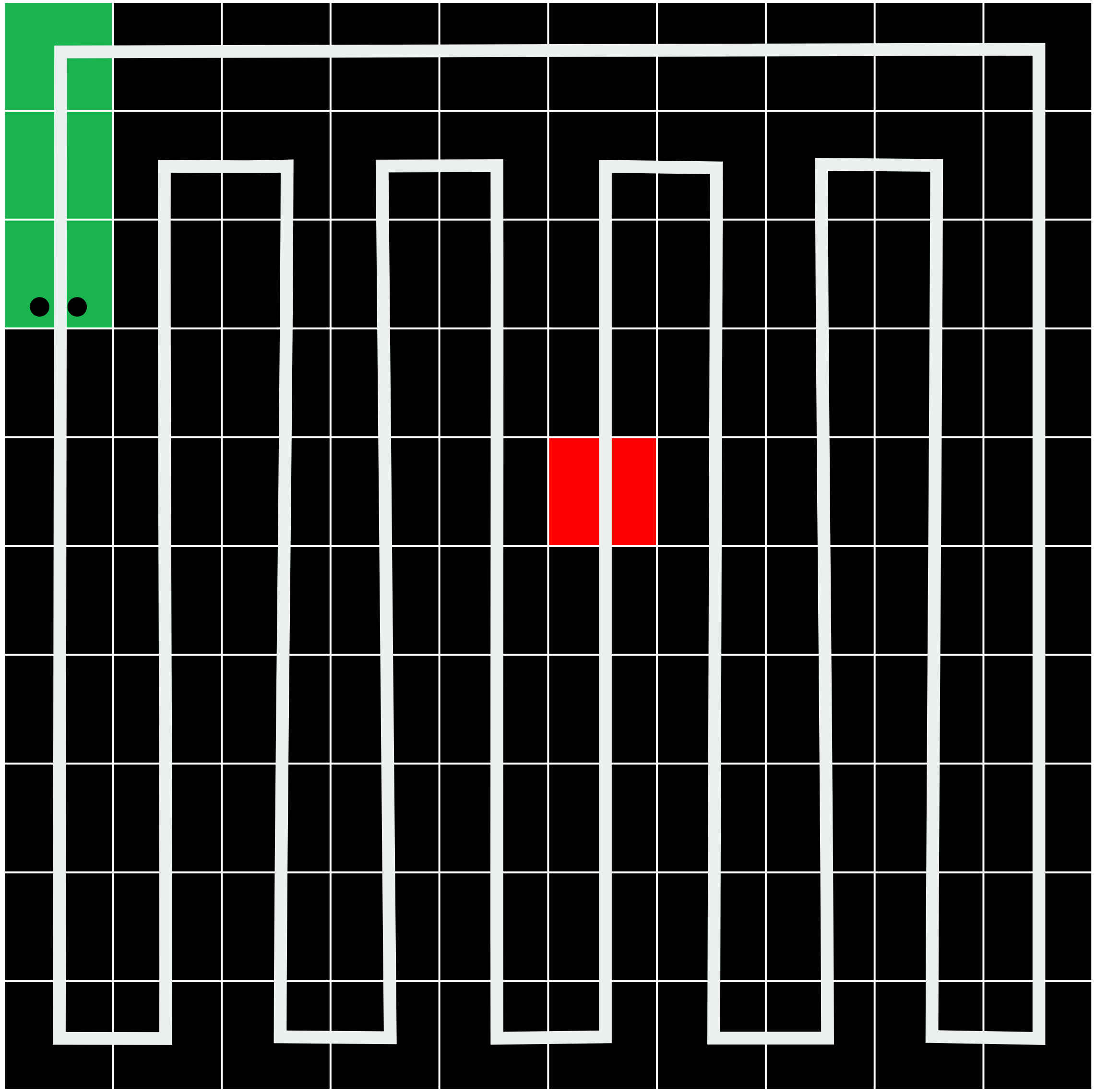


TRIVIAL WINNING STRATEGY

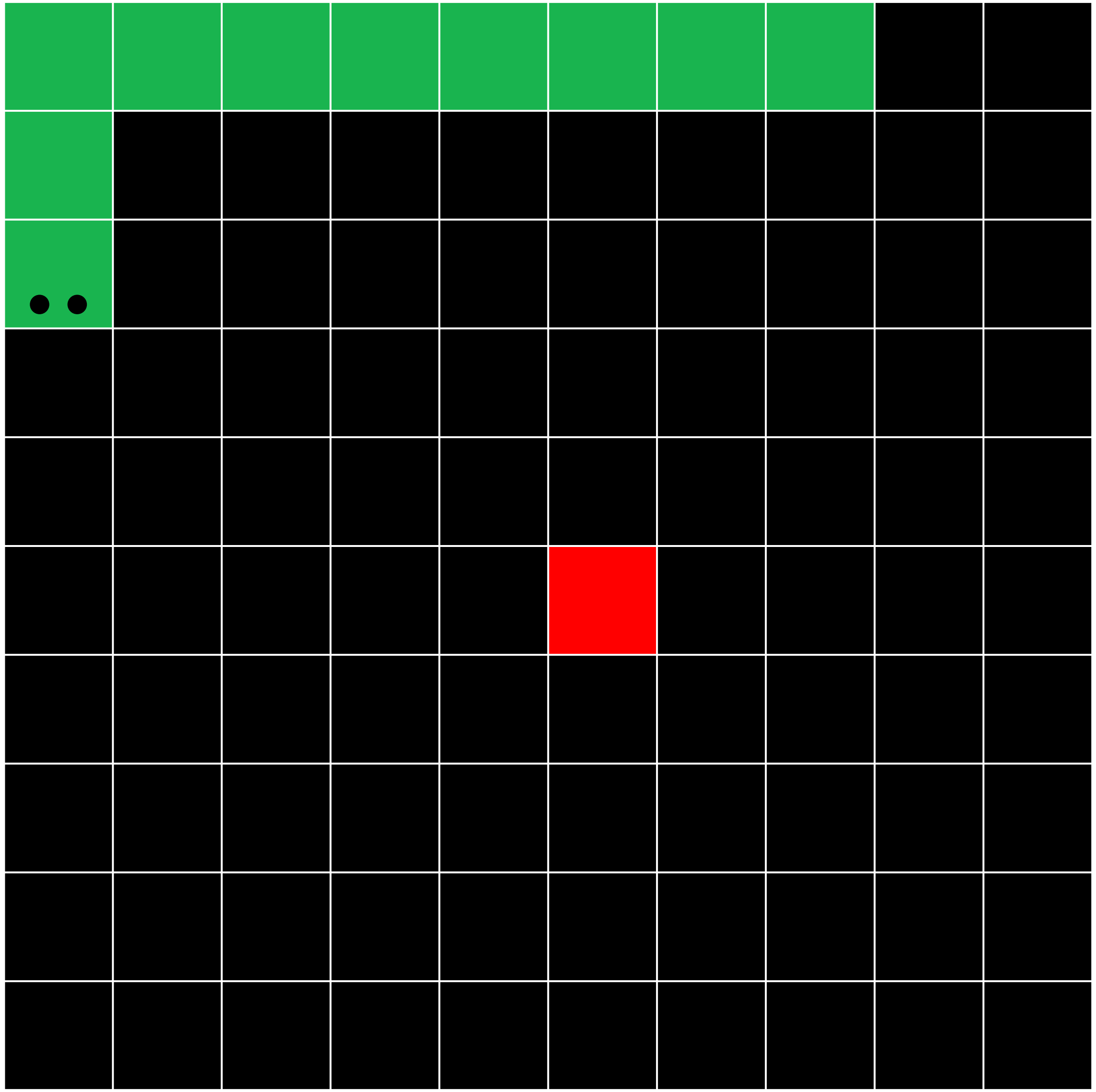
Keep following a hamiltonian cycle, i.e. a cycle that passes through every point of the board exactly once.

+ always leads to a win

- completing the game requires a high number of moves (to eat an apple the snake visits on average half of the nodes)

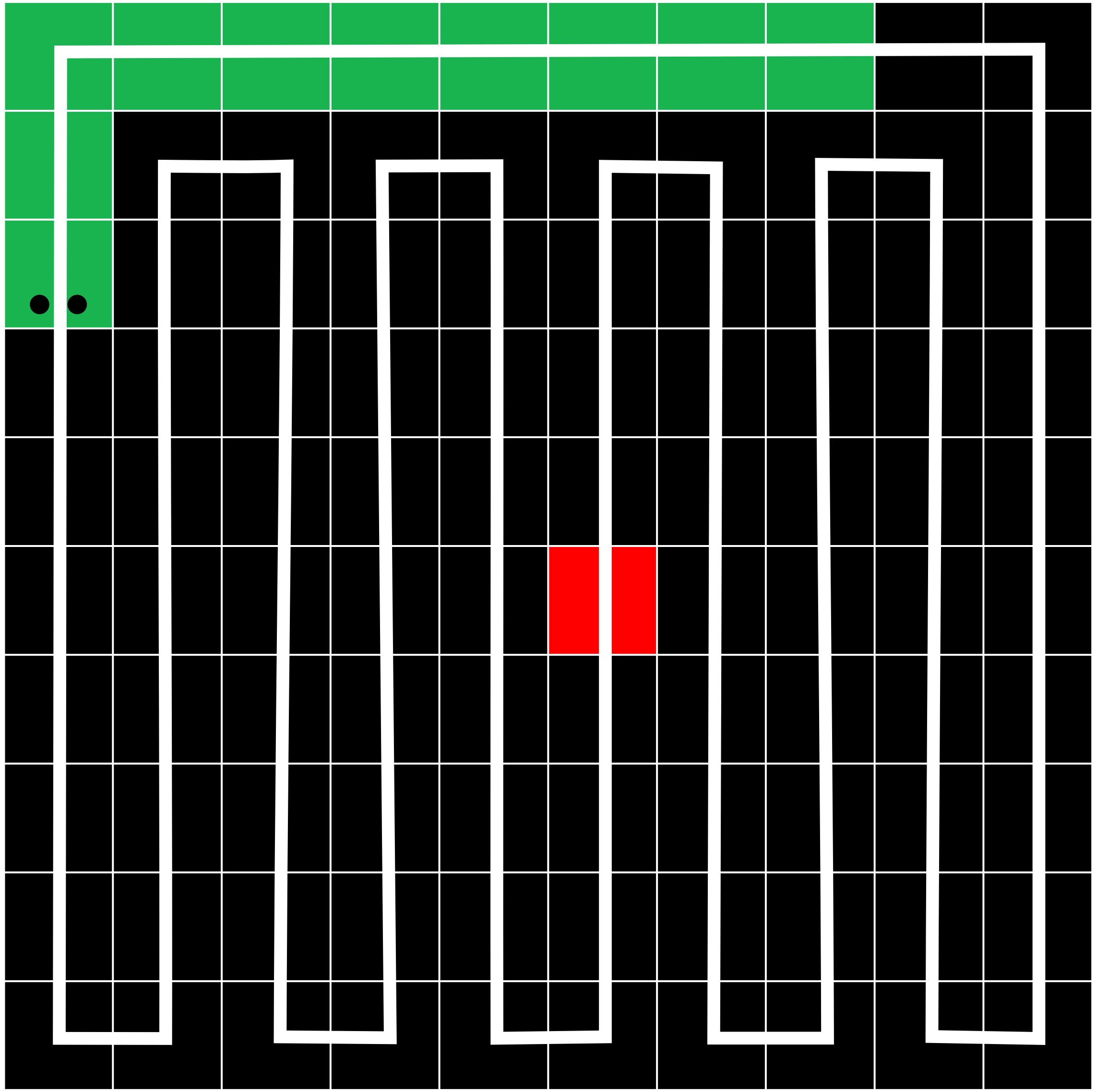


SMARTER STRATEGY



SMARTER STRATEGY

Compute a hamiltonian cycle on the grid.

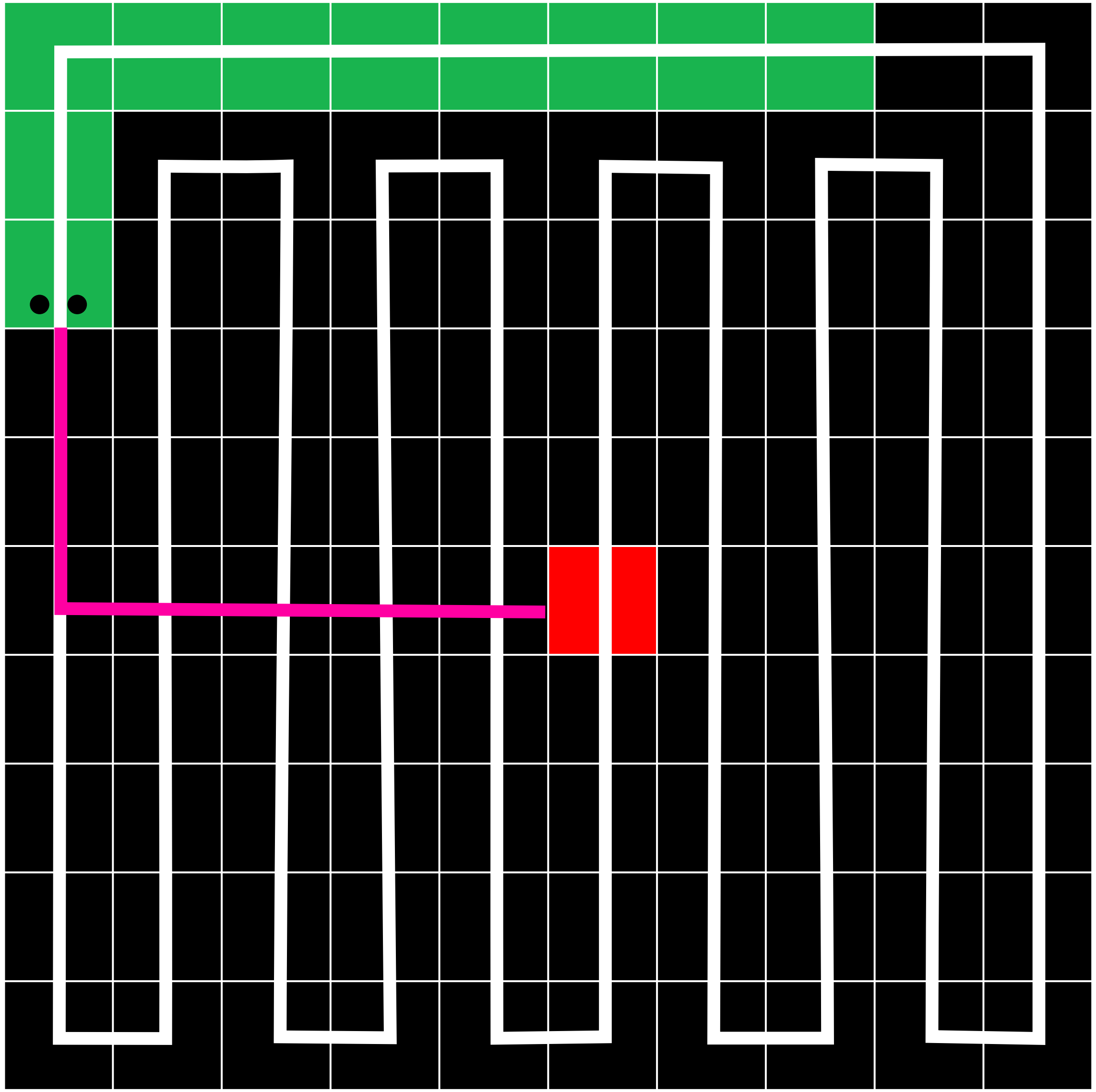


SMARTER STRATEGY

Compute a hamiltonian cycle on the grid.

For each step of the game:

    Compute the shortest path to the apple

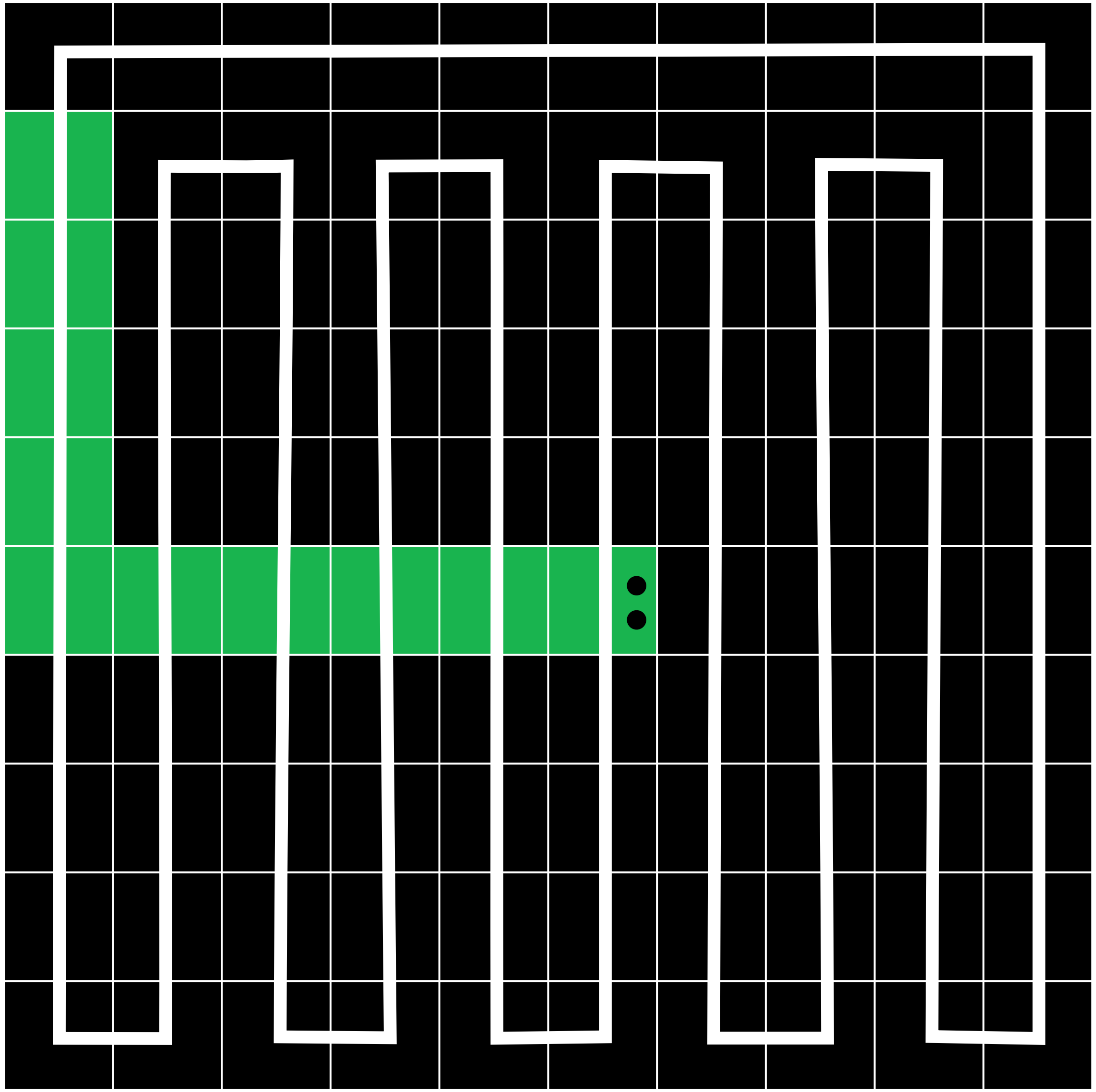


SMARTER STRATEGY

Compute a hamiltonian cycle on the grid.

For each step of the game:

> Compute the shortest path to the apple and the position of the snake once it has reached the apple through that path.

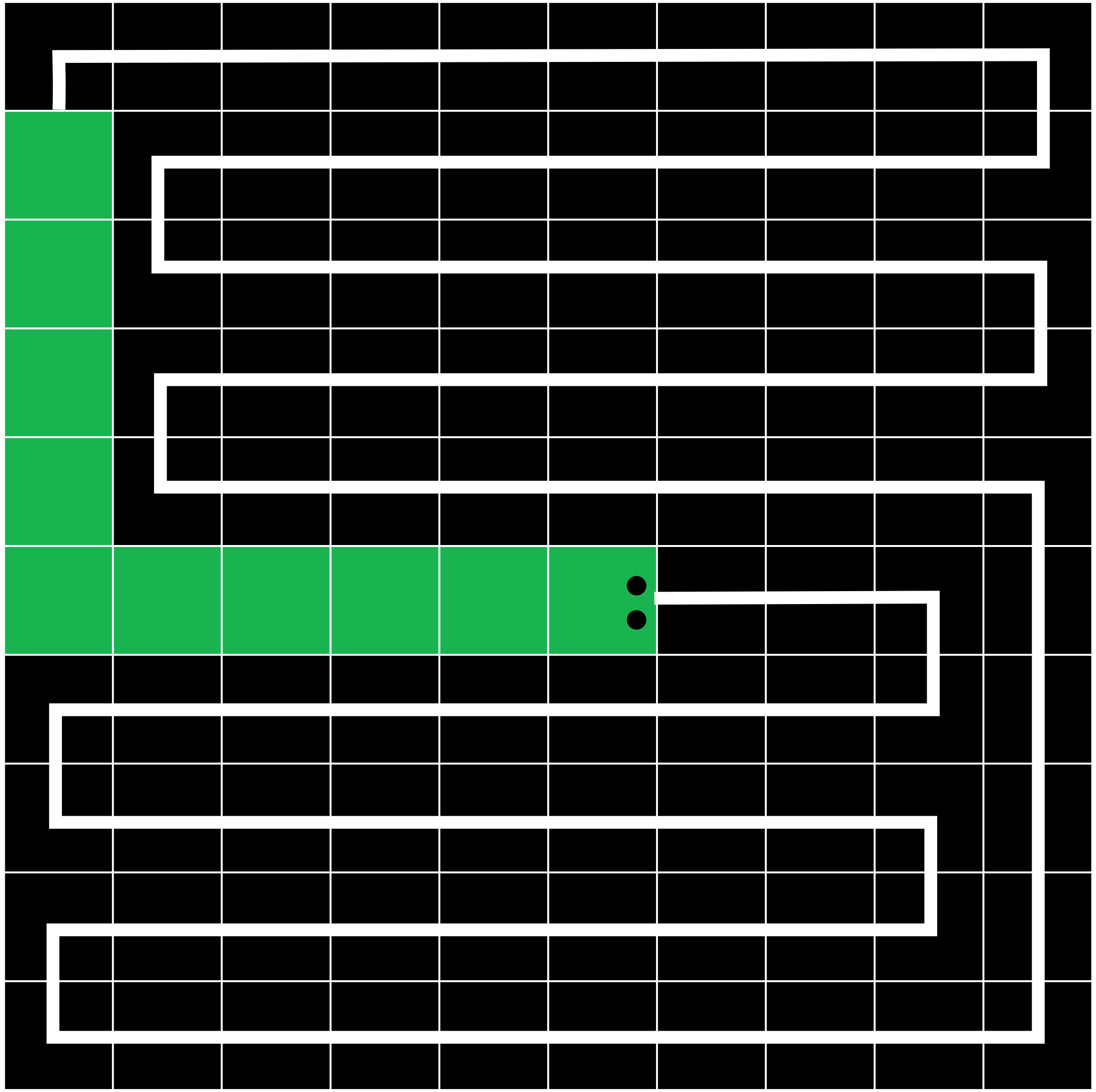


SMARTER STRATEGY

Compute a hamiltonian cycle on the grid.

For each step of the game:

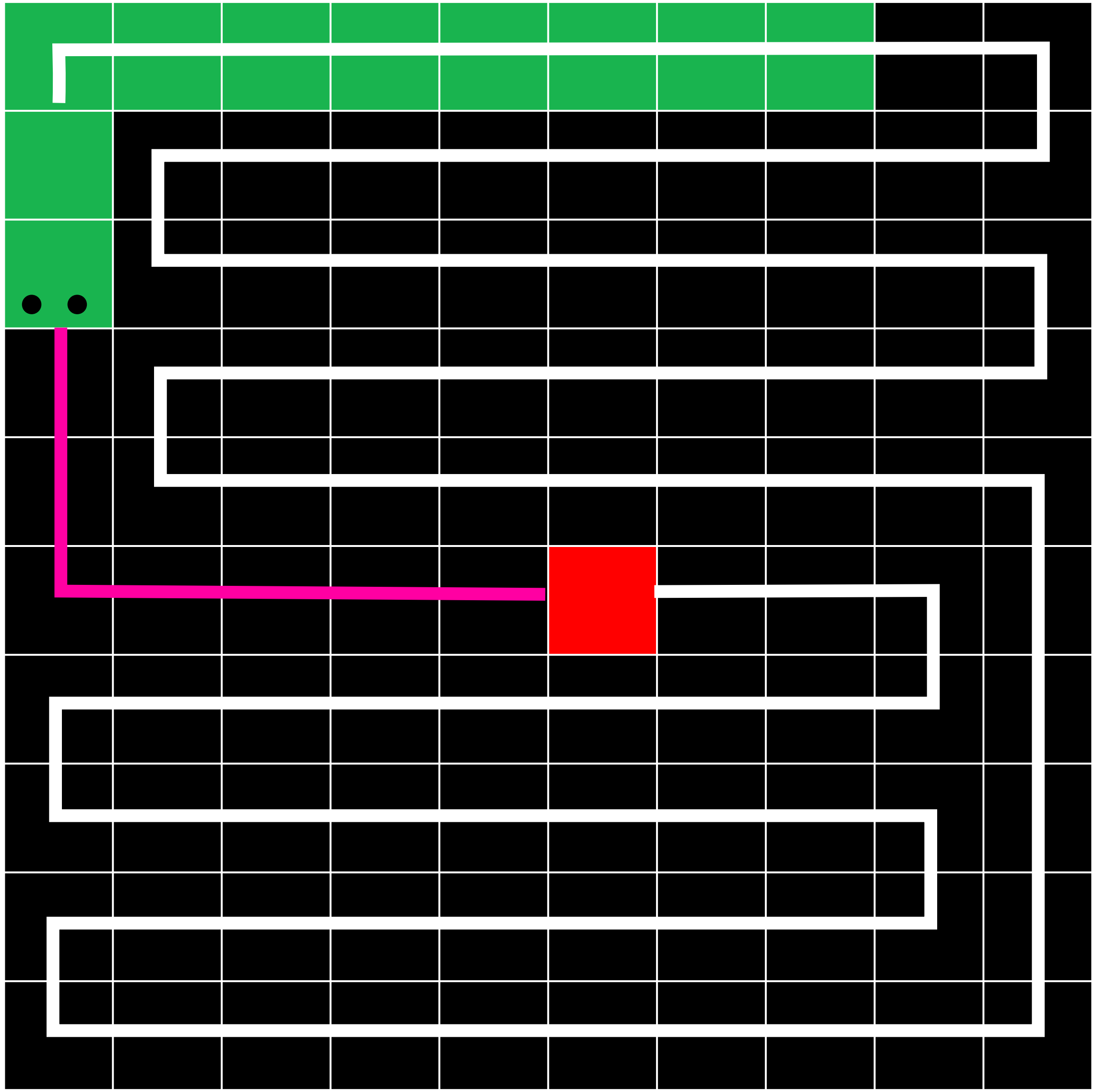
- Compute the shortest path to the apple and the position of the snake once it has reached the apple through that path.
- If there exists a hamiltonian path from the apple to the position of the tail of the snake once it has eaten the apple, then follow the shortest path and save the computed hamiltonian cycle.
- Otherwise follow the hamiltonian cycle previously computed.



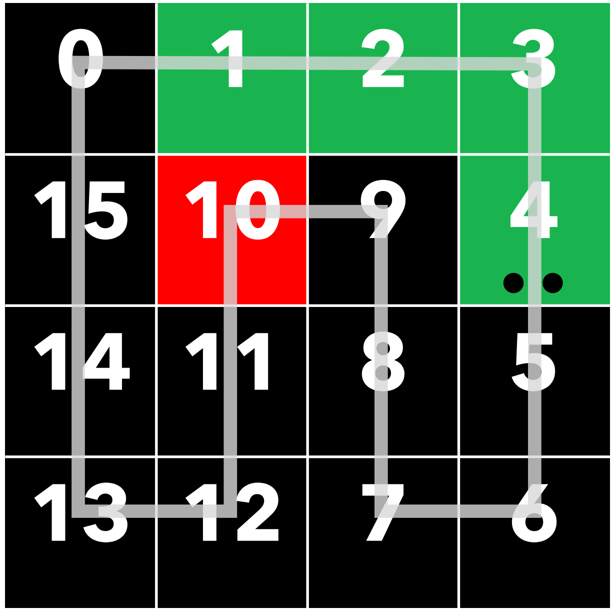
SMARTER STRATEGY

+ gets to the apple faster

- not scalable on arbitrary wide grids (deciding whether a graph contains a hamiltonian path is an NP-complete problem)

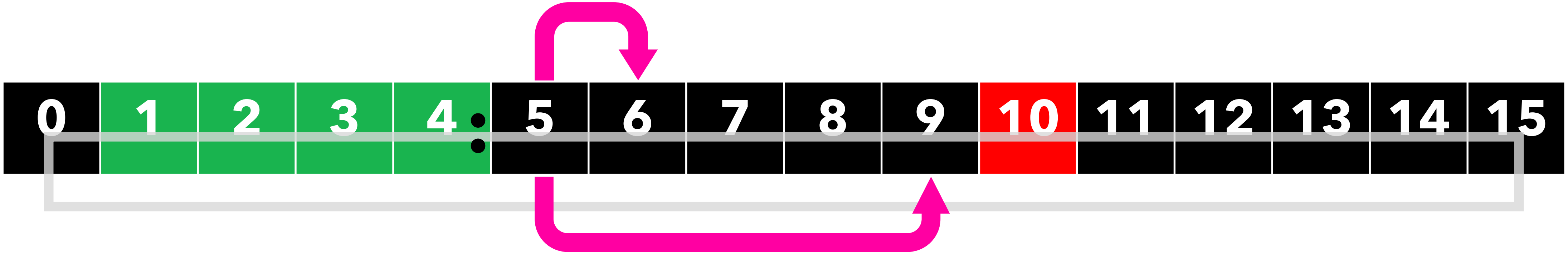


HAMILTONIAN SHORTCUTS STRATEGY

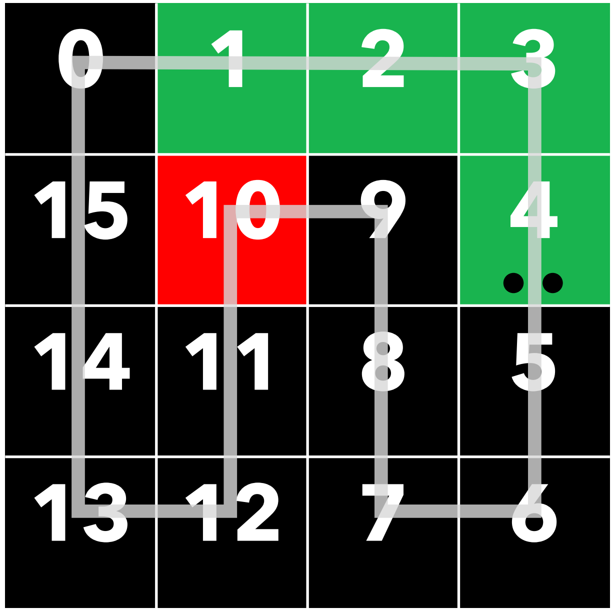


Compute a hamiltonian cycle once and try to skip sections of the cycle to reach the apple faster.

How to take shortcuts?
If all the positions occupied by the snake's body on the hamiltonian cycle are between its tail and its head, the snake will never crash into itself.



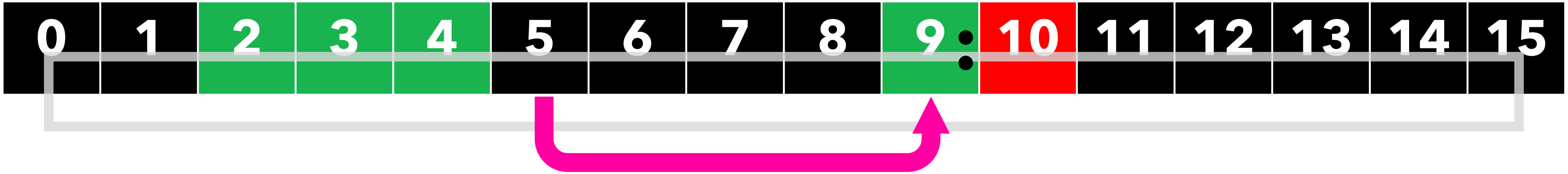
HAMILTONIAN SHORTCUTS STRATEGY



Compute a hamiltonian cycle once and try to skip sections of the cycle to reach the apple faster.

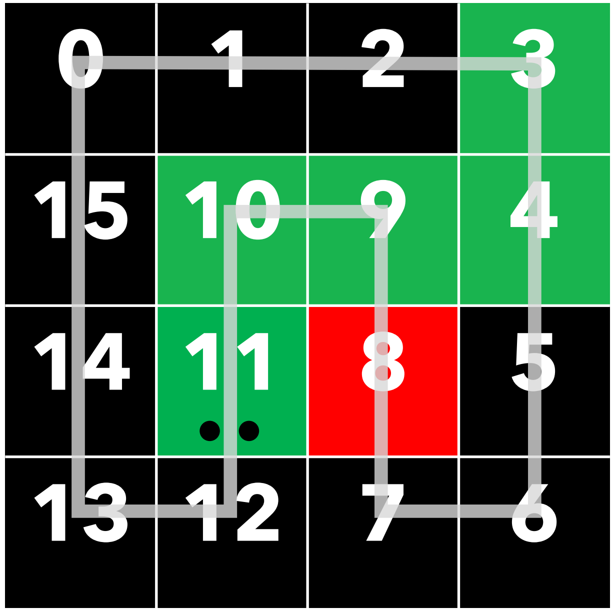
How to take shortcuts?

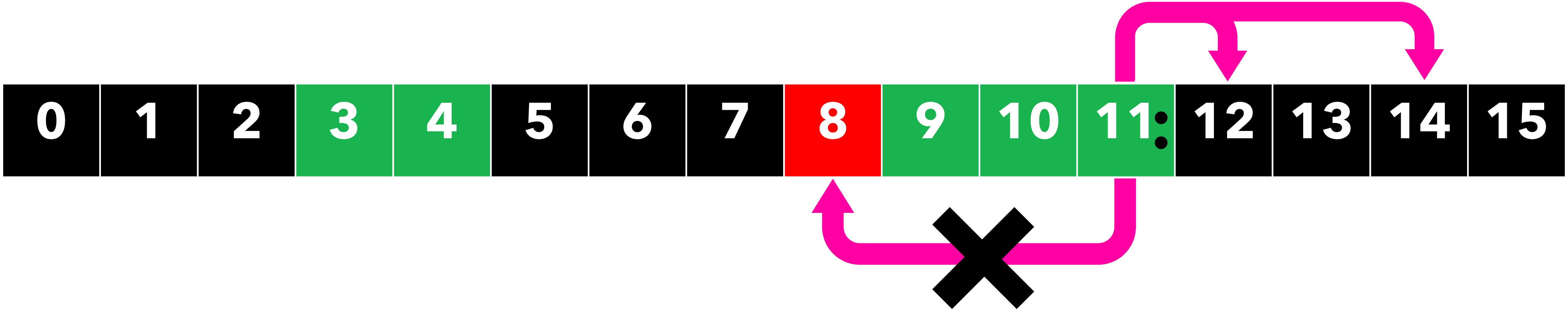
If all the positions occupied by the snake's body on the hamiltonian cycle are between its tail and its head, the snake will never crash into itself.



HAMILTONIAN SHORTCUTS STRATEGY

Another step of the game.

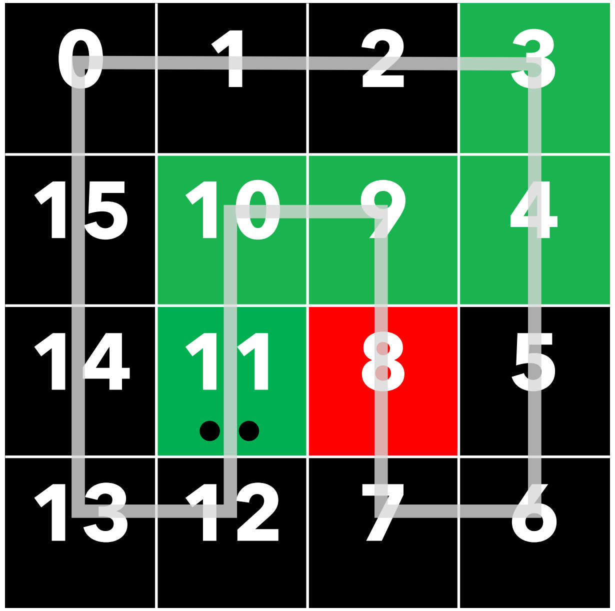


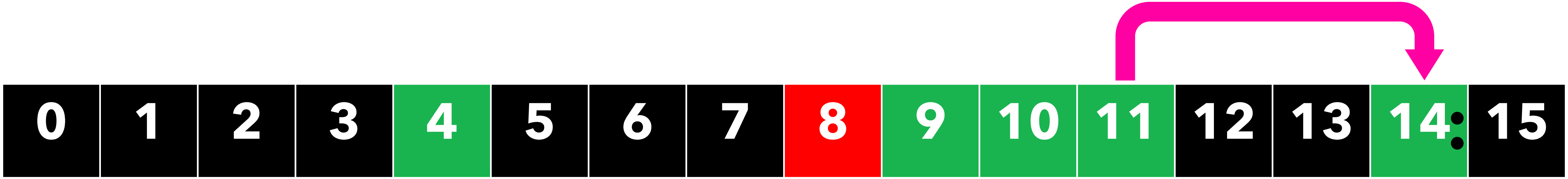


HAMILTONIAN SHORTCUTS

STRATEGY

Another step of the game.



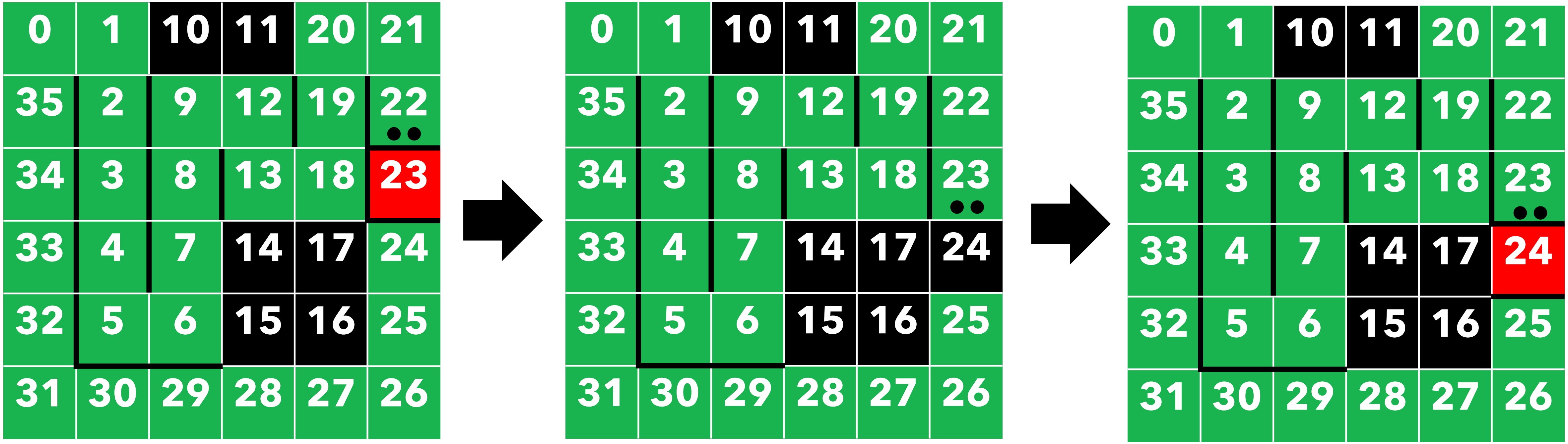


HAMILTONIAN SHORTCUTS STRATEGY

+ taking shortcuts reduces the number of steps to reach the apple

+ deciding the next move requires constant time (few modular arithmetic operations)

- taking shortcuts in the late stages of the game can occasionally lead the snake to crash into itself ⇨ the snake stops taking shortcuts when it gets long



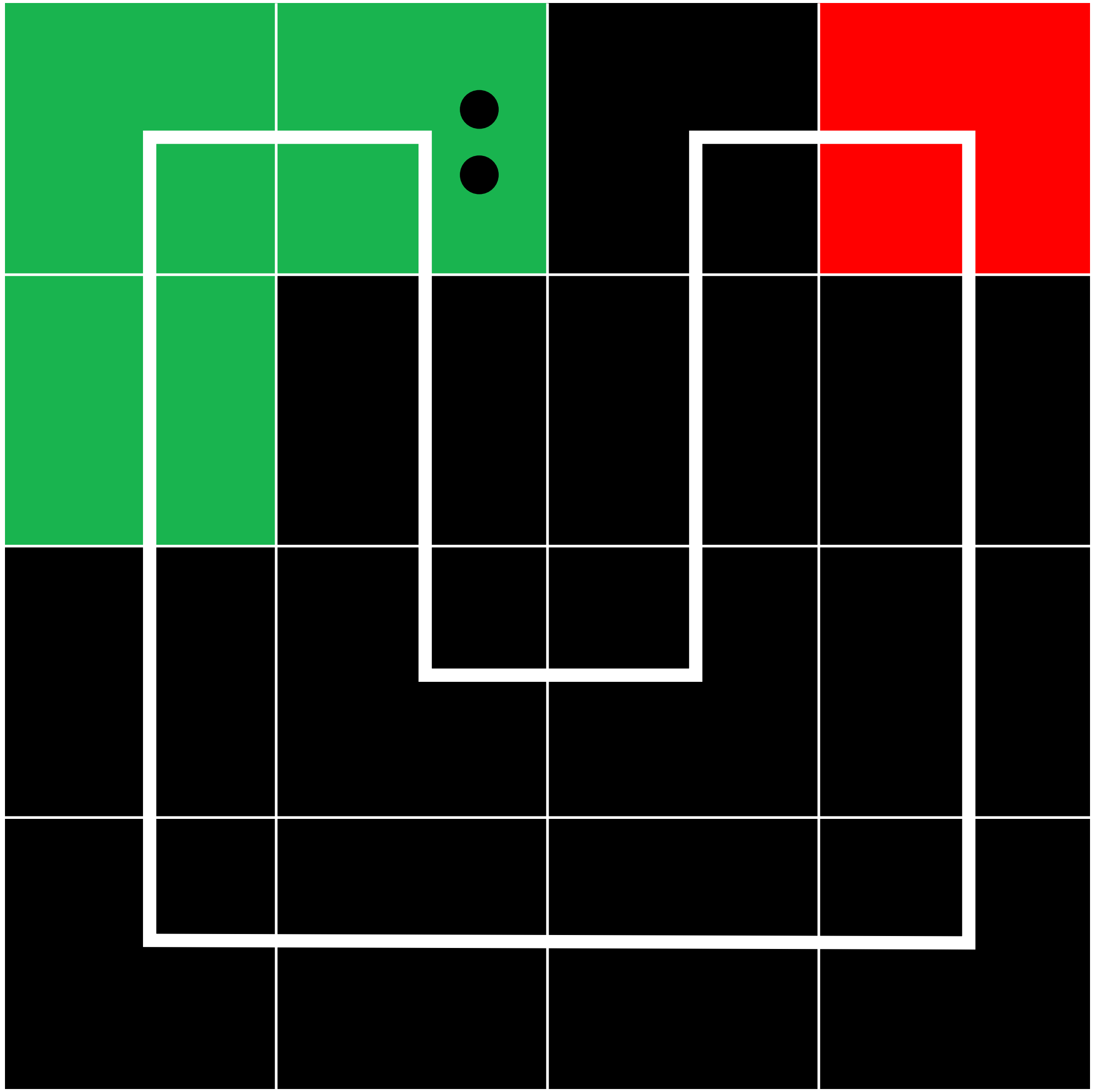
0	1	10	11	20	21
35	2	9	12	19	22
34	3	8	13	18	23
33	4	7	14	17	24
32	5	6	15	16	25
31	30	29	28	27	26

0	1	10	11	20	21
35	2	9	12	19	22
34	3	8	13	18	23
33	4	7	14	17	24
32	5	6	15	16	25
31	30	29	28	27	26

GAME OVER

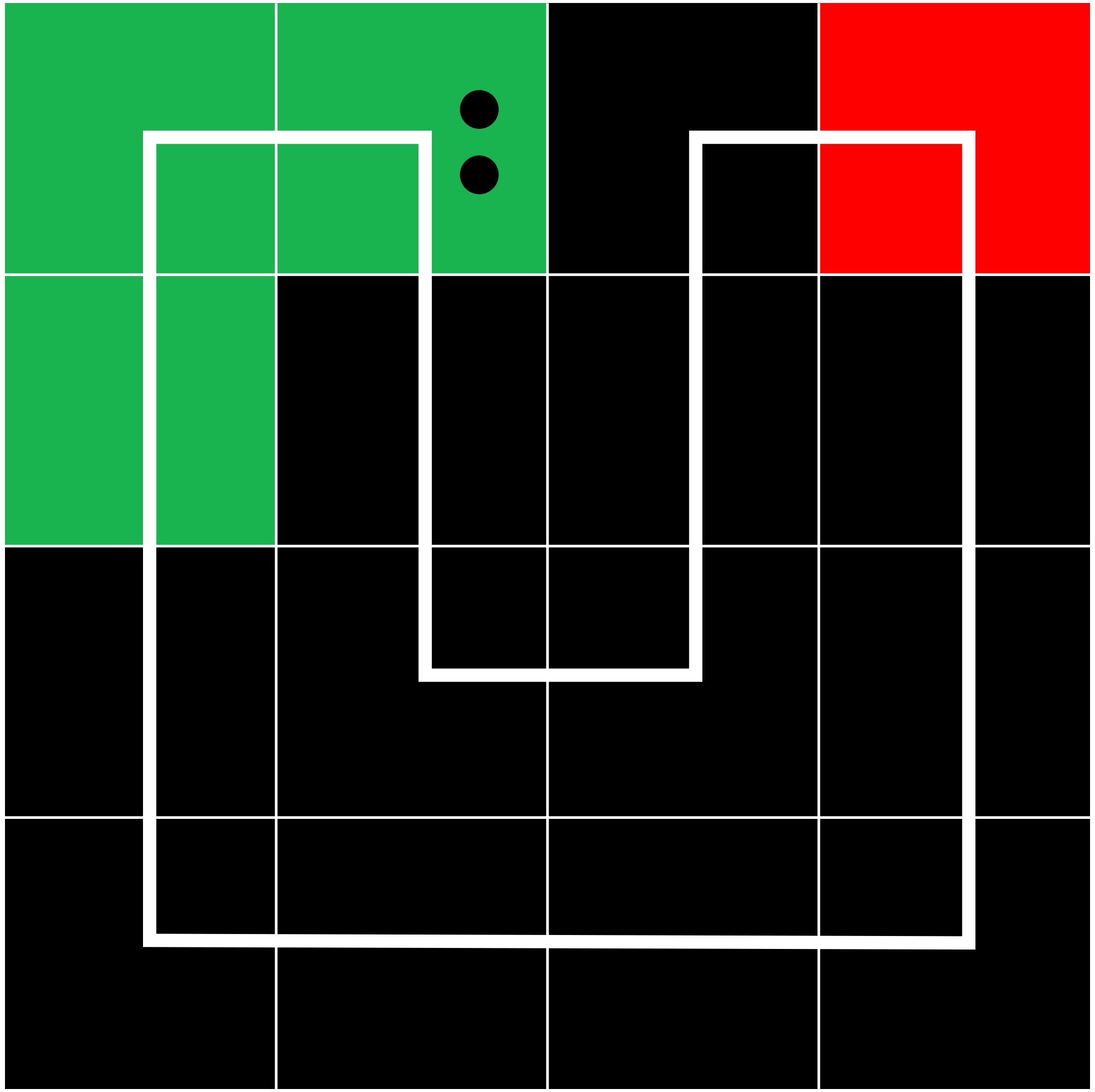
0	1	10	11	20	21
35	2	9	12	19	22
34	3	8	13	18	23
33	4	7	14	17	24
32	5	6	15	16	25
31	30	29	28	27	26

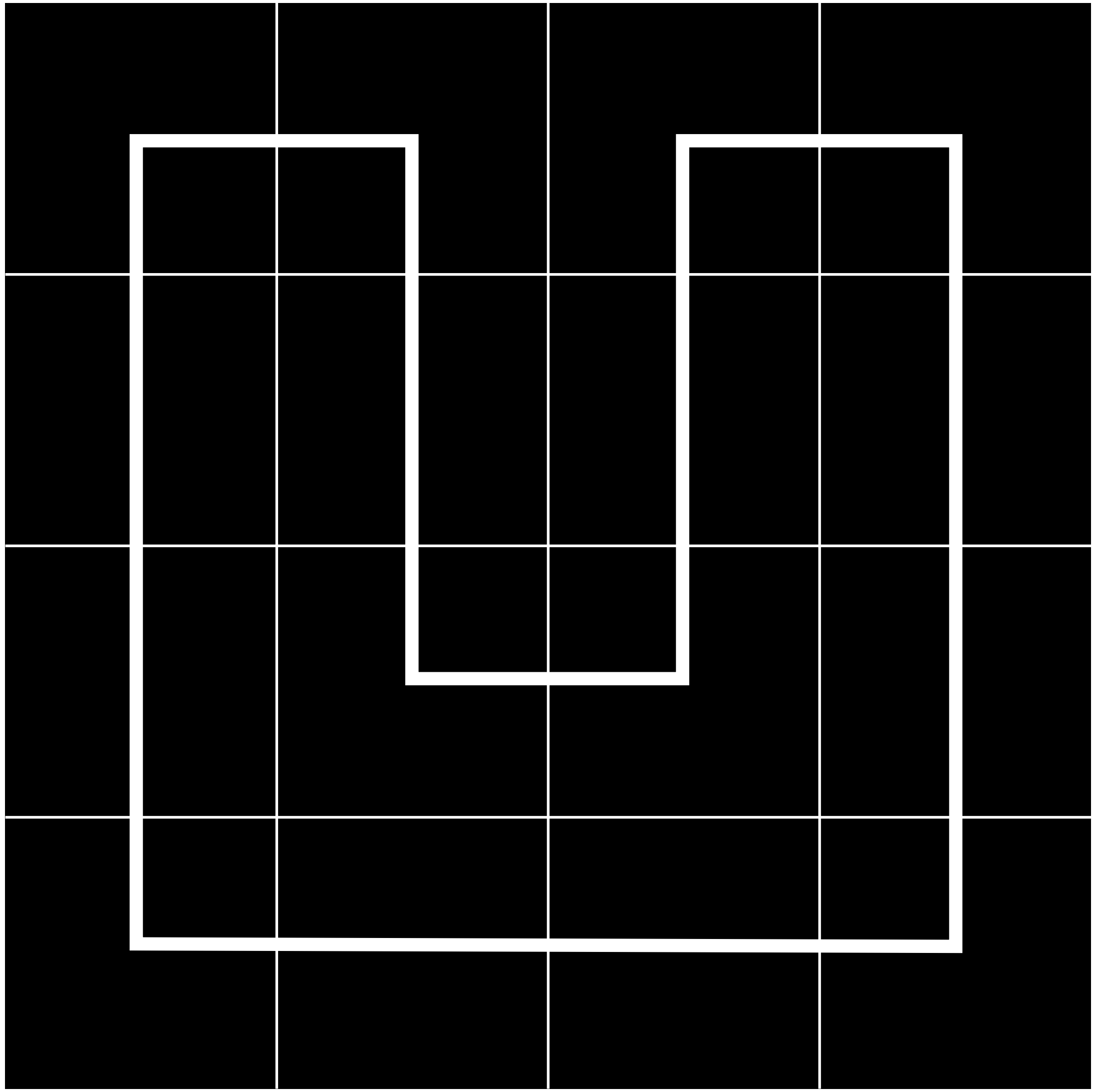
HAMILTONIAN CYCLE CHANGE STRATEGY



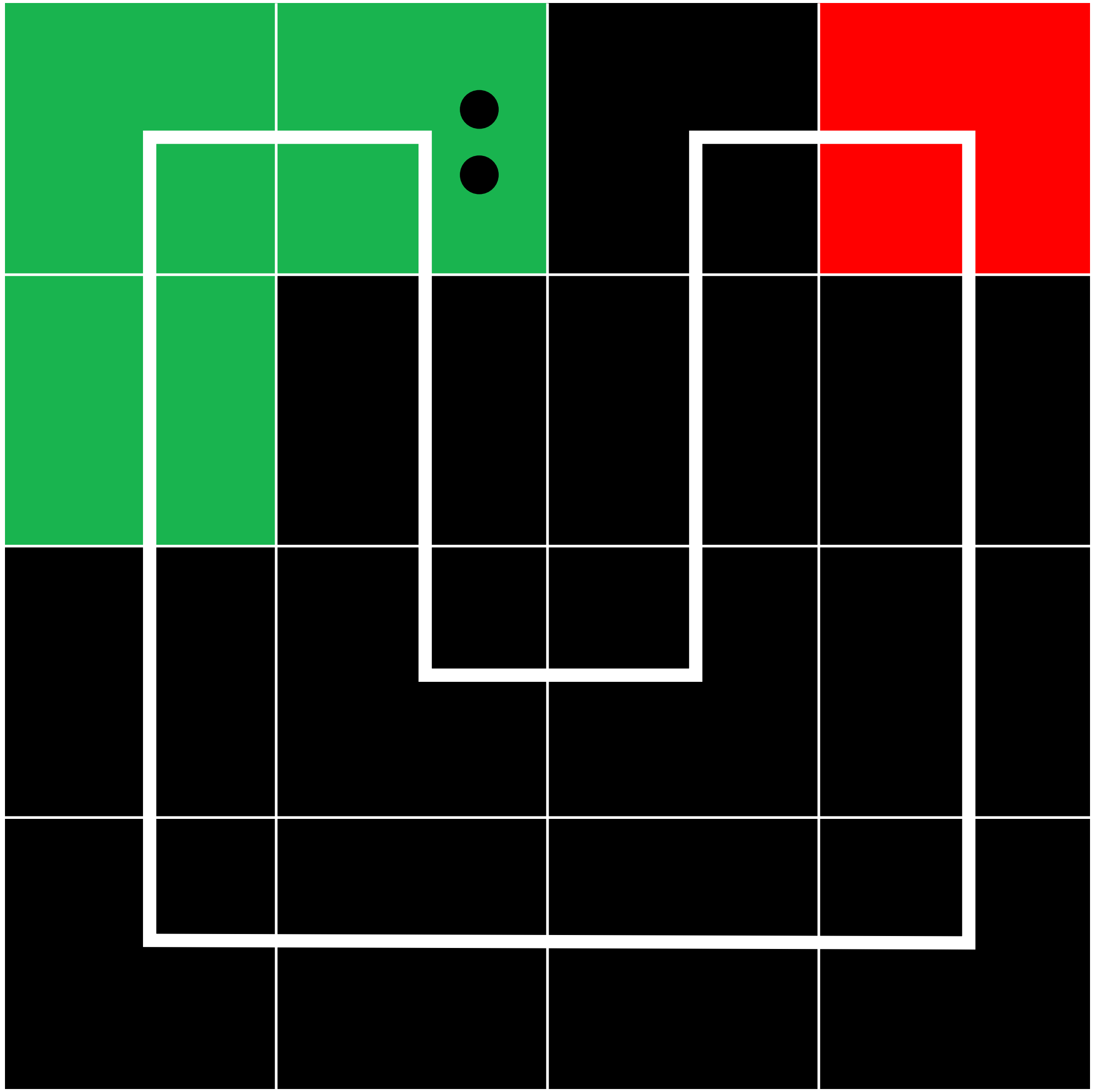
Compute a hamiltonian cycle at every iteration such that the distance between the snake's head and the food is less than the one in the previous cycle.

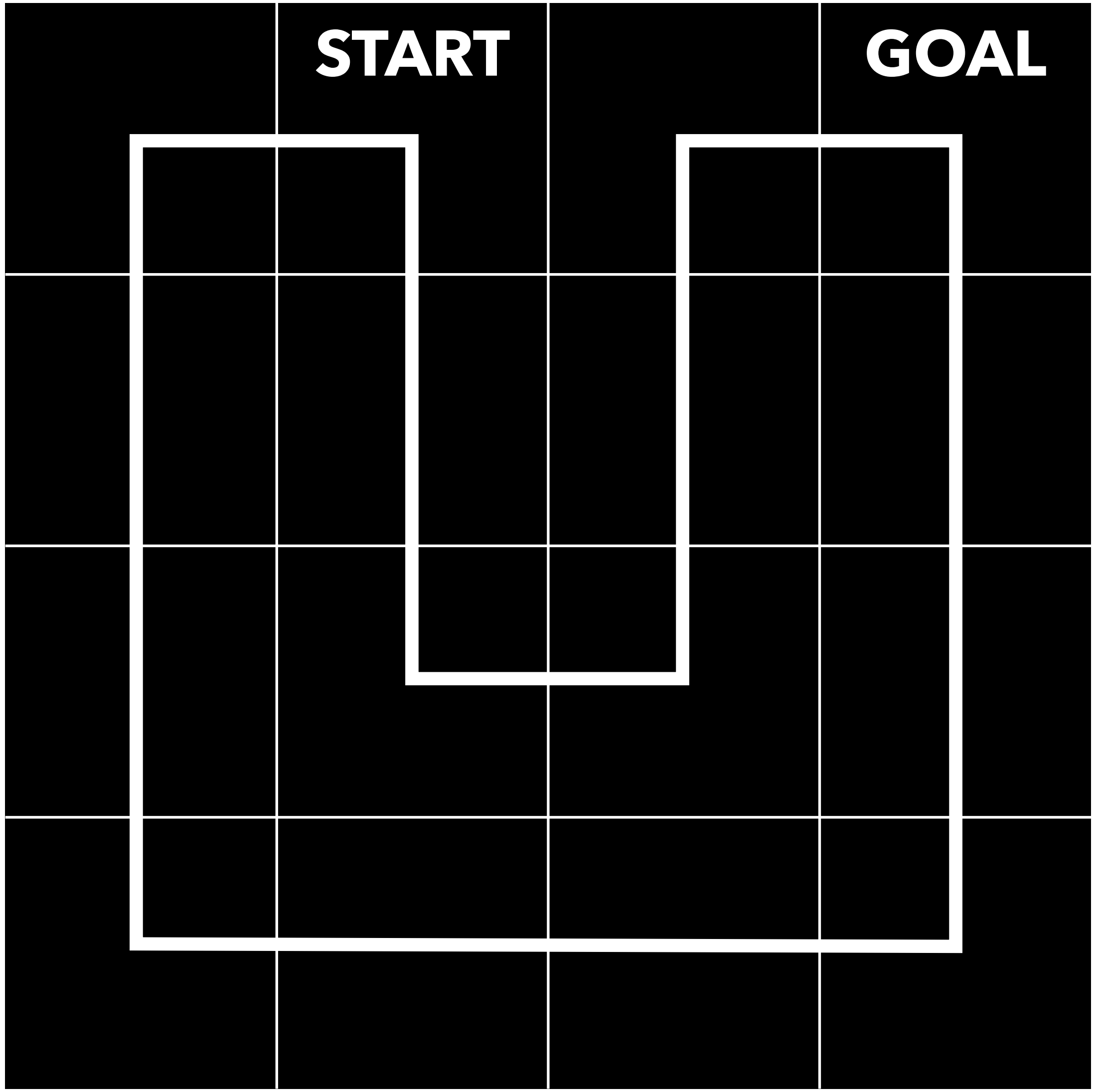
HAMILTONIAN CYCLE CHANGE STRATEGY



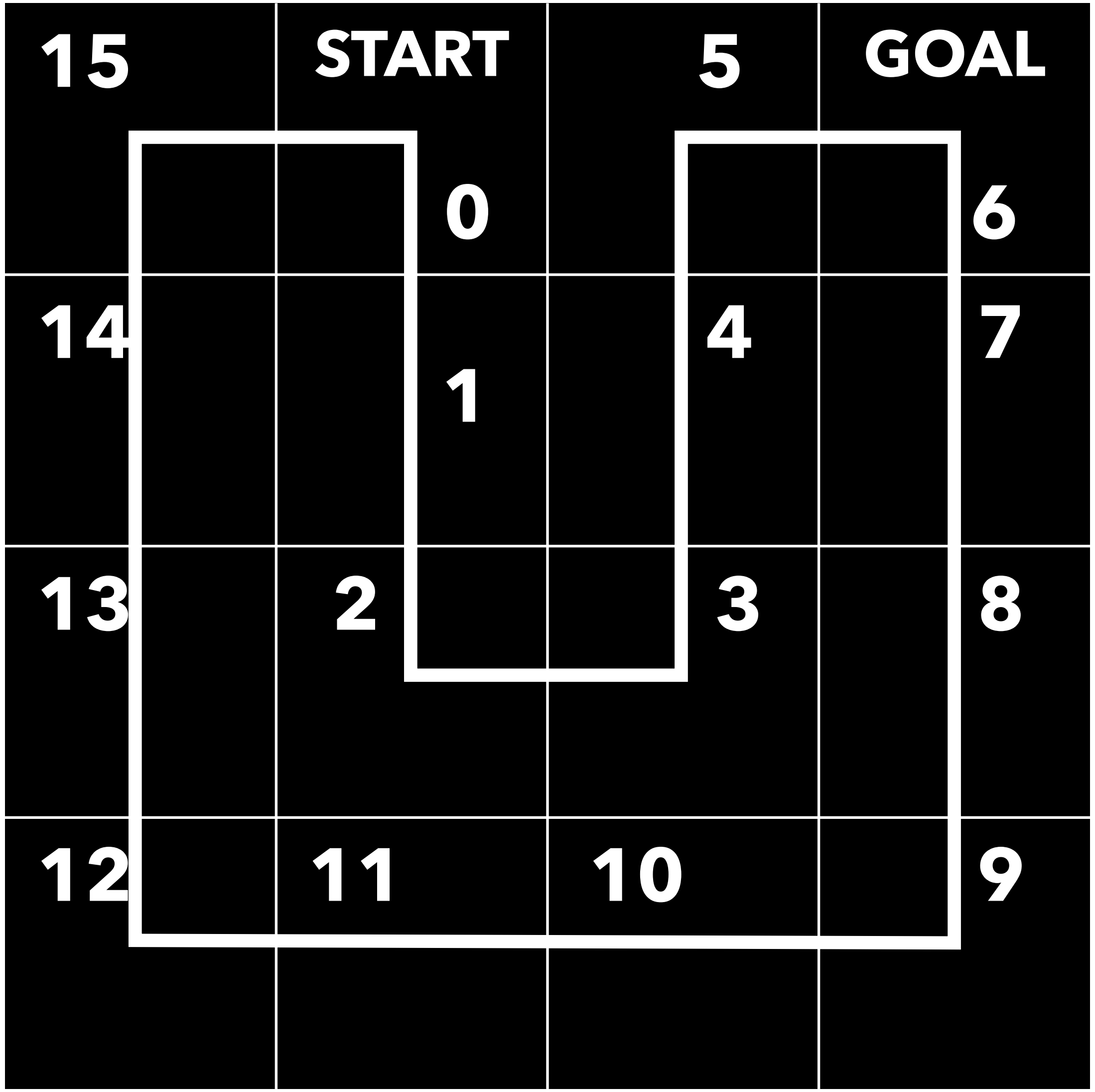


HAMILTONIAN CYCLE CHANGE STRATEGY



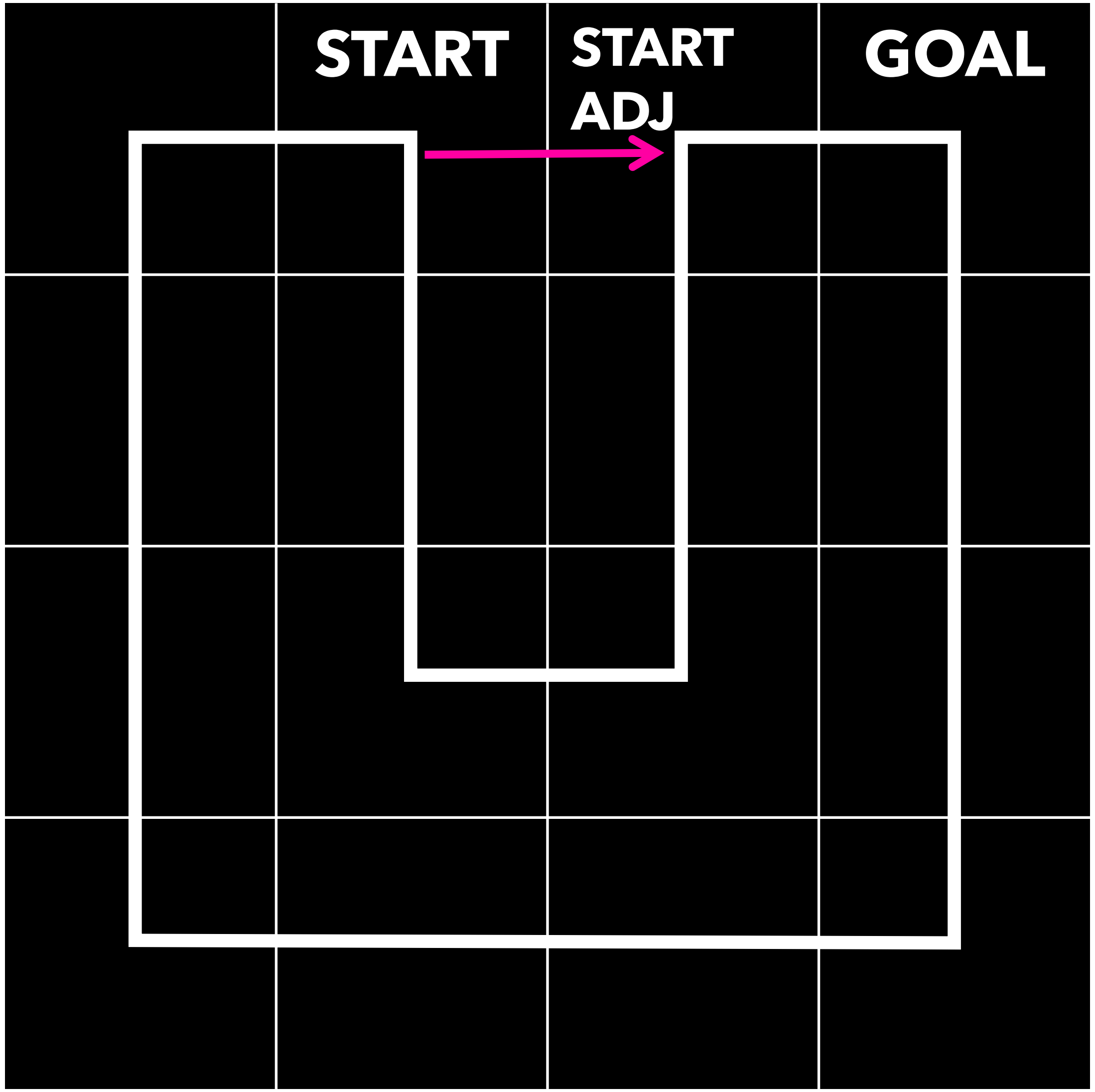


HAMILTONIAN CYCLE CHANGE STRATEGY



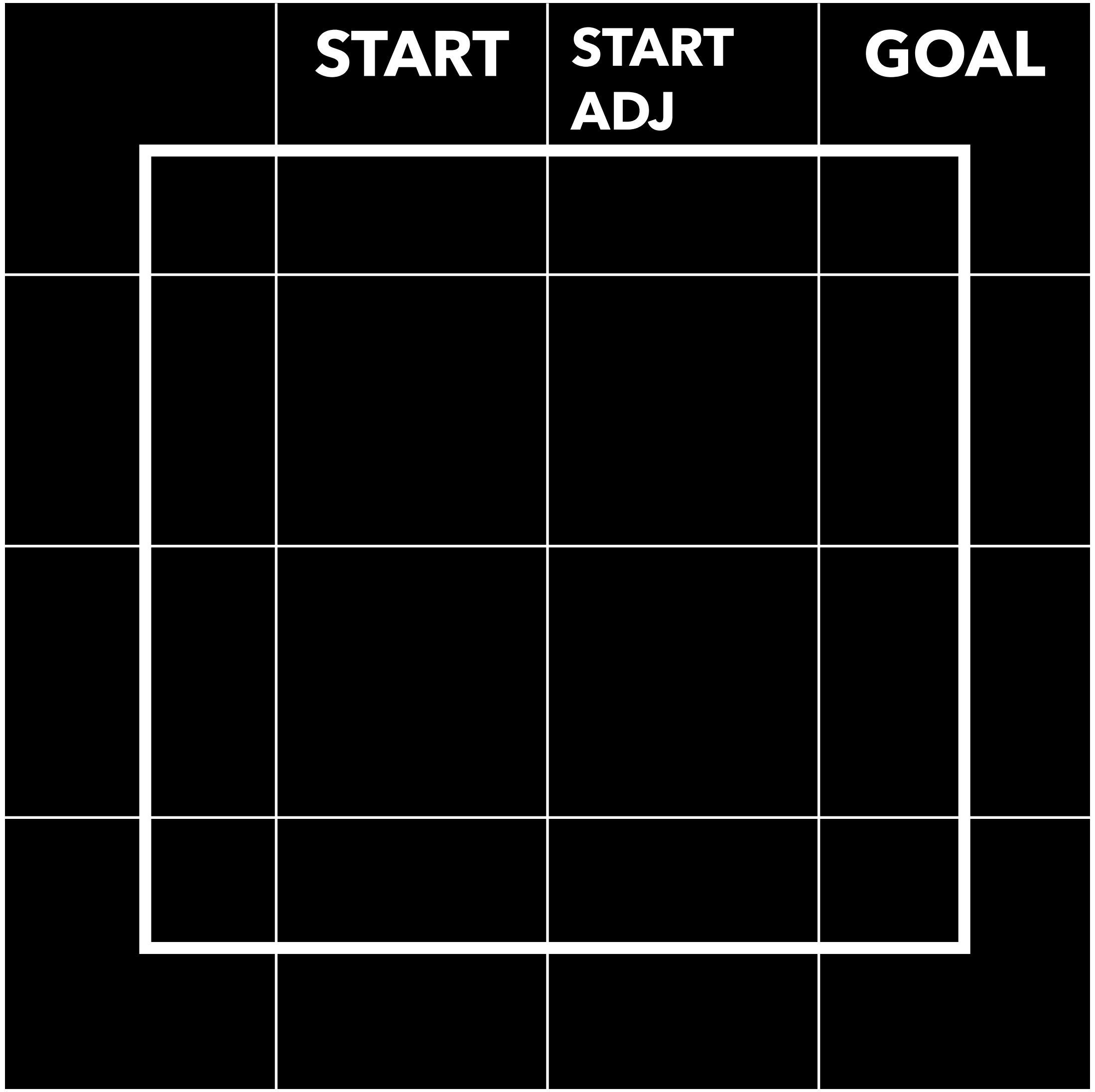
Assign a value
to each node following the
cycle.

HAMILTONIAN CYCLE CHANGE STRATEGY



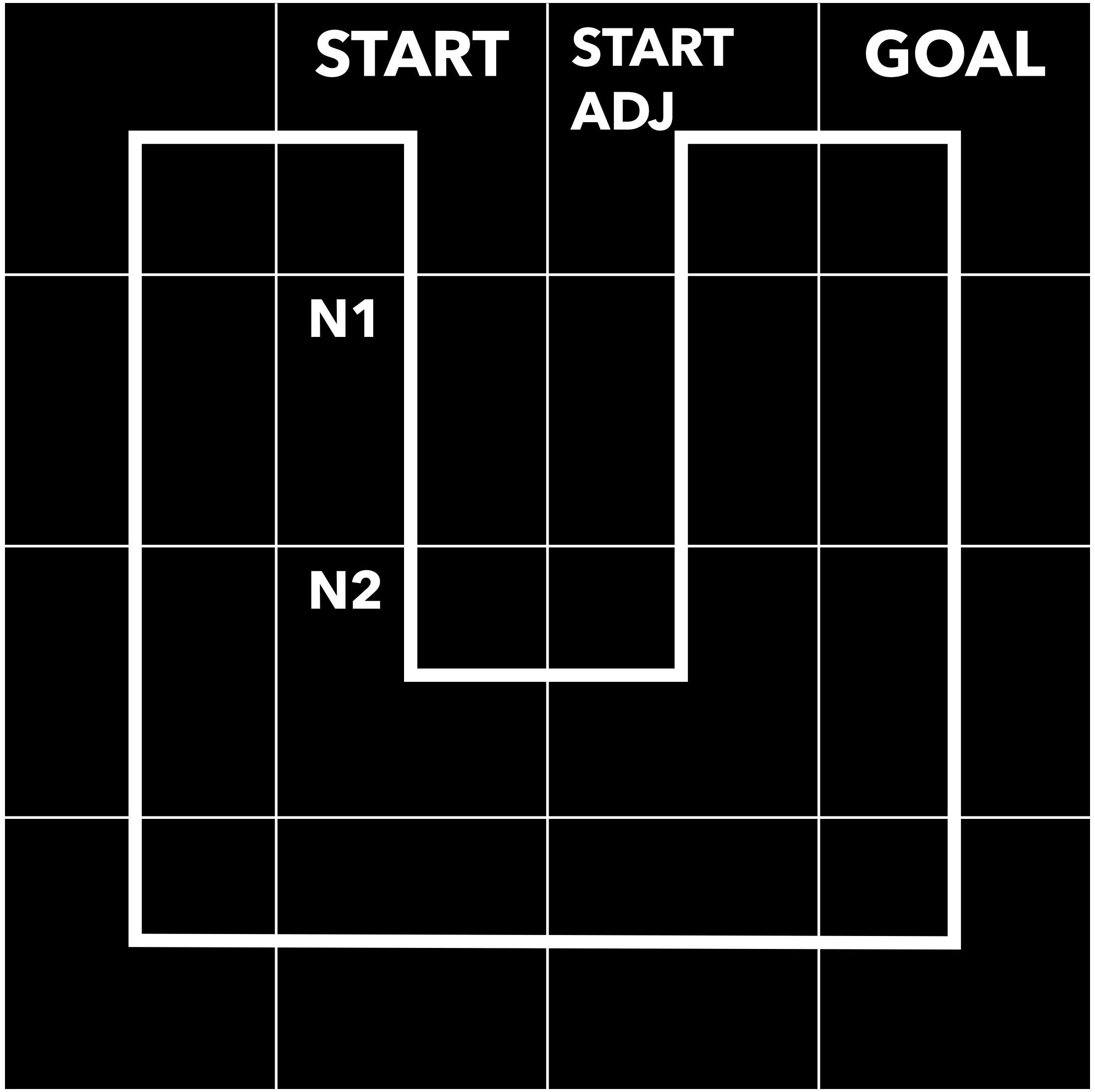
START_ADJ.VALUE in
(START.VALUE + 2, GOAL.VALUE)

HAMILTONIAN CYCLE CHANGE STRATEGY



How to insert the excluded nodes in the new cycle?

HAMILTONIAN CYCLE CHANGE STRATEGY

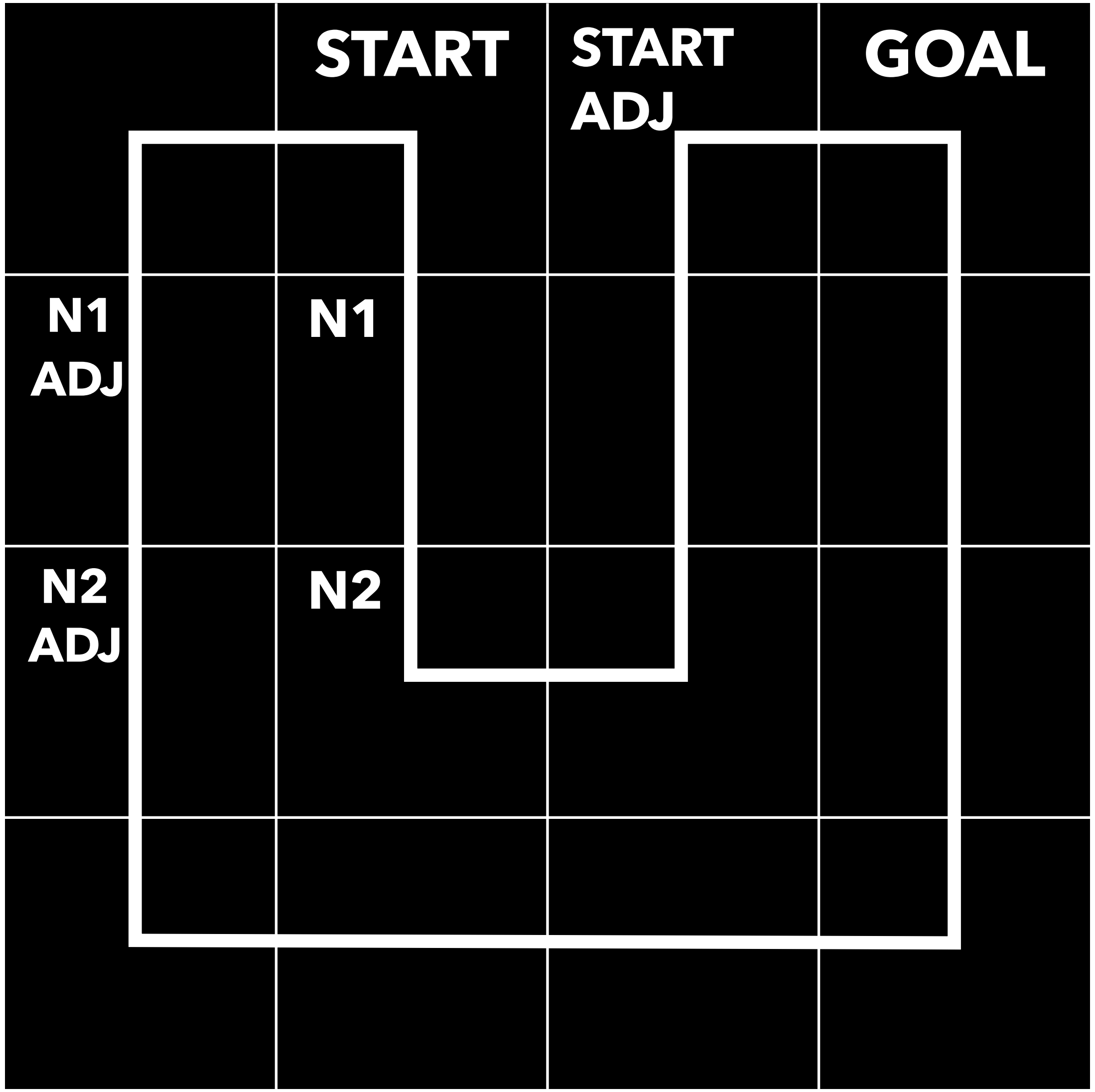


N1.VALUE, N2.VALUE in
(START.VALUE, START_ADJ.VALUE)

and

N2.VALUE == N1.VALUE +1

HAMILTONIAN CYCLE CHANGE STRATEGY



N1_ADJ.VALUE, N2_ADJ.VALUE in (GOAL.VALUE, START.VALUE)

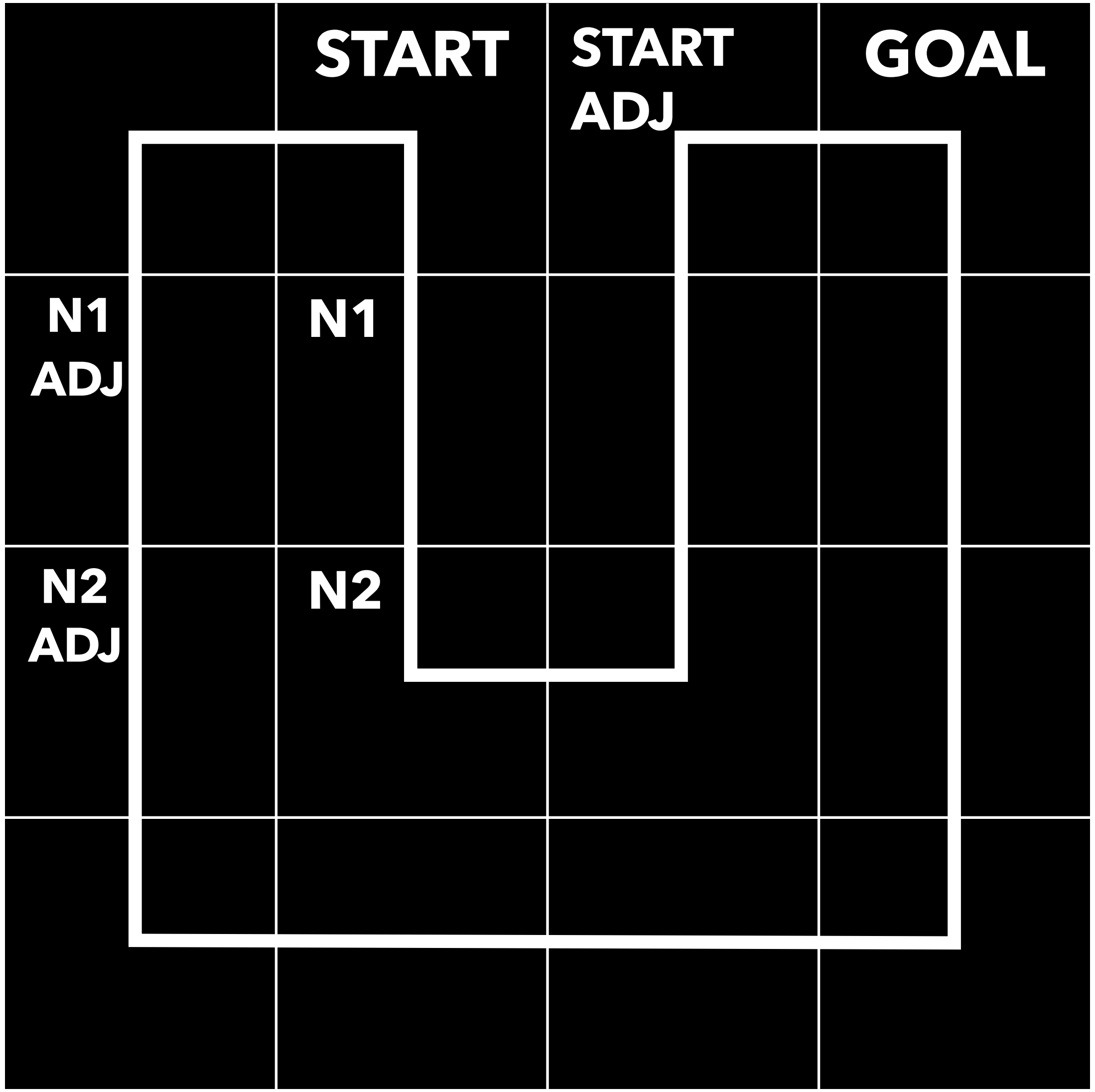
and

N1_ADJ adjacent to N1 in the grid graph
N2_ADJ adjacent to N2

and

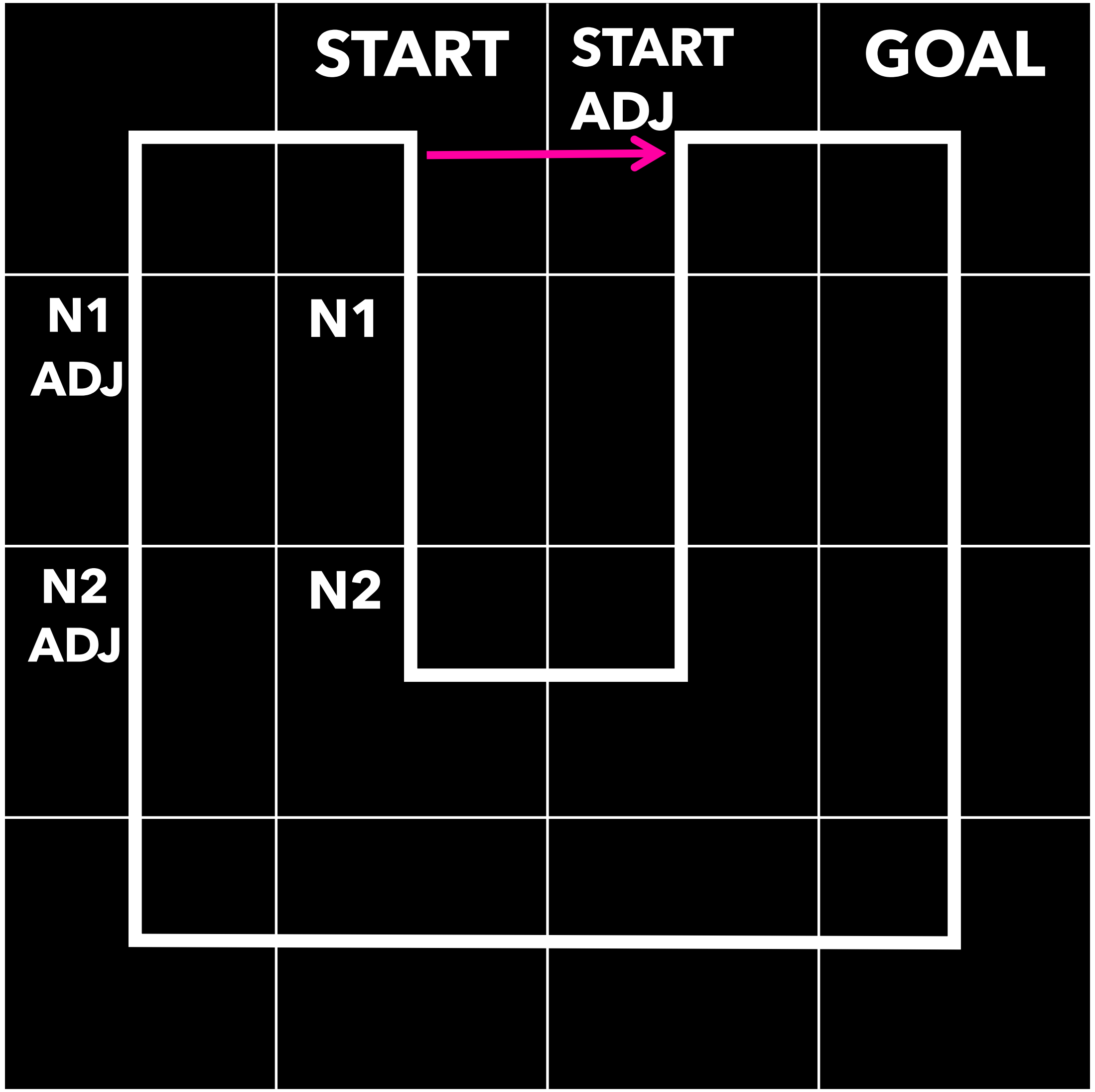
N2_ADJ.VALUE == N1_ADJ.VALUE - 1

HAMILTONIAN CYCLE CHANGE STRATEGY



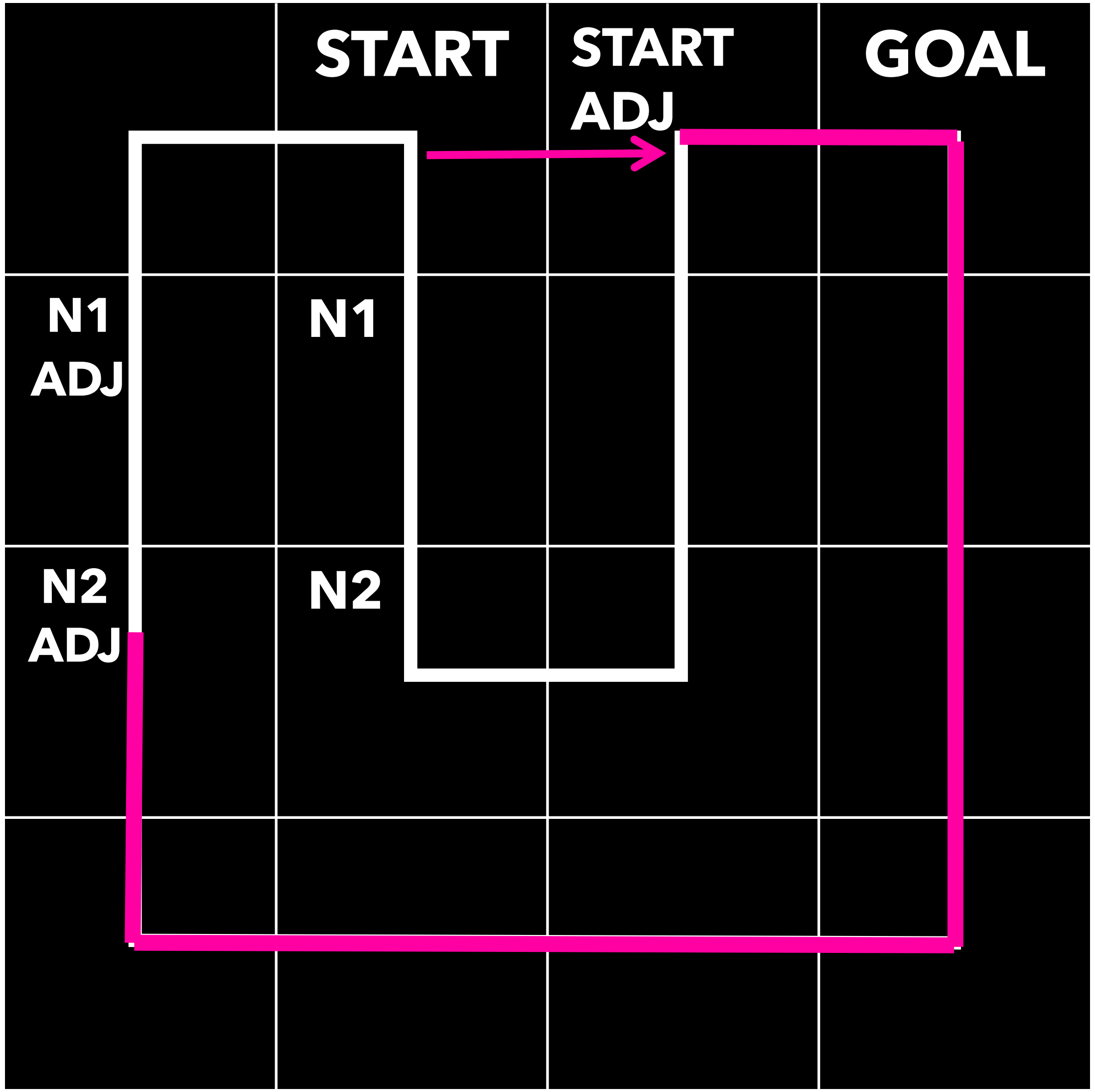
Compute the new cycle.

HAMILTONIAN CYCLE CHANGE STRATEGY



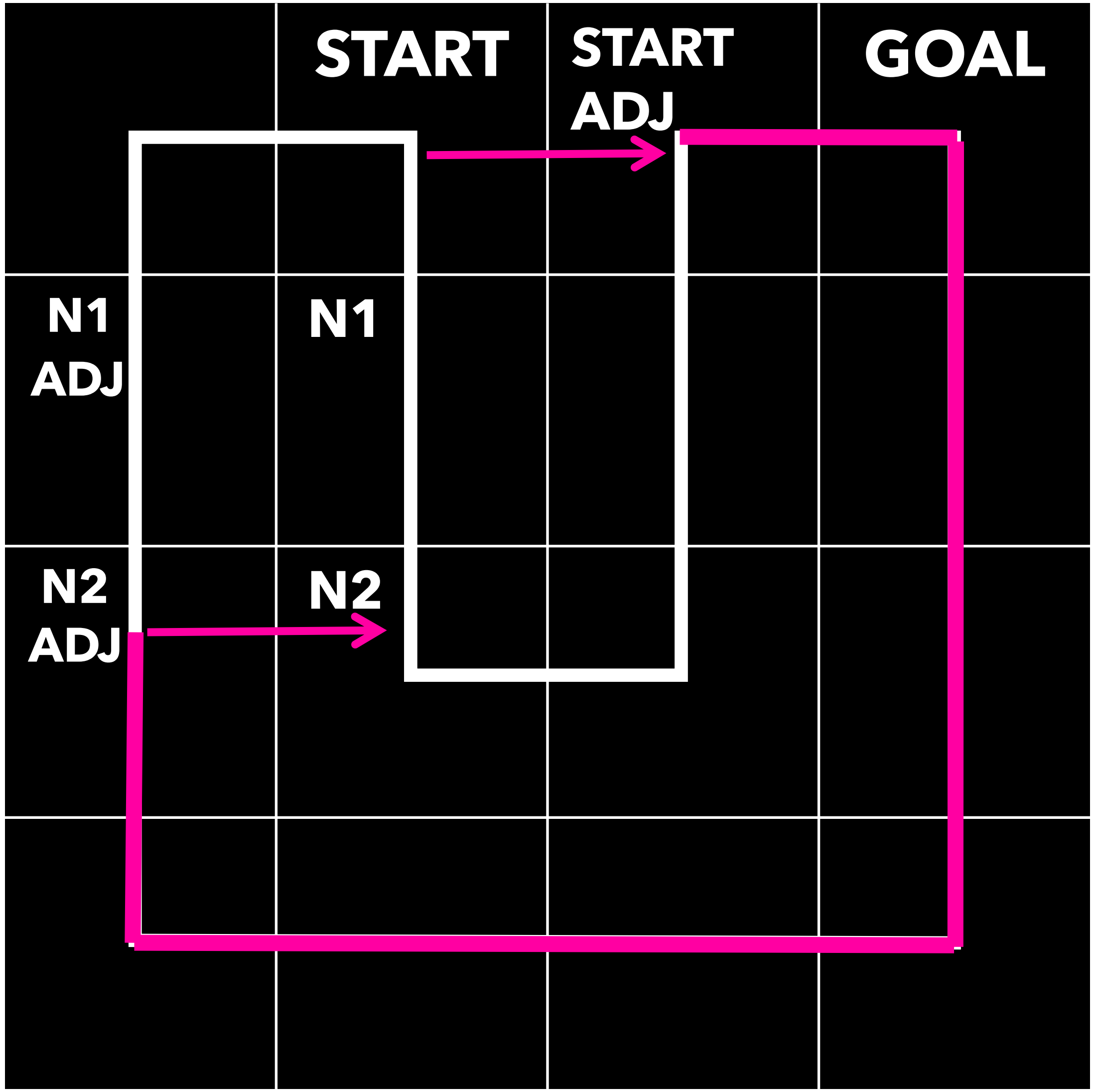
Connect
START with START_ADJ.

HAMILTONIAN CYCLE CHANGE STRATEGY



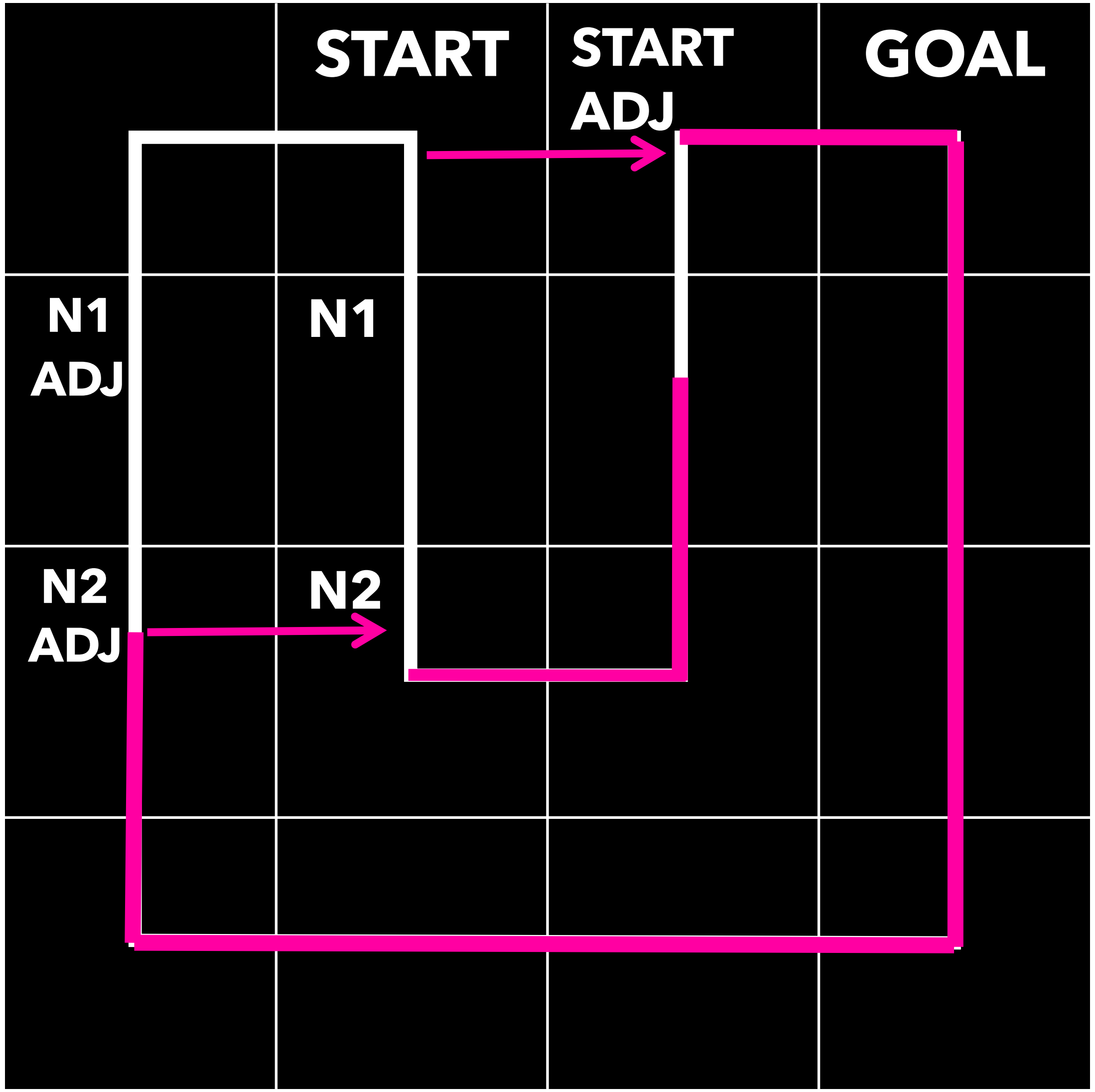
Follow the cycle from START_ADJ to N2_ADJ.

HAMILTONIAN CYCLE CHANGE STRATEGY



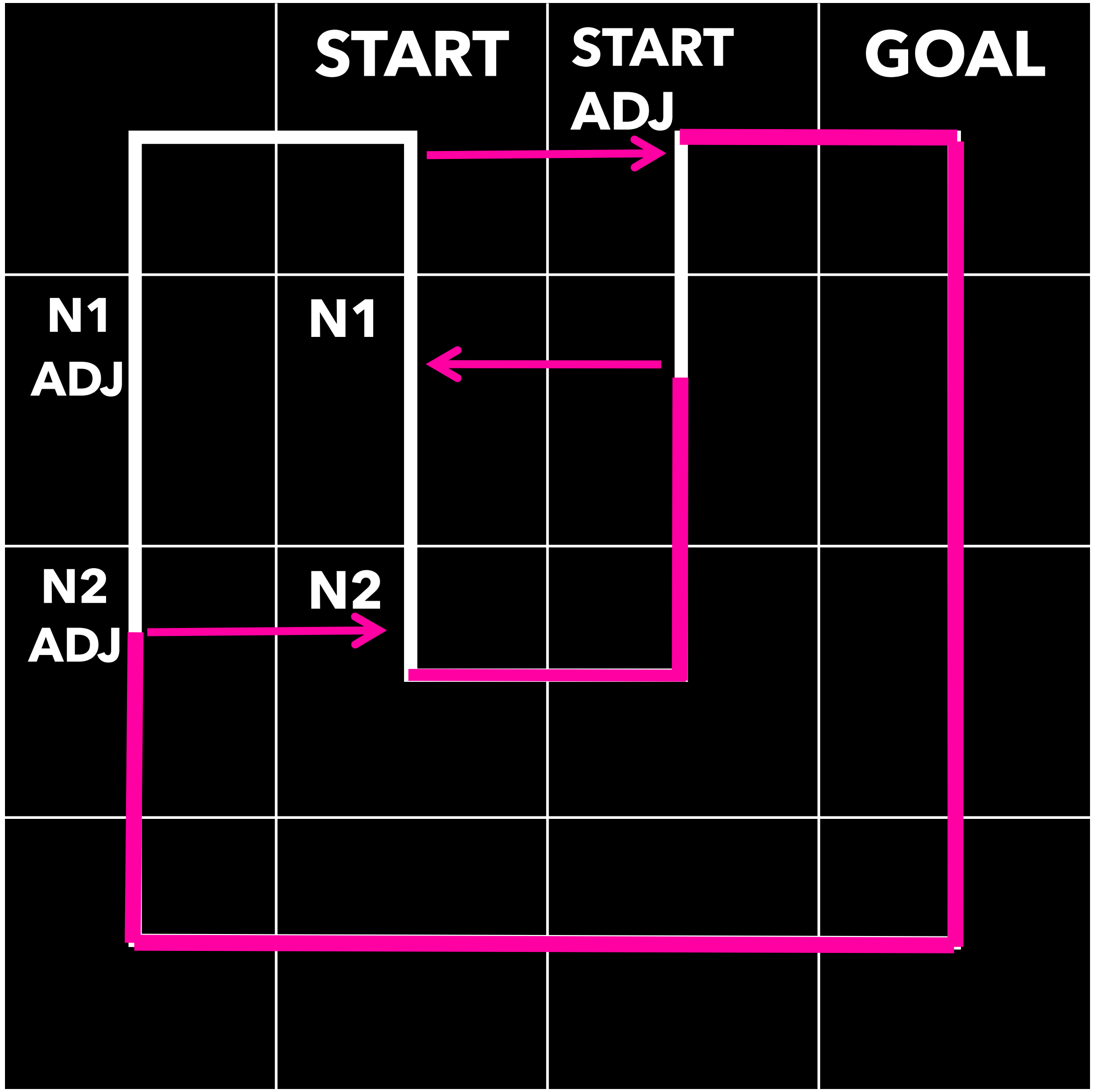
Connect N2_ADJ to N2.

HAMILTONIAN CYCLE CHANGE STRATEGY



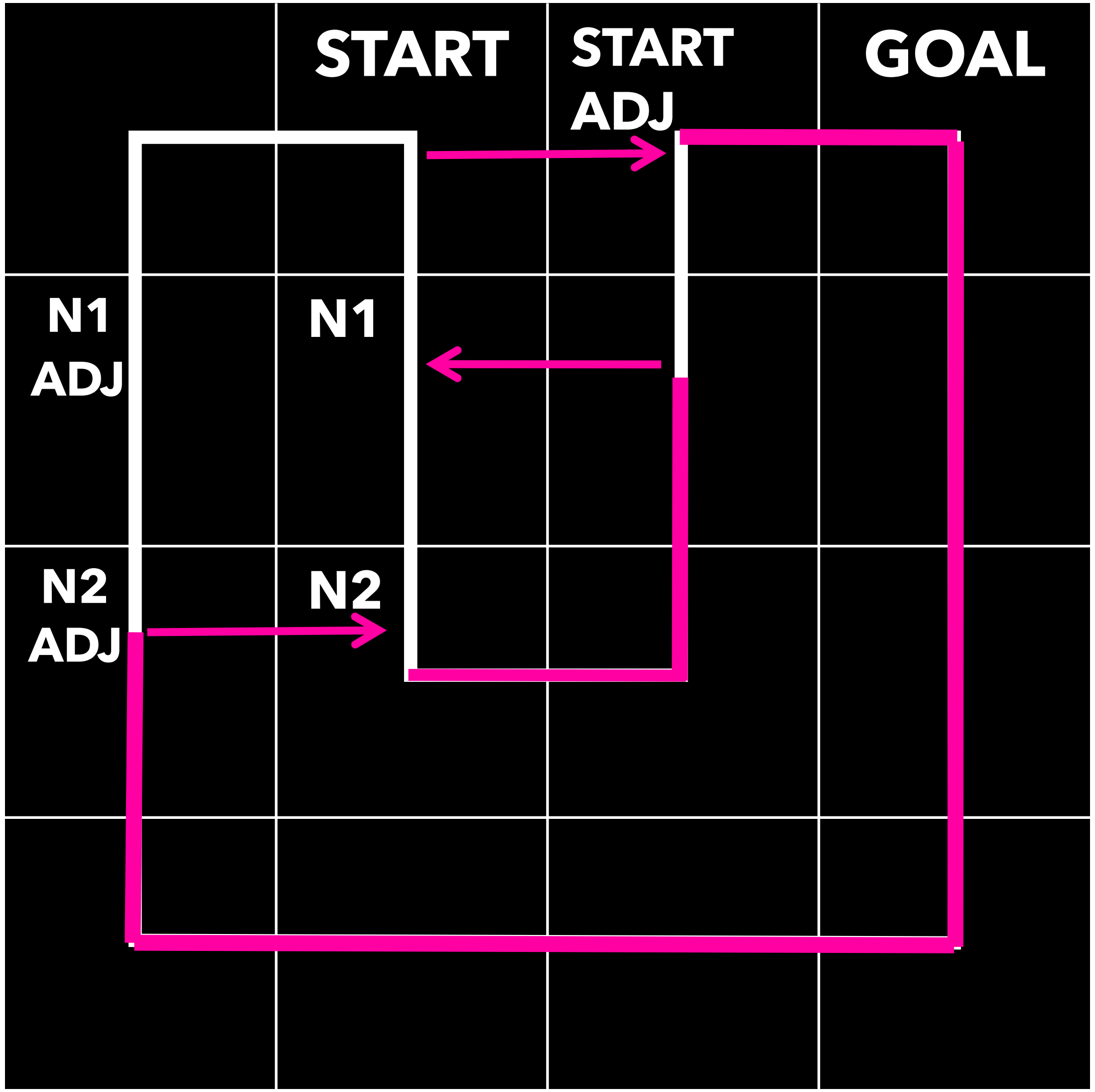
Follow the cycle from N2 to the node before START ADJ.

HAMILTONIAN CYCLE CHANGE STRATEGY



Connect to the node following START.

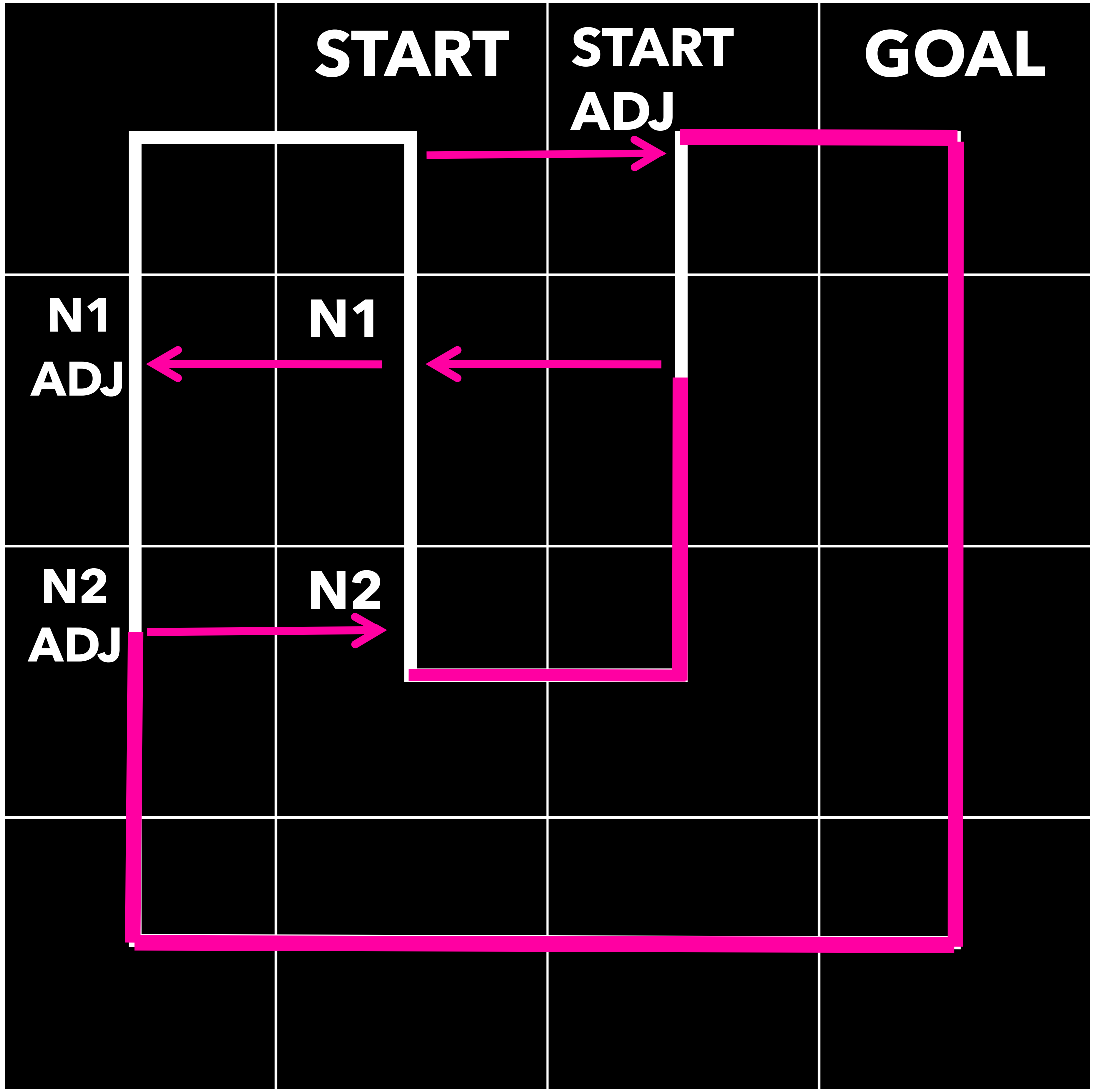
HAMILTONIAN CYCLE CHANGE STRATEGY



Connect to the node following START.

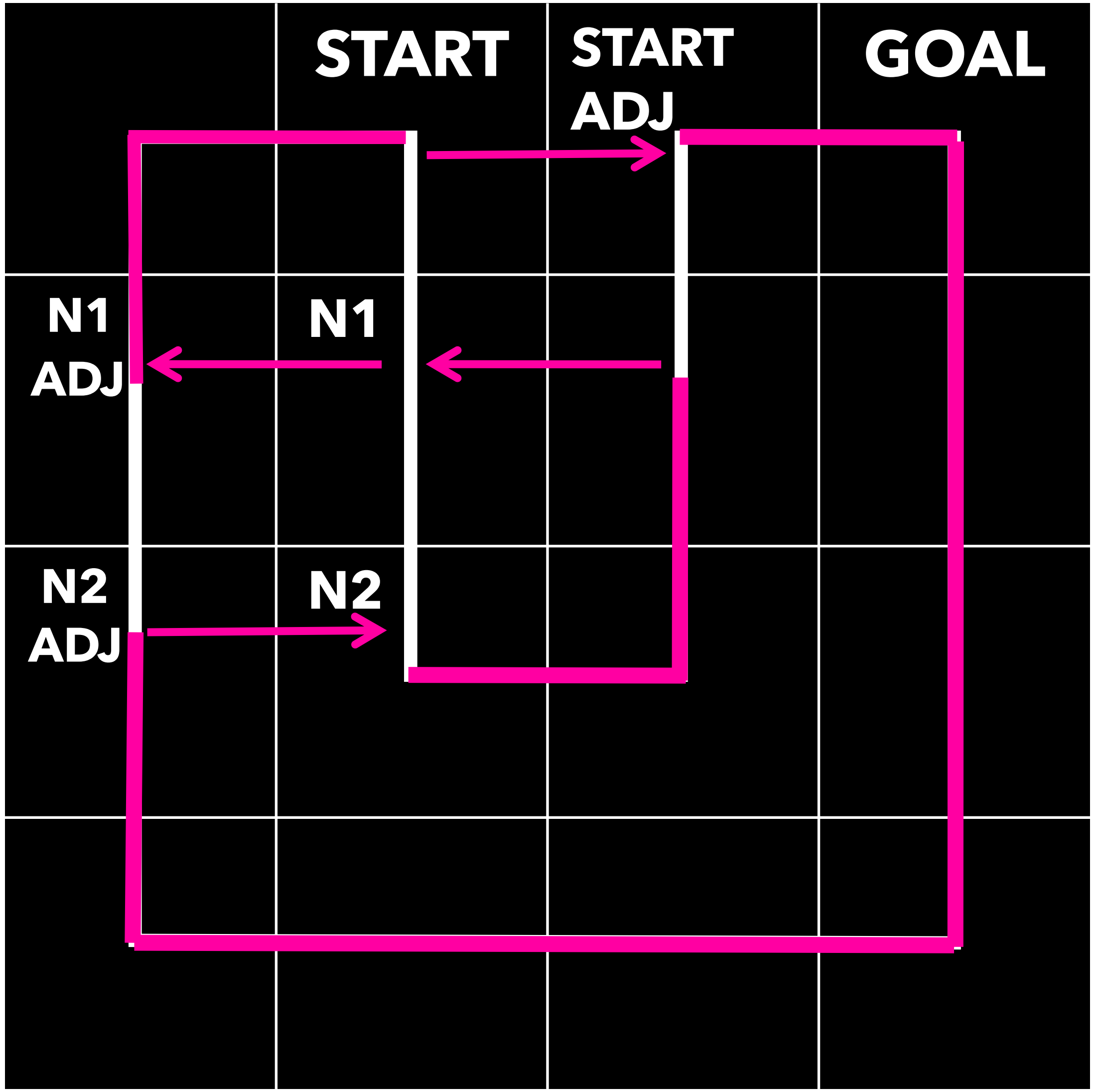
Follow the cycle in the reverse way until N1.

HAMILTONIAN CYCLE CHANGE STRATEGY



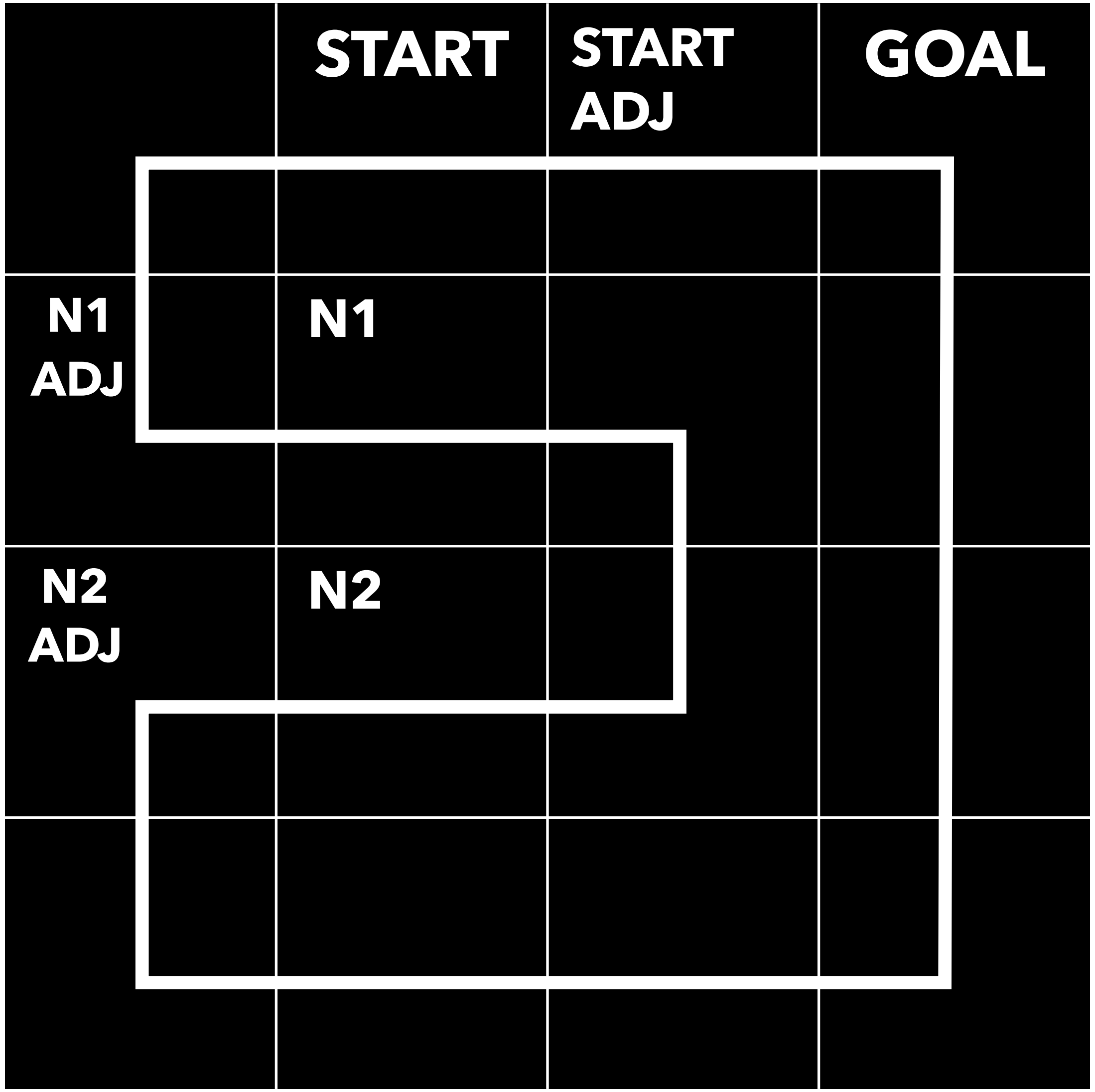
Connect N1 to N1_ADJ.

HAMILTONIAN CYCLE CHANGE STRATEGY



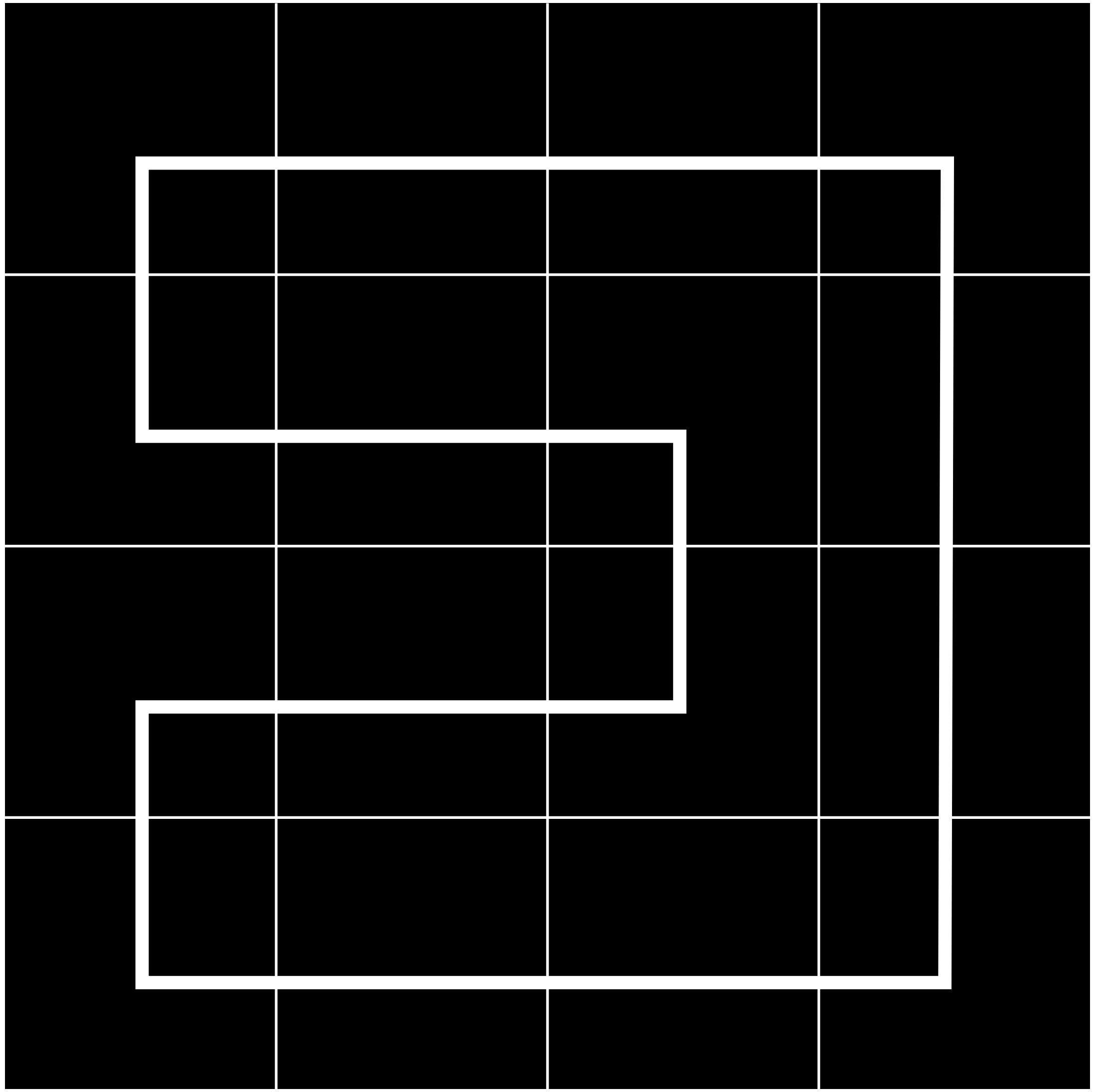
Follow the cycle from N1_ADJ to START.

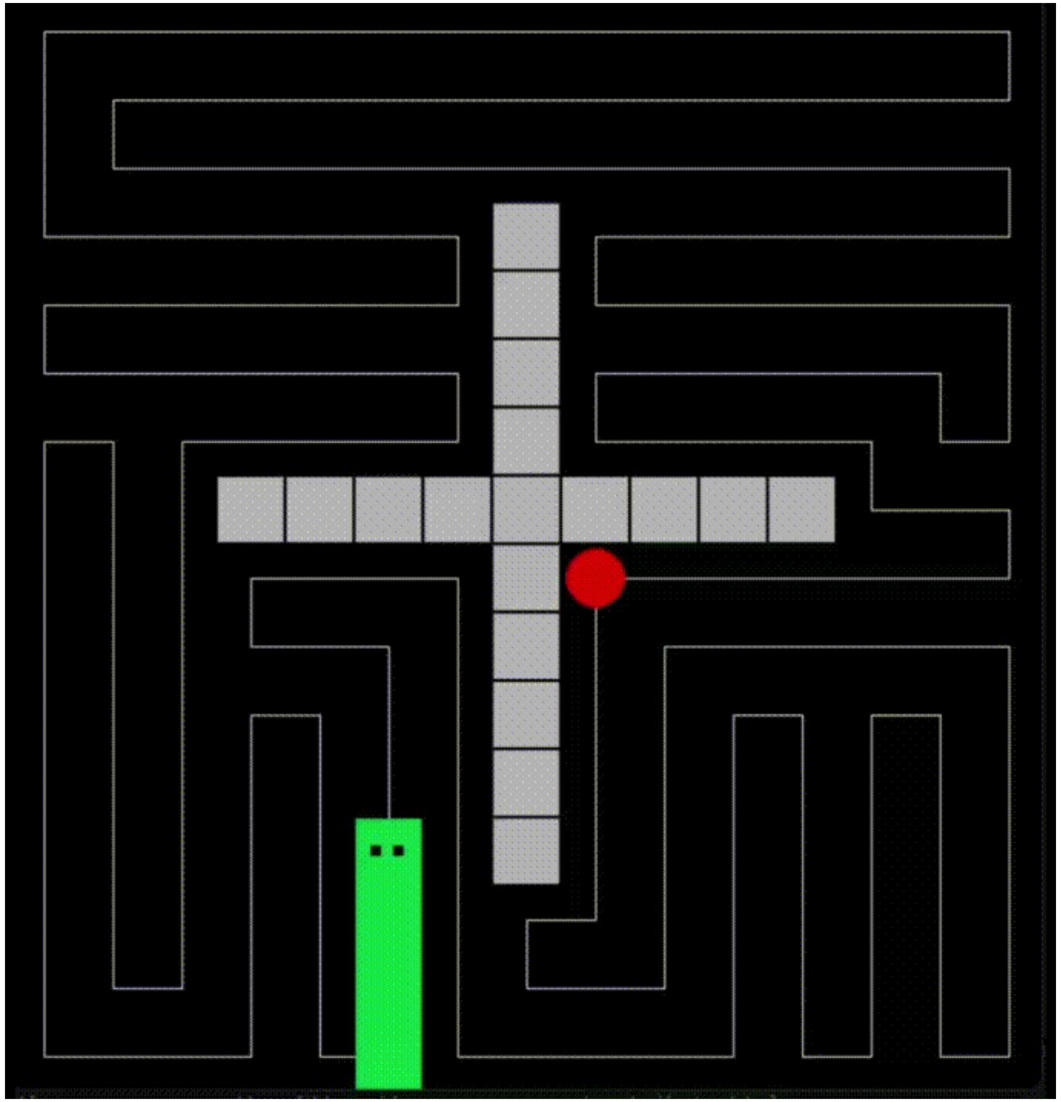
HAMILTONIAN CYCLE CHANGE STRATEGY



Connect all the nodes.

HAMILTONIAN CYCLE CHANGE STRATEGY



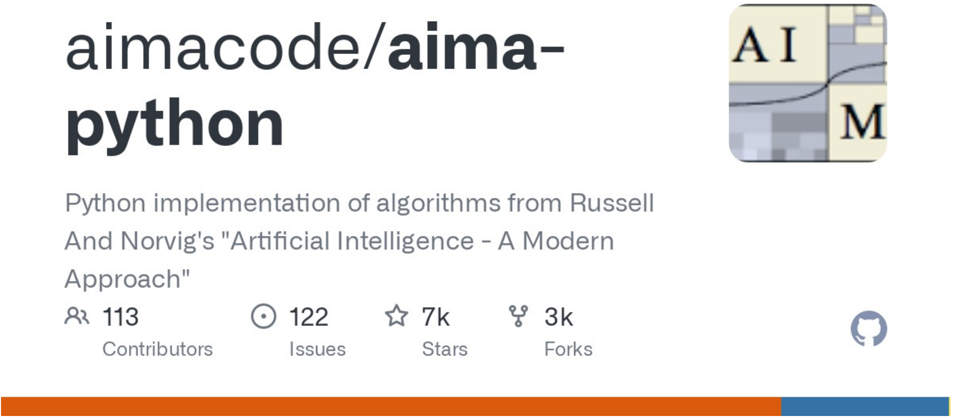


REFERENCES

❑ Snake (history of the game)
❑ Hamiltonian shortcuts strategy
❑ Longest path heuristic
❑ Hamiltonian cycle change strategy

SOFTWARE

❑ pygame

❑ 

aimacode/**aima-python**

Python implementation of algorithms from Russell And Norvig's "Artificial Intelligence - A Modern Approach"

113 Contributors 122 Issues 7k Stars 3k Forks

THANK YOU FOR YOUR ATTENTION!