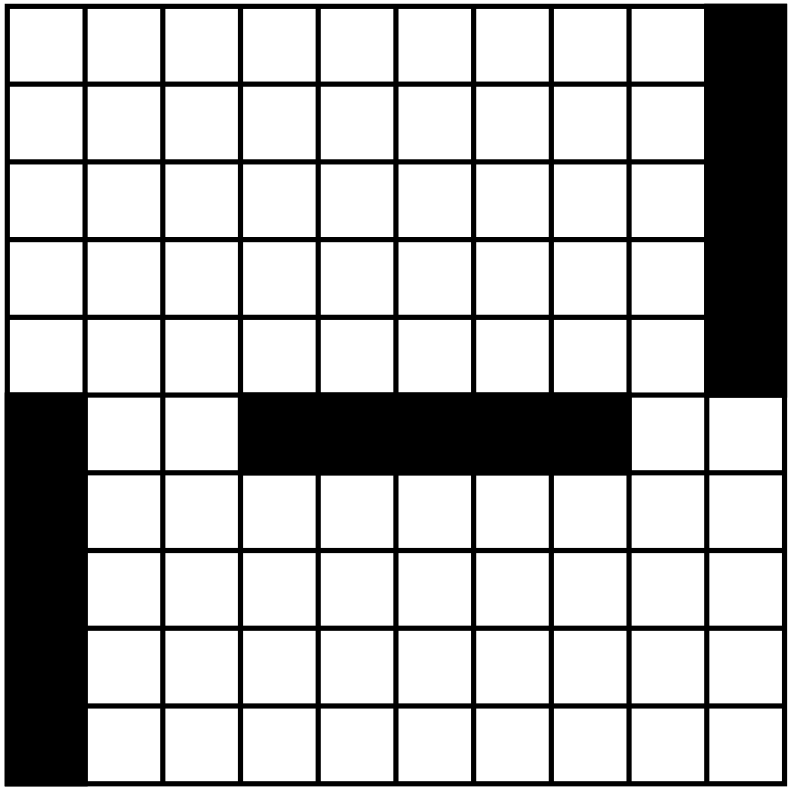
MOBILE ROBOTICS

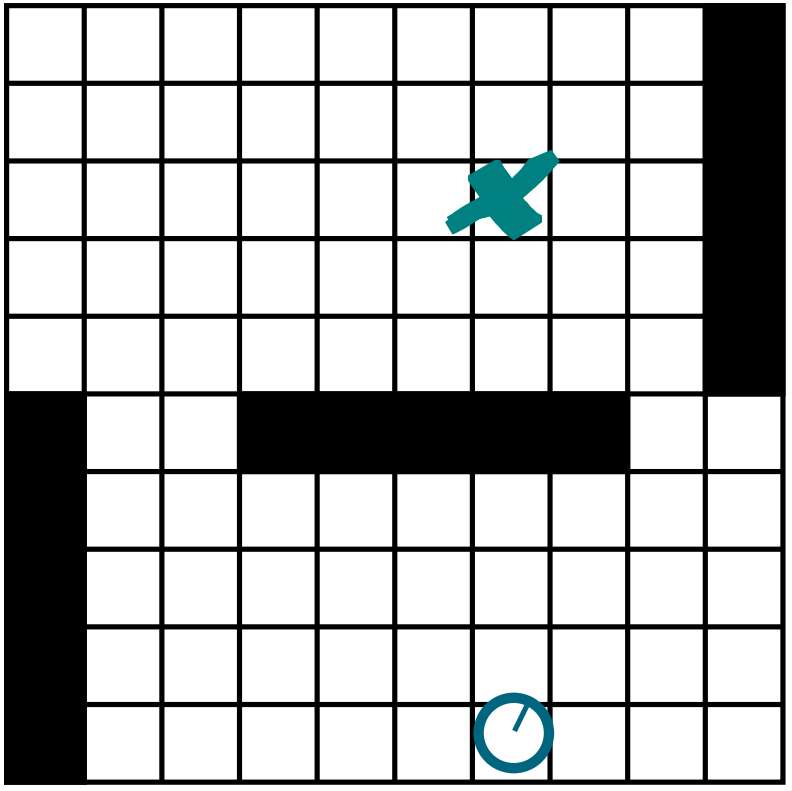
A* Algorithm

Problem presentation



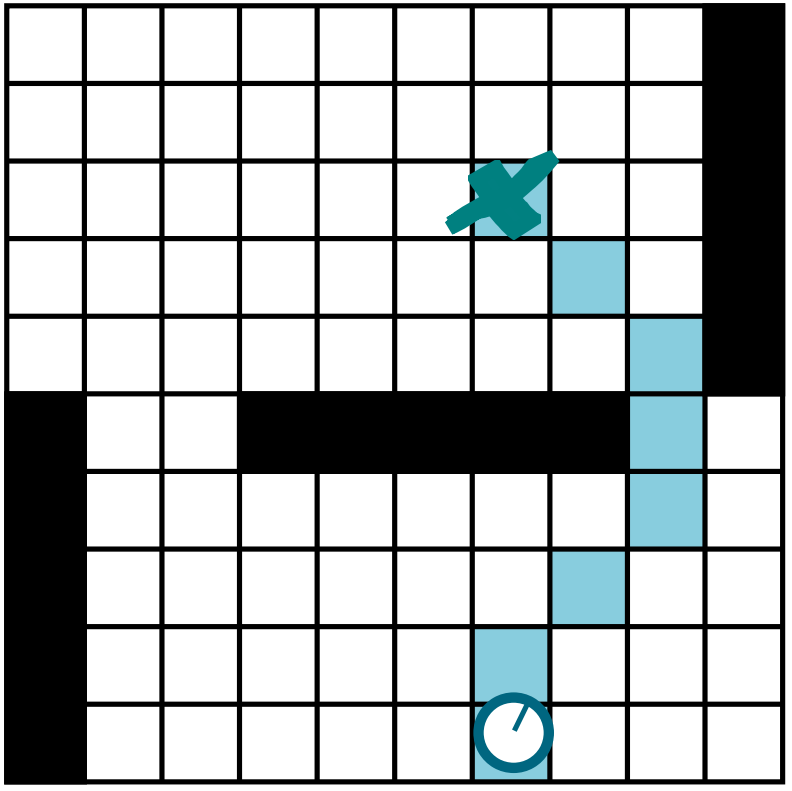
- Map: occupation grid

Problem presentation



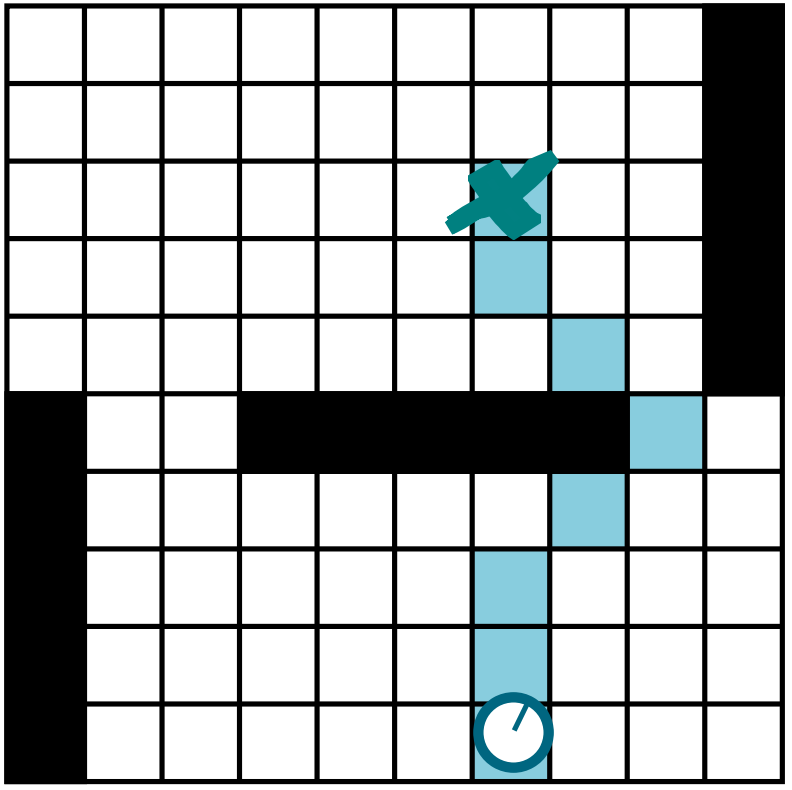
- Map: occupation grid
- Robot’s pose: known
  - At each time step
- Target defined

Problem presentation



- Map: occupation grid
- Robot's pose: known
  - At each time step
- Target defined
- Find the shortest path from the robot's position to the target

Problem presentation

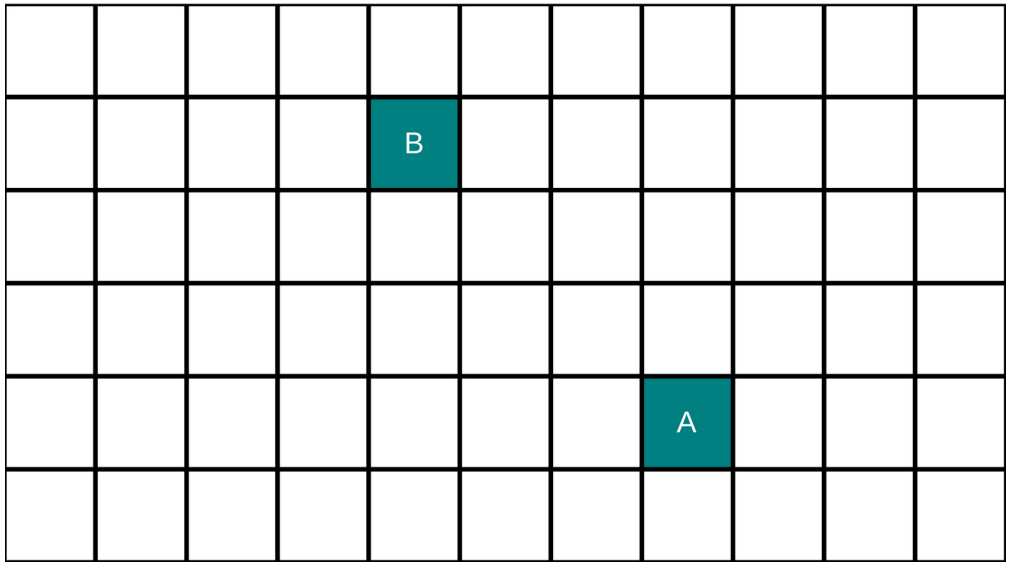


- Map: occupation grid
- Robot's pose: known
  - At each time step
- Target defined
- Find the shortest path from the robot's position to the target
- May be several solutions

Shortest path

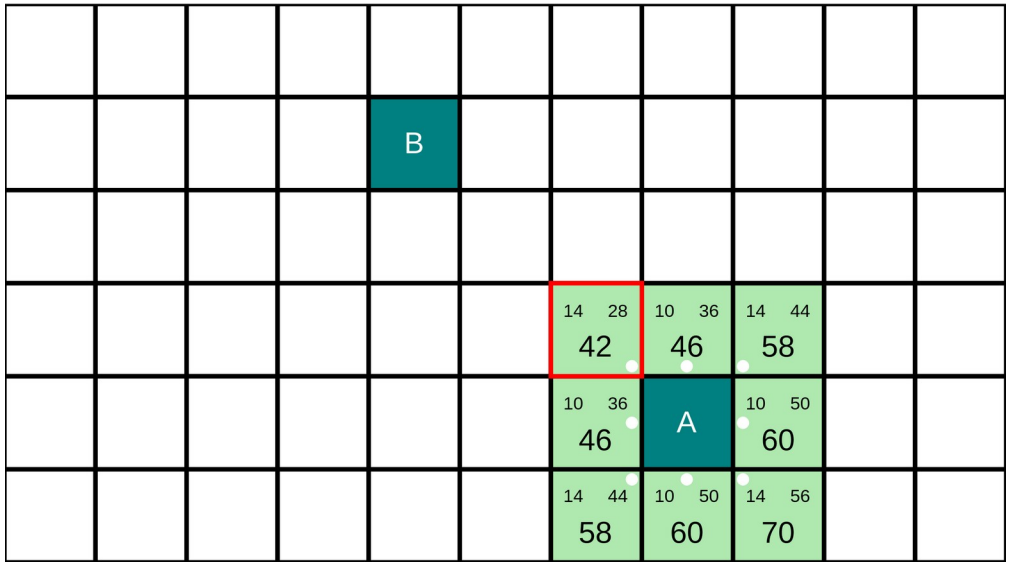
- shortest_path_ui.py
  - Dijkstra
  - A*
  - Weighted A*

A *



Find the shortest path from A to B

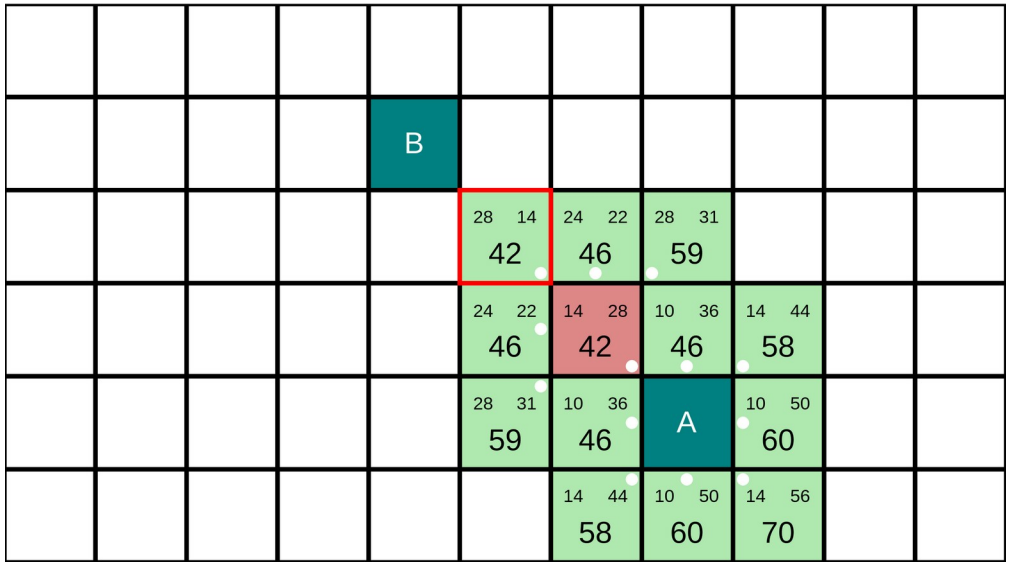
A *



- Compute the costs for all the neighbor nodes
- Tag the nodes as opened (if not opened yet)
- Save the parent node for all the opened nodes
- Select the “smallest” node as next one
  - smallest F cost
    - smallest S cost

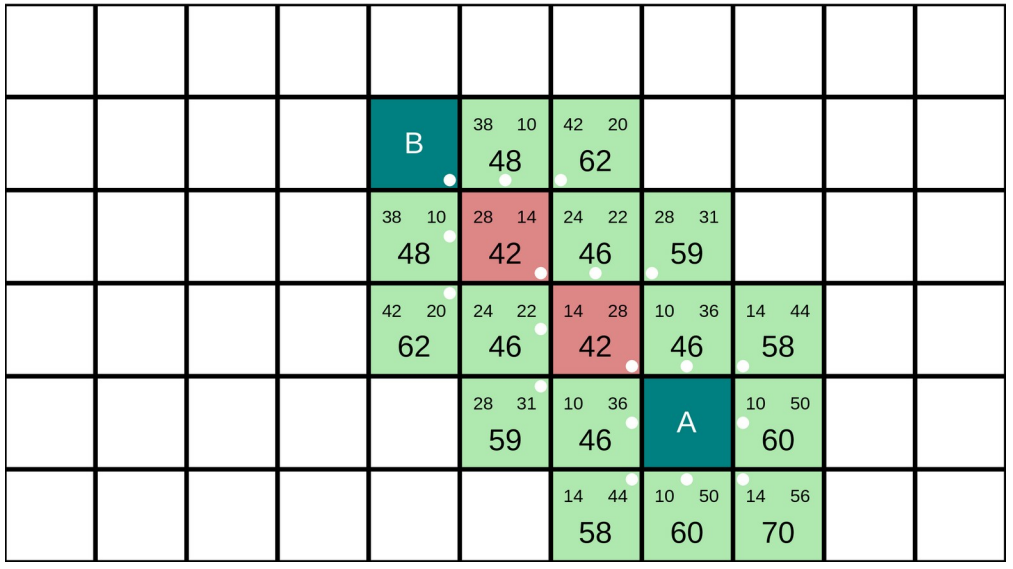
S cost: distance to the start node A (top left corner)
H cost: distance to the target node B (heuristic, top right corner)
F cost: S cost + H cost (center)
 - To ease the computation, the costs are: distance*10 (i.e. sqrt(2)=14)

A *



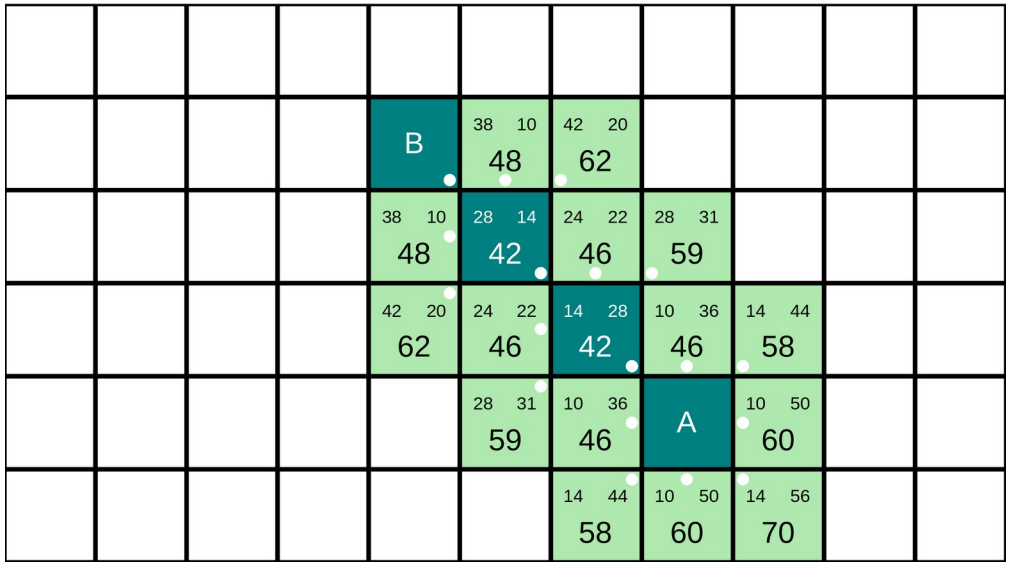
Tag each processed node as closed

A *



… until you reach the target

A *



- Then you just have to go from the target back to the start, one parent at a time

Work to do

- Files to upload to Moodle
  - tp_astar/node.py
  - tp_astar/a_star.py
  - tp_astar/utils.py
- Warnings
  - Test your code functions after functions
  - Do not modify the other files
  - Do not add any library (numpy for instance)
- You should use run_astar_tests to run unit tests

__lt__() : lower than

- Function called when doing the operator node1 < node2
- A node is lower than an other if the F cost is lower. If both F costs are the same, the lower node is the one with the lower S cost

update_costs()

- This function update the costs of the node
  - Always keep the smallest costs (S cost and H cost)
  - Update the node parent if needed
  - Update the F cost when needed (if S cost or H cost changes)
    - F cost = S cost + H cost

get_world_coordinates_from_index()

- This function provides a Point2D with world coordinates according to a Point2D with grid indexes
  - From a grid cell x and z indexes, provides the x and z world coordinates of that cell center

heuristic()

- Function that returns the euclidean distance (x10) from the position to the target

udpate_node()

- This function update a node according to a parent
  - pos_parent: the parent of the node
  - dx, dz the coordinates of the node according to the parent
    - dx, dz = {-1, 0, 1}
    - The considered node is a neighbor of the parent node
- Remarks:
  - An obstacle should not be updated
  - The distance to the parent should be 14 (if diagonal) or 10
  - The costs have to be updated
  - If the node was not processed yet, it should be noted opened and added to the opened list

find_path()

- This function implements the A* algorithm
- The instructions are:
  - Loop over the opened nodes until
    - There is no opened nodes anymore
    - The target is found
  - While looping over the opened nodes
    - Close the smallest node
    - Open all the neighbor
  - If the target has been found
    - Update all the nodes that are part of the path (is_path attribute)
    - Update the path list that should contain all the path positions (for the robot)

follow_path()

- Function that uses the path of the A* to move the robot to the target
- The robot’s position is known (robot variable)
- The path should be updated each time the robot reaches one step