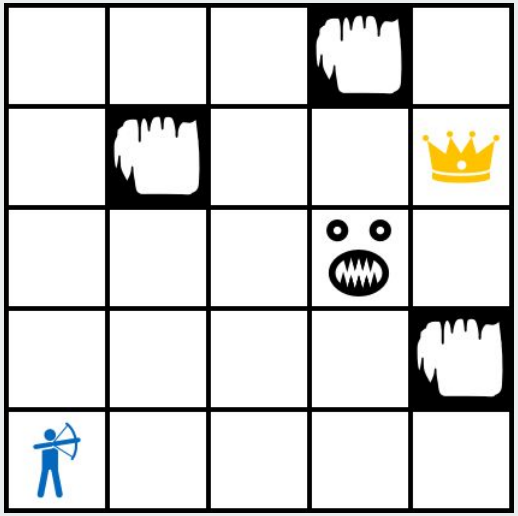
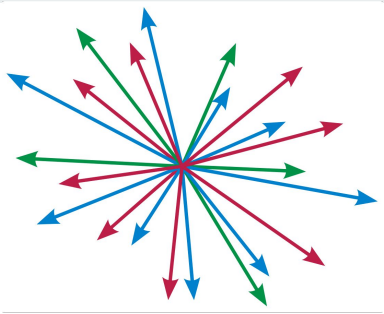
Hunt The Wumpus



Green Team

Marco Di Panfilo, Alessandra Lorefice, Denis Mugisha, Gianluigi Pellè

World environment - Linear space



linear_space.SmartCoordinate

- __init__(self, x=0, y=0)
- __neg__(self)
- __add__(self, other)
- __radd__(self, other)
- __sub__(self, other)
- __hash__(self)
- __eq__(self, other)
- __lt__(self, other)
- __str__(self)
- __repr__(self)

- x
- y

linear_space.SmartVector

- from_coordinate(coordinate)
- get_north()
- get_east()
- get_south()
- get_west()
- __init__(self, x=0, y=0)
- __neg__(self)
- __add__(self, other)
- __radd__(self, other)
- __sub__(self, other)
- __mul__(self, other)
- __rmul__(self, other)
- __hash__(self)
- __eq__(self, other)
- __str__(self)
- __repr__(self)
- get_perpendicular_vector_clockwise(self)
- get_perpendicular_vectors(self)

- x
- y

Architecture Design

hybrid_agent_model_WIP.HuntWumpusHybridAgent

- __init__(self)
- get_next_action_from(self, *, percept)
- plan_route(self, *, to_goal_locations, with_allowed_locations, ...)
- astar_search(self, *, problem)

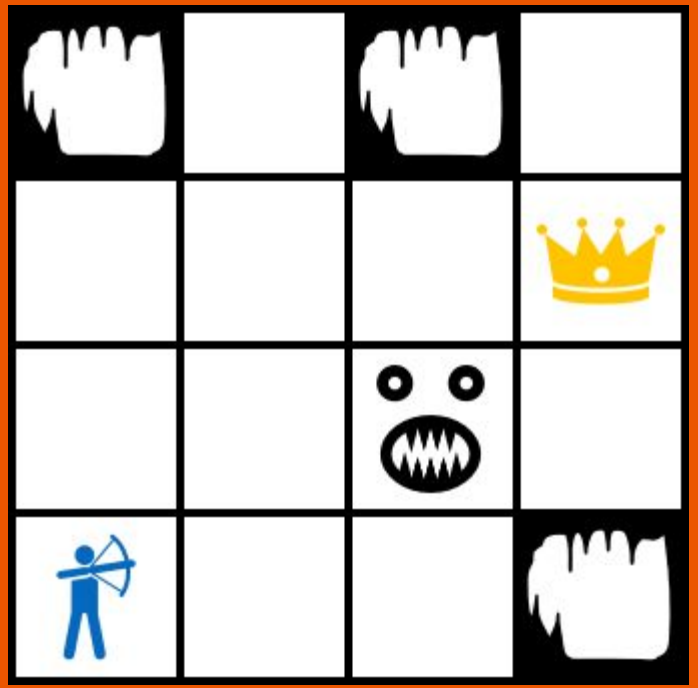
- kb
- plan_actions
- previous_agent_action

hybrid_agent_model_WIP.KnowledgeBase

- __init__(self)
- __str__(self)
- __repr__(self)
- get_adjacent_locations_from(self, *, location)
- filter_locations_from_unknown_world(self, *, from_locations)
- update_with(self, *, previous_agent_action, current_percept)
- update_wumpus(self, *, with_possible_wumpus_locations=set(), with_no_wumpus_locations=set(), scream=False)
- get_safe_locations(self)
- get_safe_unvisited_locations(self)
- get_safe_from_pit_but_possible_wumpus_locations(self)
- get_smt_bool_variables_from(self, *, locations, with_prefix)
- add_assertions(self, *, for_variables, with_value, to_solver)
- get_bool_value_if_certain(self, *, of_bool_variable, in_solver)
- update_pit_locations(self)
- make_bayesian_proposition(self, *, with_name, and_values, and_probabilities)
- make_evidence_bool_bayesian_proposition(self, *, with_name, and_values, and_evidence, with_bool_function)
- make_bayesian_bool_propositions_dict(self, *, for_locations, with_name_prefix, and_probabilities)
- get_bayesian_model(self)
- get_evidence_dict(self)
- check_pit_probability_of(self, *, location, bayesian_model, evidence_dict)
- check_safe_probability_of(self, *, location, bayesian_model, evidence_dict)

- is_arrow_available
- pit_SAT_solver
- known_wumpus_location
- no_pit_locations
- is_wumpus_alive
- has_agent_bumped_east
- pit_locations
- visited_locations
- possible_wumpus_locations
- has_agent_bumped_north
- exit_locations
- world_size_height
- no_wumpus_locations
- agent_location
- world_size_width
- breeze_locations
- agent_orientation
- fringe_locations
- no_breeze_locations

Our Strategy



Overall strategy

1) If perceive glitter: grab gold, go to exit and climb.
2) If there are safe cells: explore them.
3) Calculate lowest pit probability of cells in fringe:
    a) Risk to go to nearest cell with lowest pit probability that is below risk threshold.
    b) If all cells above risk threshold: go to exit.

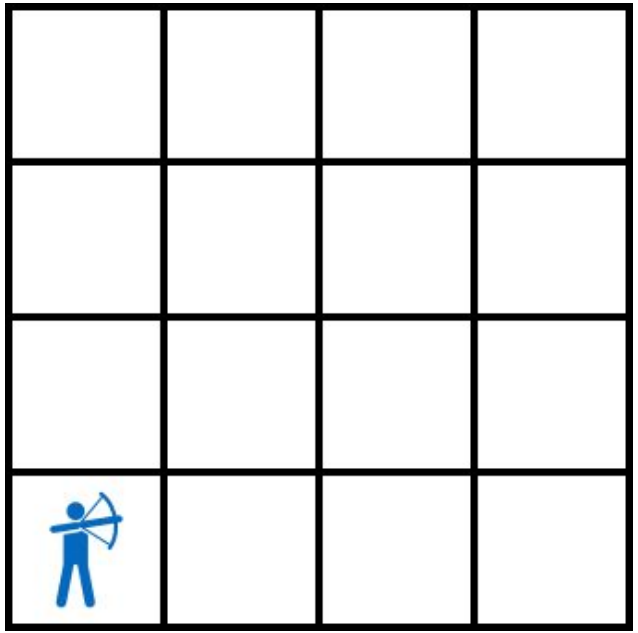
Start with world size (1, 1) and increase each time a cell outside the known range is visited and no bump is perceived.

Wumpus strategy

1) Try to locate Wumpus when exploring safe cells and cells with a lower pit probability than possible Wumpus.
2) Once located, it behaves like other cells with a cost of +10 to shoot if an arrow is available.
3) If localisation is not possible and several locations may have a Wumpus, shoot the closer one.

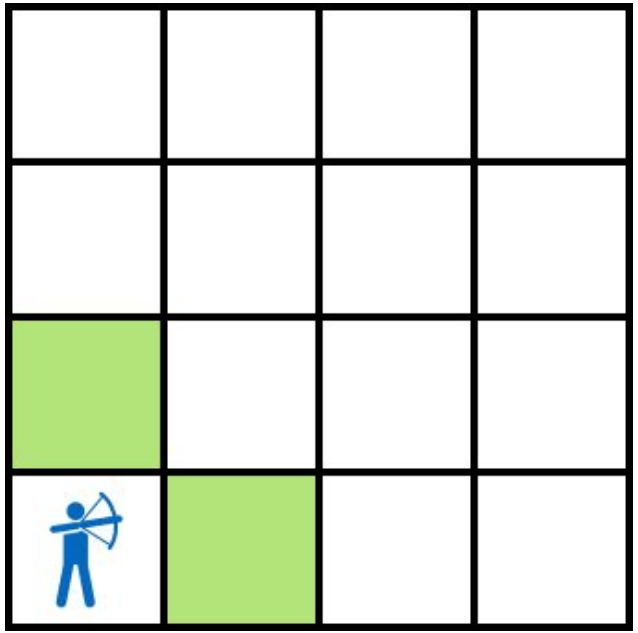
Check all the safe cells

- Neighbours of cells with **no stench** and **no breeze** are safe
- If we identified the precise position of the wumpus we can consider it safe by shooting him



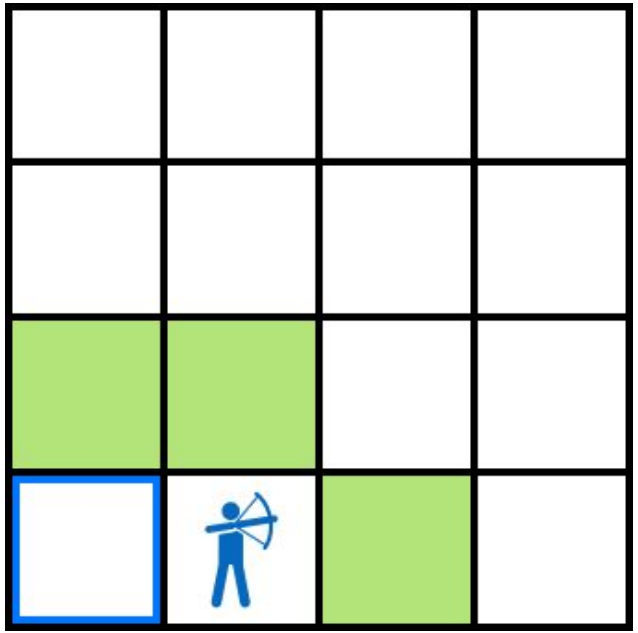
Check all the safe cells

- Neighbours of cells with **no stench** and **no breeze** are safe
- If we identified the precise position of the wumpus we can consider it safe by shooting him



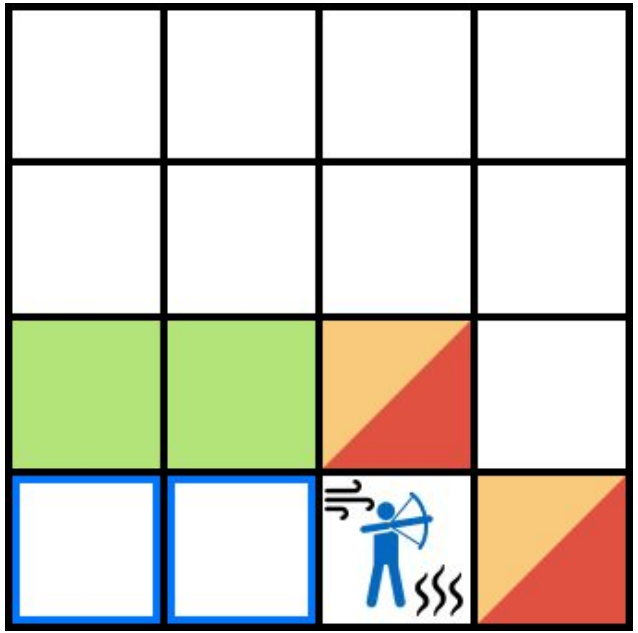
Check all the safe cells

- Neighbours of cells with **no stench** and **no breeze** are safe
- If we identified the precise position of the wumpus we can consider it safe by shooting him



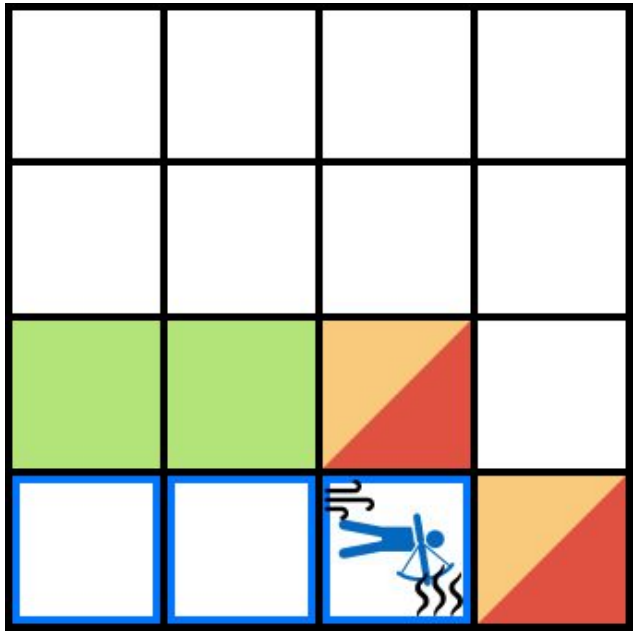
Found breeze and stench

- When we perceive **stench** we add not visited neighbour cells to **possible wumpus**
- When we perceive breeze we only memorize that we perceived a breeze



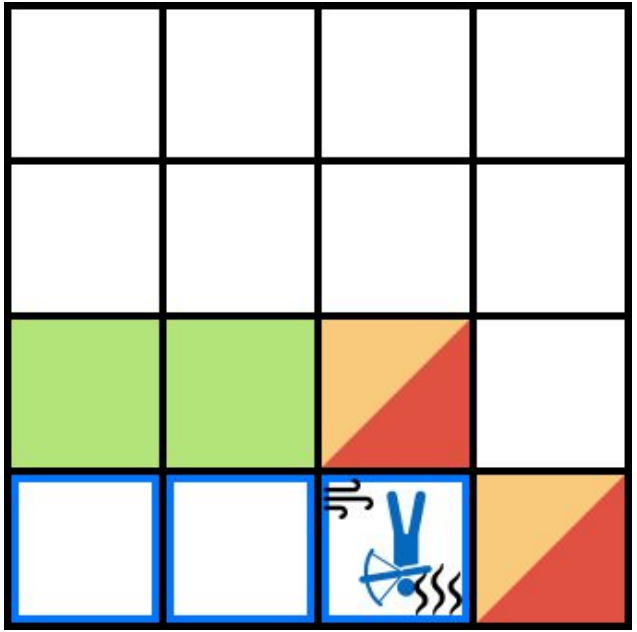
Check all the safe cells

- When we perceive stench we add not visited neighbour cells to possible wumpus
- When we perceive breeze we only memorize that we perceived a breeze



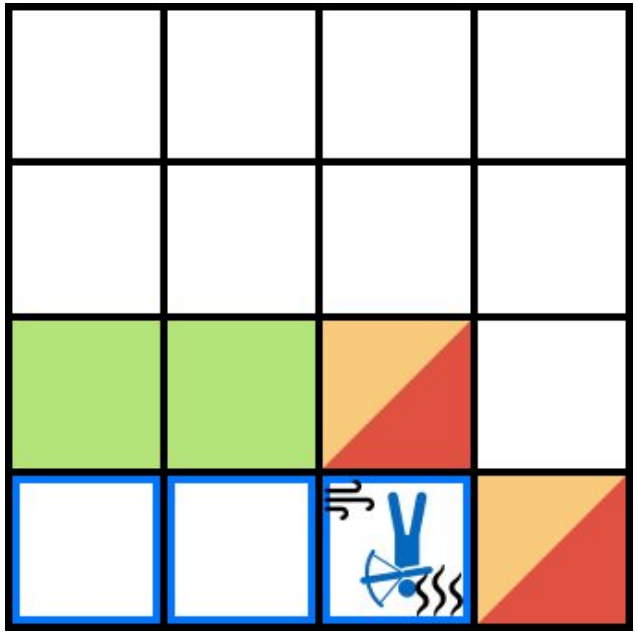
Check all the safe cells

- When we perceive stench we add not visited neighbour cells to possible wumpus
- When we perceive breeze we only memorize that we perceived a breeze



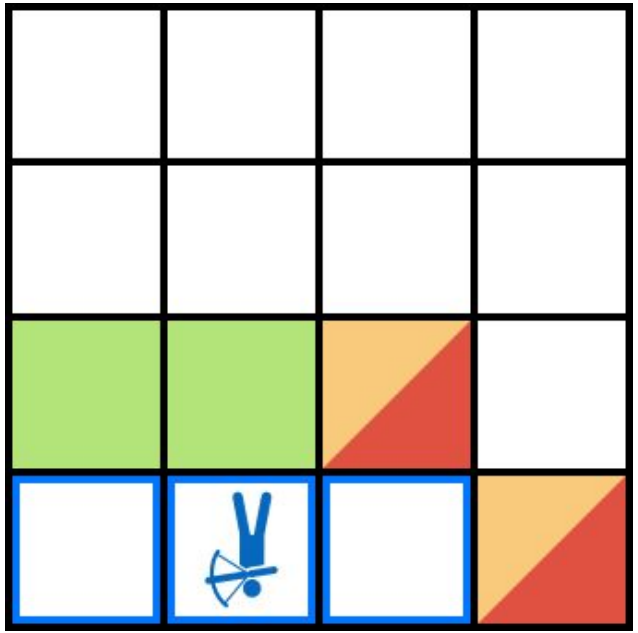
Check all the safe cells

- When we perceive stench we add not visited neighbour cells to possible wumpus
- When we perceive breeze we only memorize that we perceived a breeze



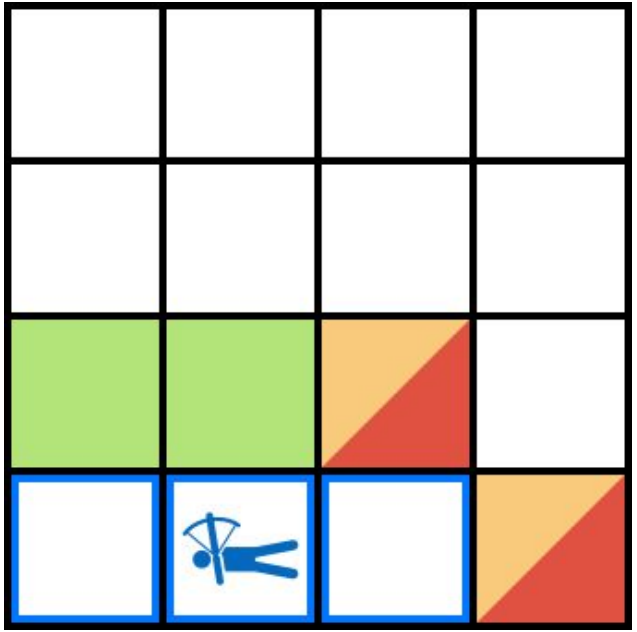
Check all the safe cells

- When we perceive stench we add not visited neighbour cells to possible wumpus
- When we perceive breeze we only memorize that we perceived a breeze



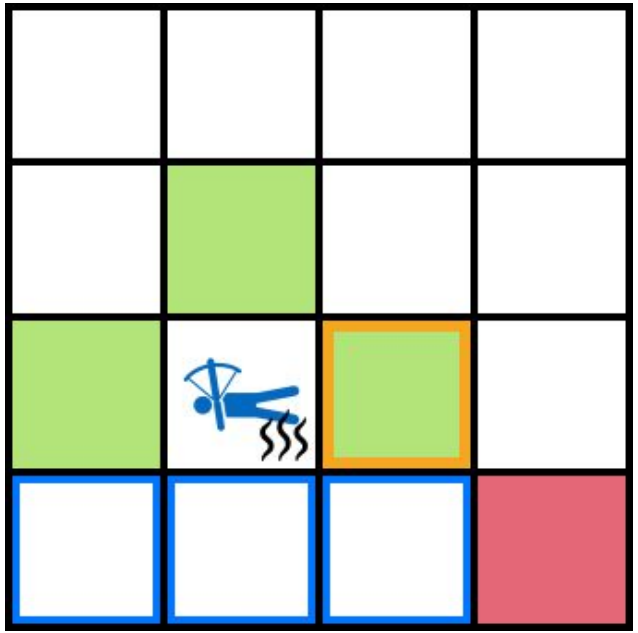
Check all the safe cells

- When we perceive stench we add not visited neighbour cells to possible wumpus
- When we perceive breeze we only memorize that we perceived a breeze



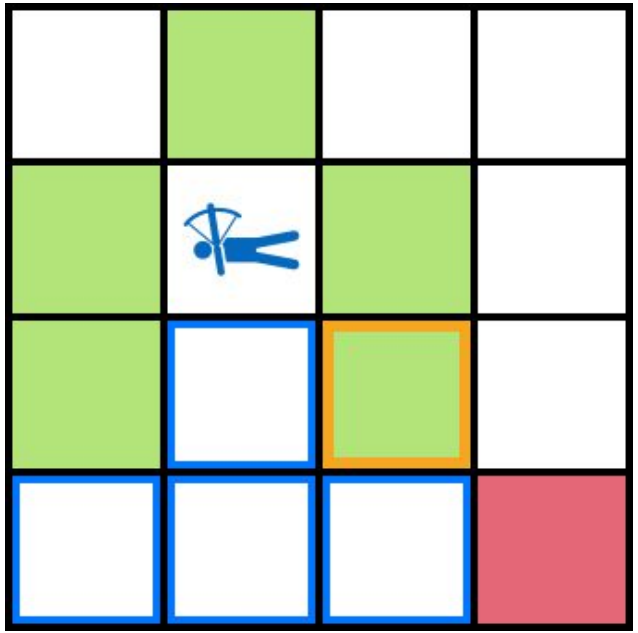
Found second stench

- When we perceive a **stench again**, we can do the **intersection** of current neighbours and the previous possible wumpus cells and **locate the wumpus** if there is only 1 common element (orange border).
- Wumpus can be considered safe, with a higher cost, if we still have the arrow to shoot him.



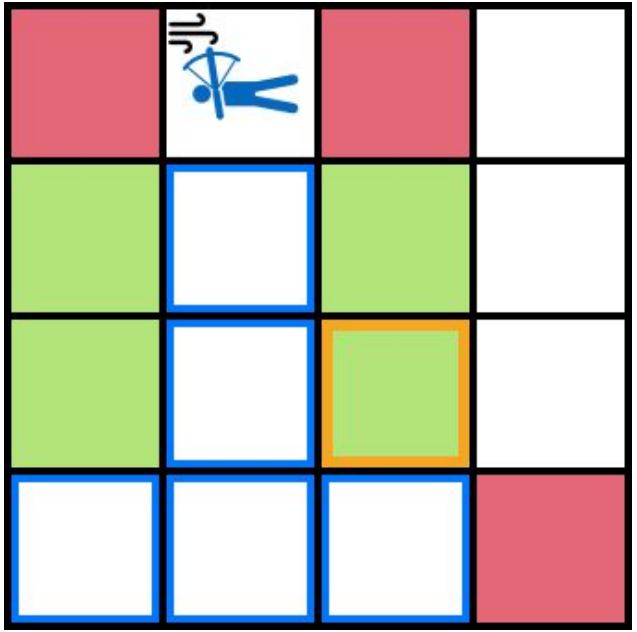
Check all the safe cells

- When we perceive a **stench again**, we can do the **intersection** of current neighbours and the previous possible wumpus cells and **locate the wumpus** if there is only 1 common element (orange border).
- Wumpus can be considered safe, with a higher cost, if we still have the arrow to shoot him.



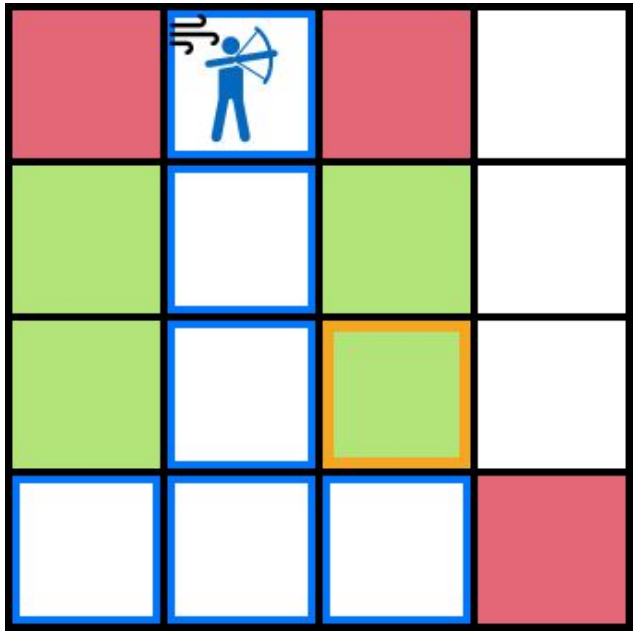
Found breeze

- When we feel breeze we don’t add the neighbours to the safe cells and look for remaining safe cells to explore.



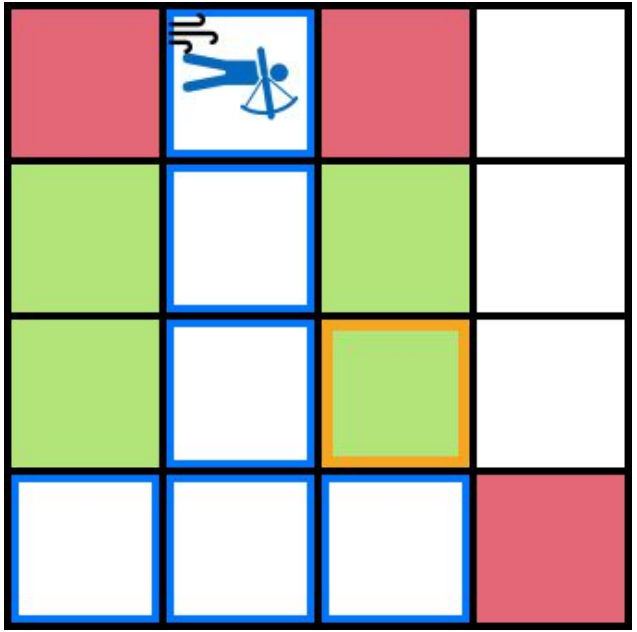
Check all the safe cells

- Exploring closest safe cells



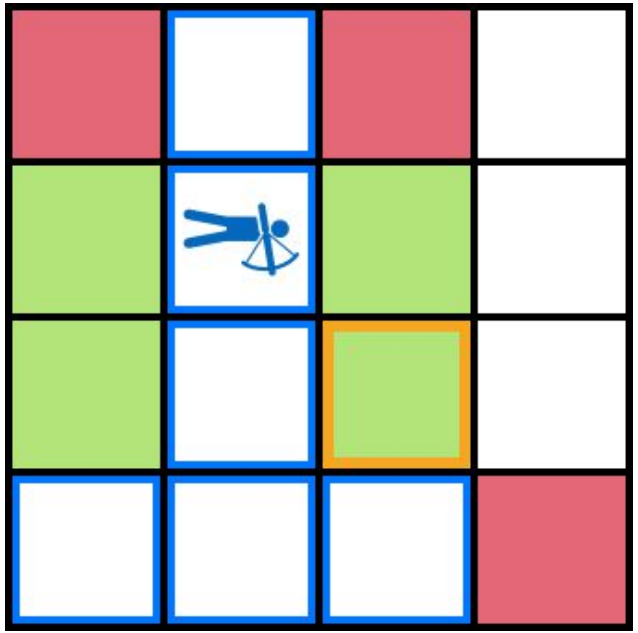
Check all the safe cells

- Exploring closest safe cells



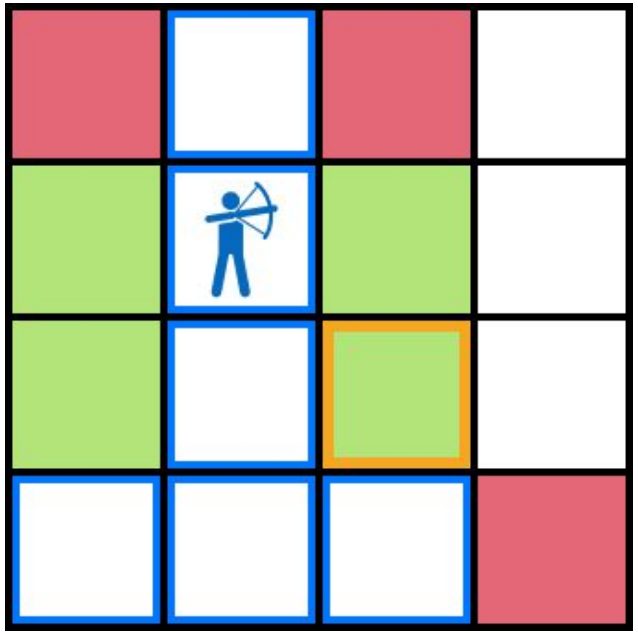
Check all the safe cells

- Exploring closest safe cells



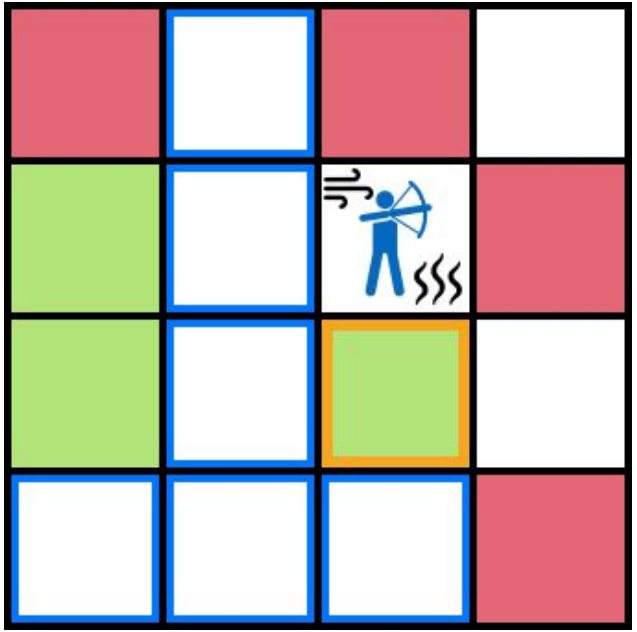
Check all the safe cells

- Exploring closest safe cells



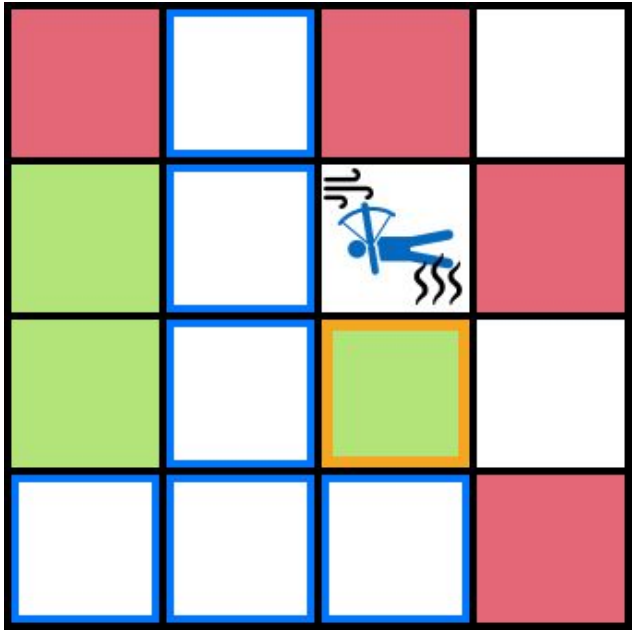
Found breeze and stench

- When we feel breeze we don’t add the neighbours to the safe cells and look for remaining safe cells to explore.



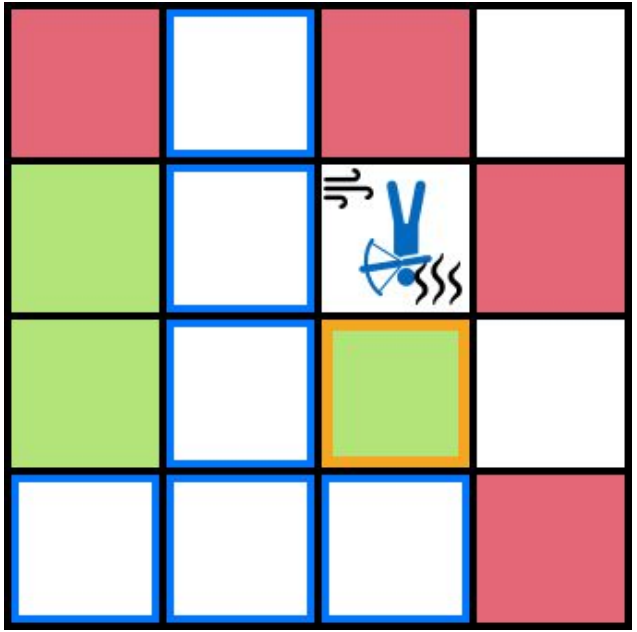
Check all the safe cells

- Exploring closest safe cells



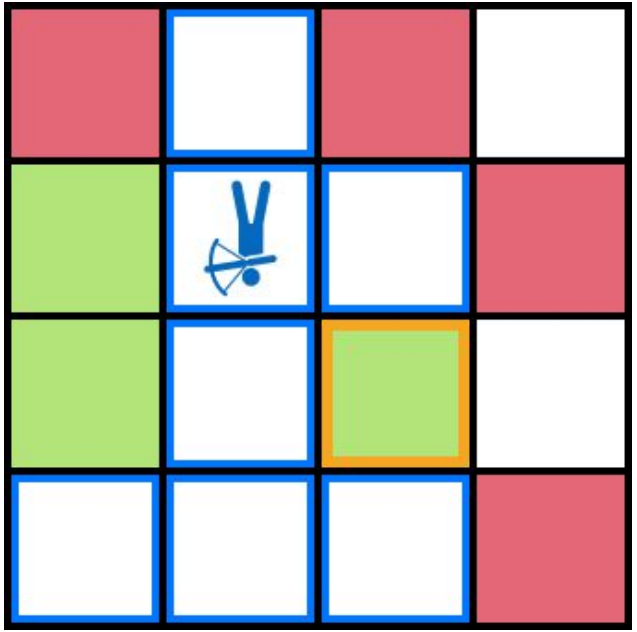
Check all the safe cells

- Exploring closest safe cells



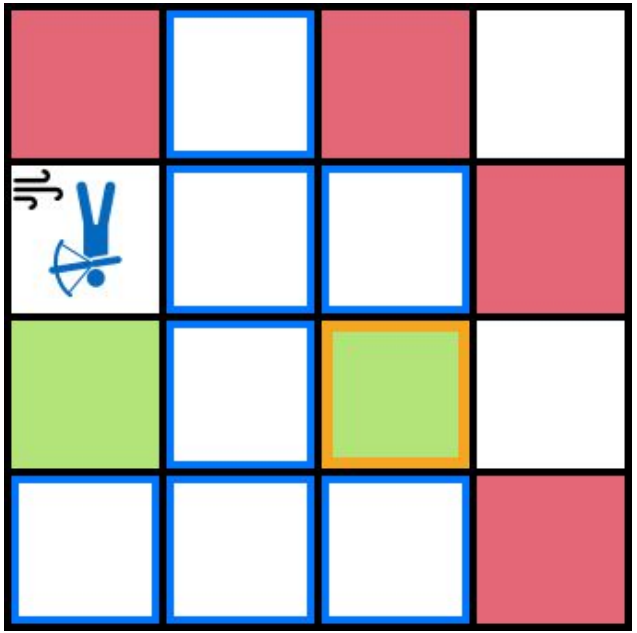
Check all the safe cells

- Exploring closest safe cells



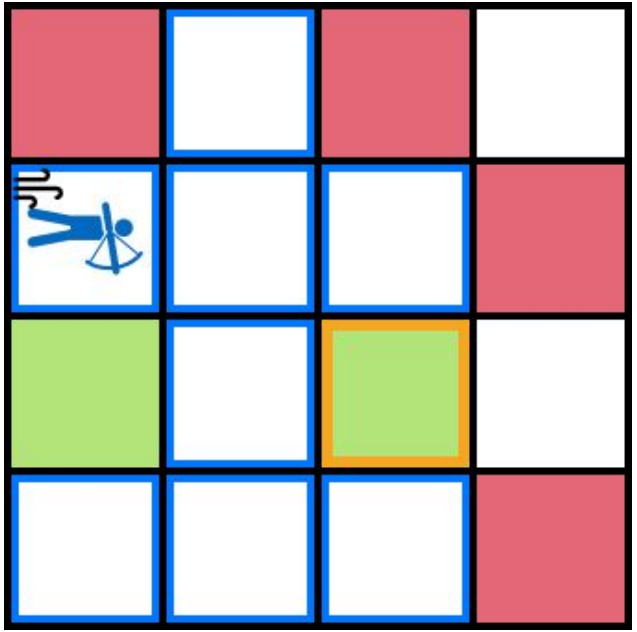
Check all the safe cells

- Exploring closest safe cells



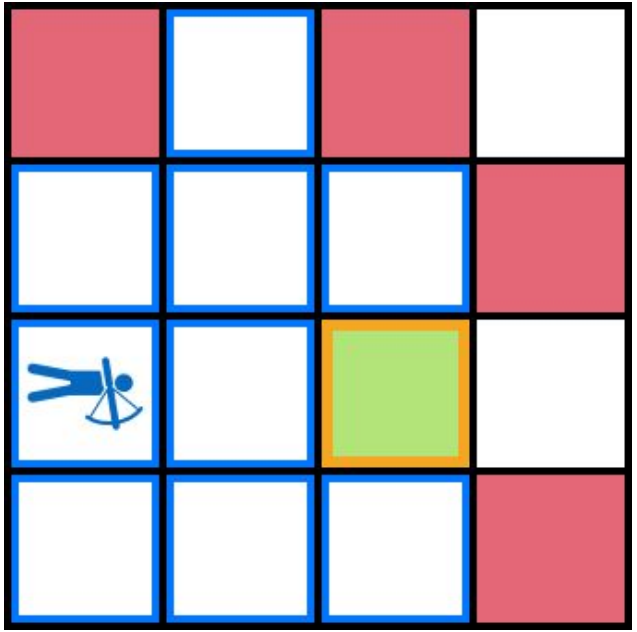
Check all the safe cells

- Exploring closest safe cells



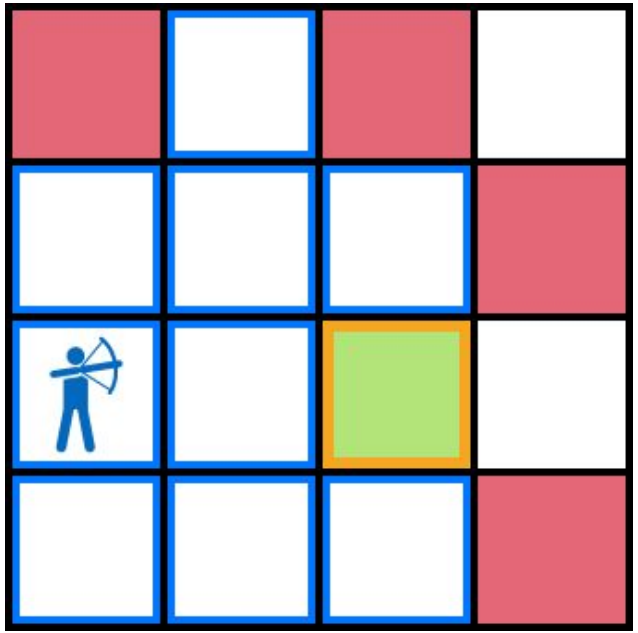
Check all the safe cells

- The final safe cell is the one with the wumpus. We have to align to the wumpus and kill him.



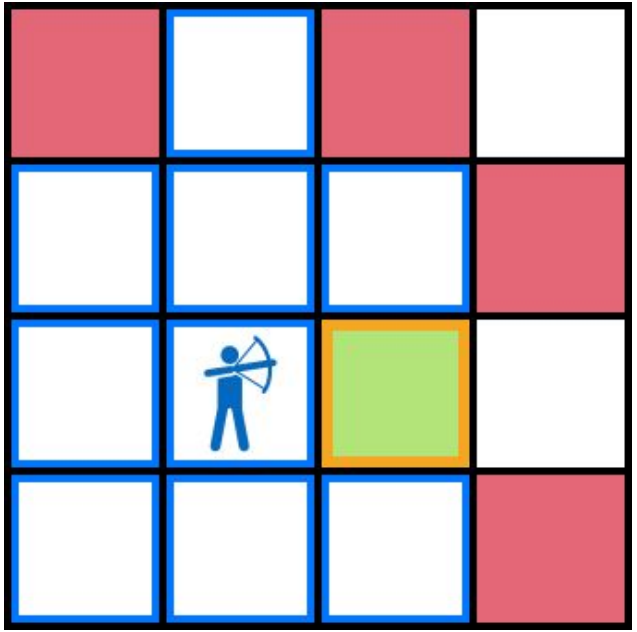
Check all the safe cells

- The final safe cell is the one with the wumpus. We have to align to the wumpus and kill him.



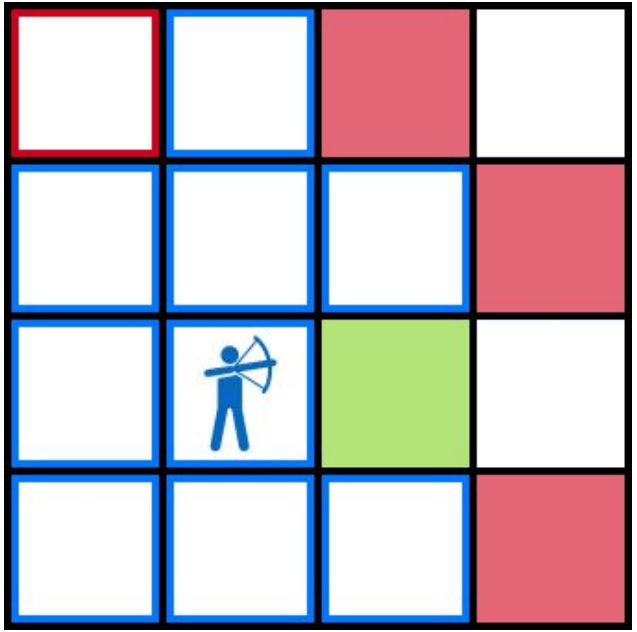
Check all the safe cells

- The final safe cell is the one with the wumpus. We have to align to the wumpus and kill him.



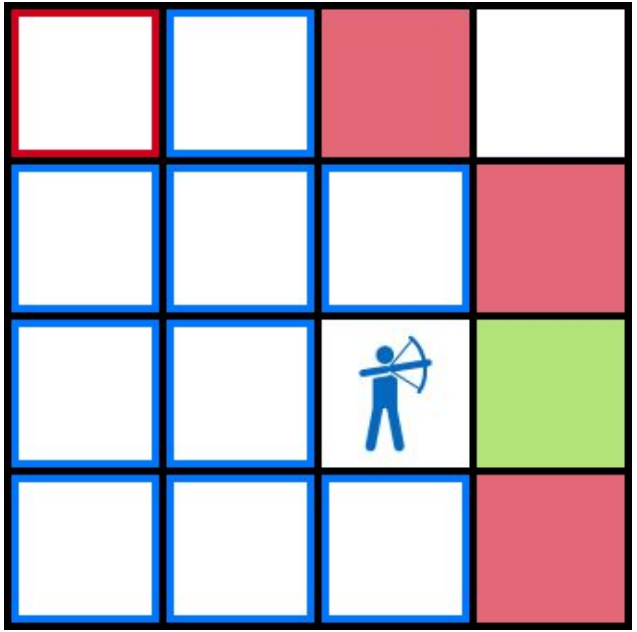
Check all the safe cells

- The final safe cell is the one with the wumpus. We have to align to the wumpus and kill him.



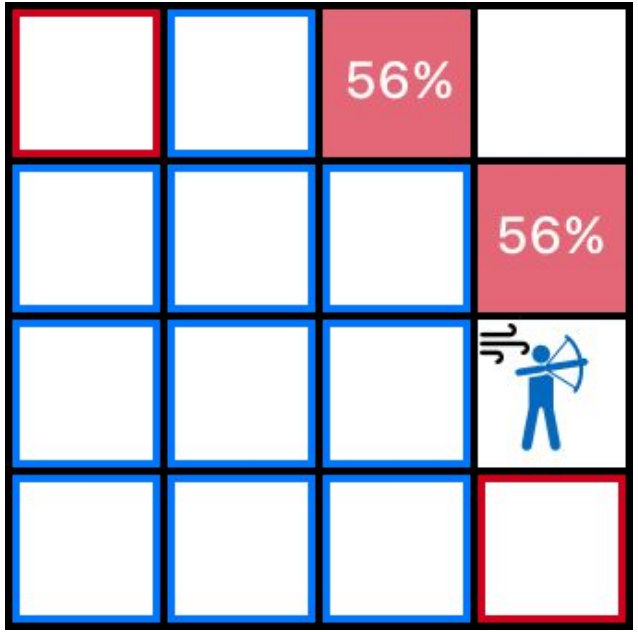
Check all the safe cells

- Exploring closest safe cells



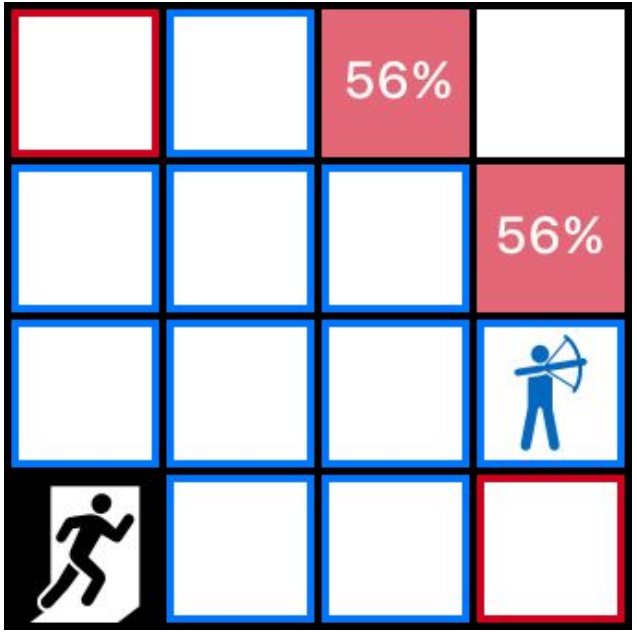
Search safest cell in fringe

- We calculate the certain pit on the corners with a SAT Solver and then we calculate the pit probability from the remaining cells in the fring by constructing a Bayesian Network.
- Since all the cells have a higher pit probability then the threshold the agent decides to go home.



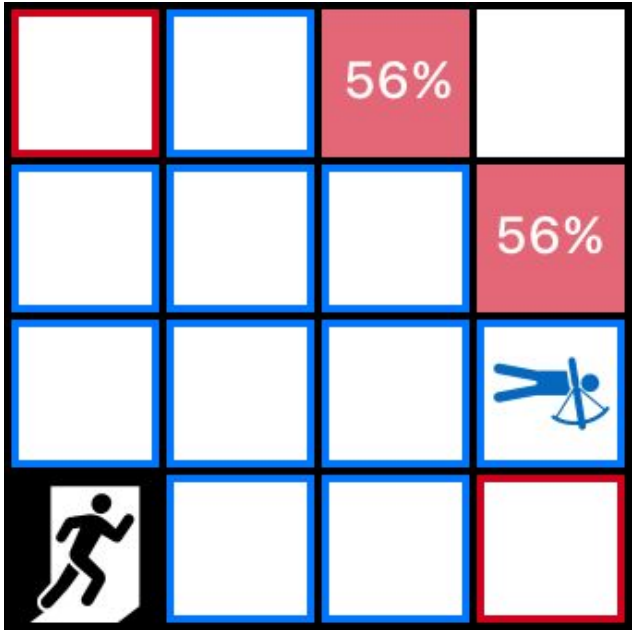
Going to exit

- Returning home without gold



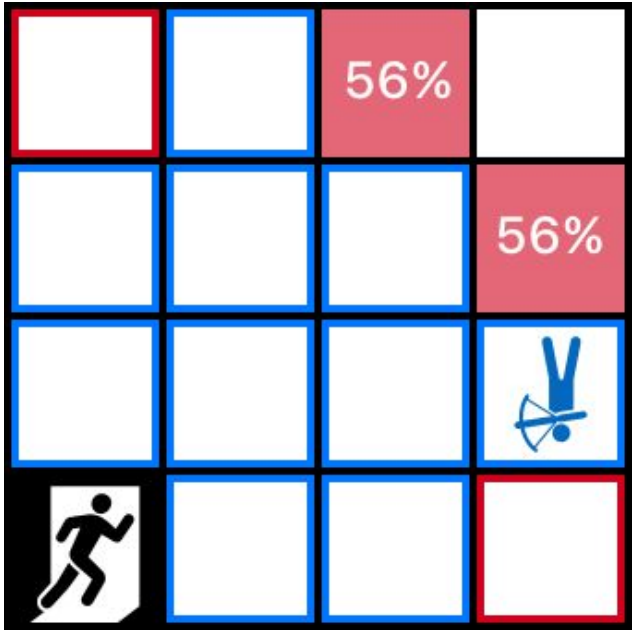
Going to exit

- Returning home without gold



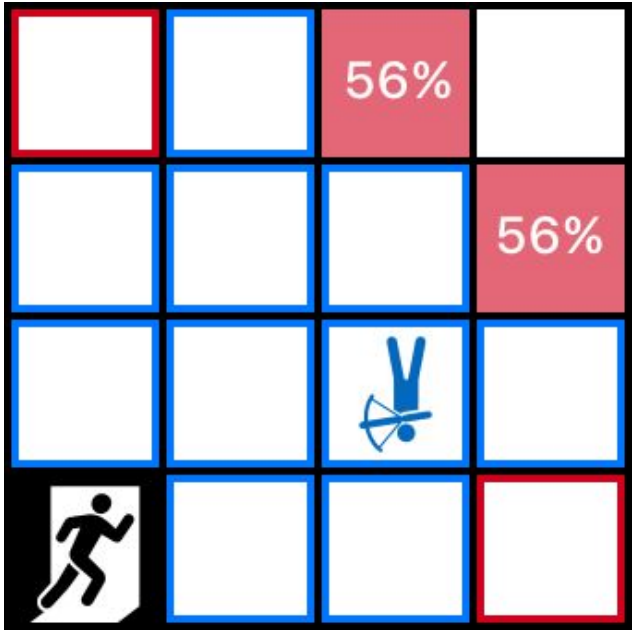
Going to exit

- Returning home without gold



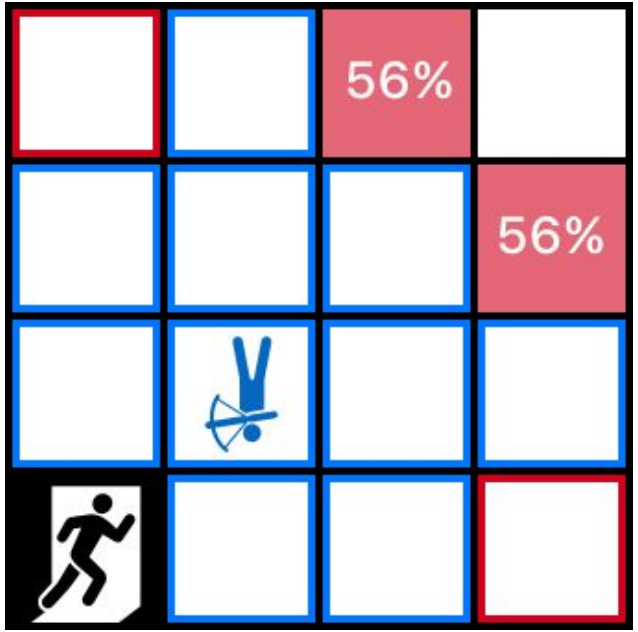
Going to exit

- Returning home without gold



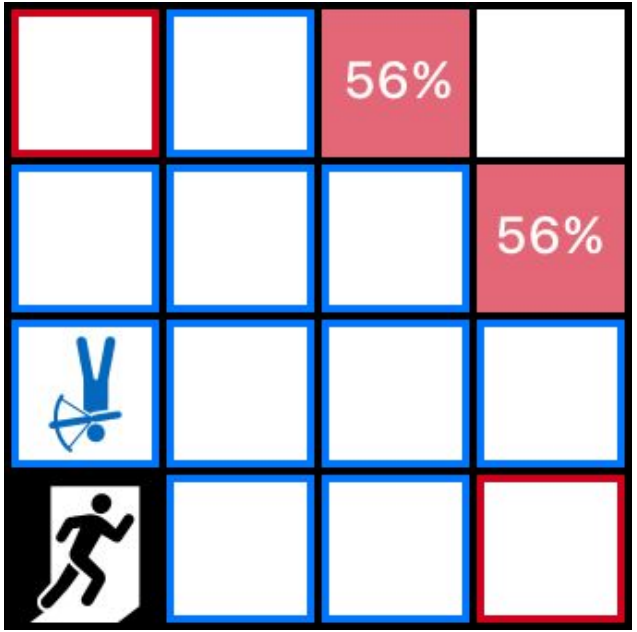
Going to exit

- Returning home without gold



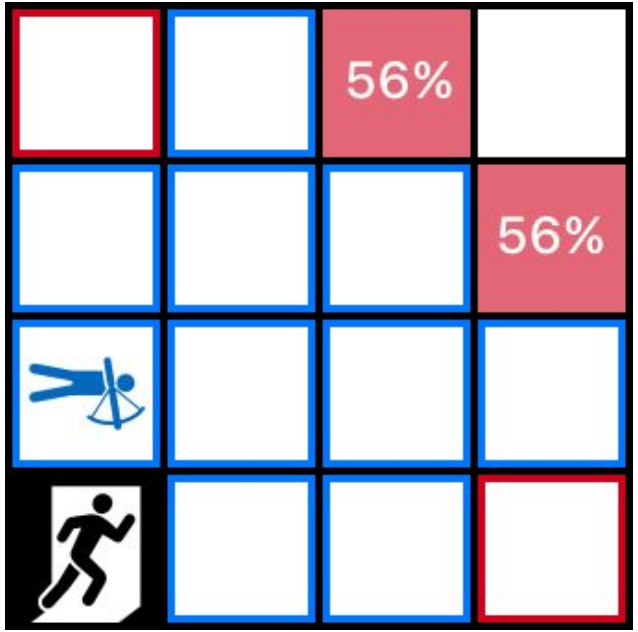
Going to exit

- Returning home without gold



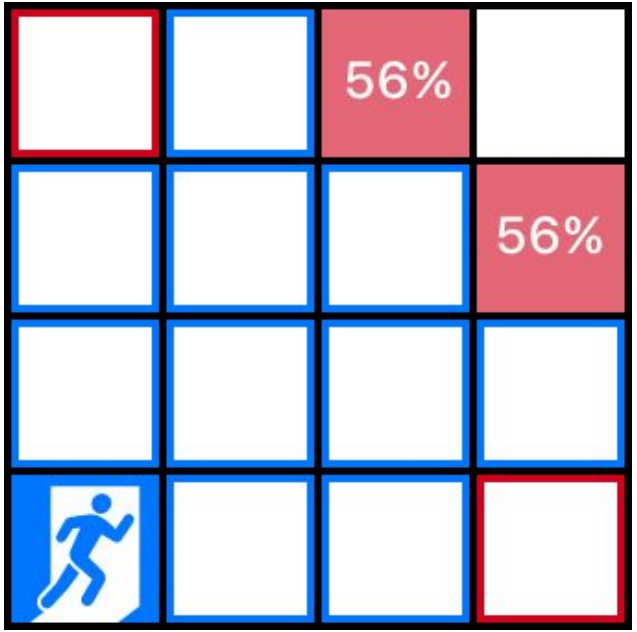
Going to exit

- Returning home without gold



Going to exit

- Returning home without gold



Offline search - A* search

- Three different search problems:

1. We grabbed the gold and we need a way back to the exit
2. The best path to reach the closest safe location
3. The best path to go and kill the most convenient wumpus location (if more than one are available)

We grabbed the gold and we need a way back to the exit

Safe_locations: all locations we are sure to be safe from pits (including the ones with possible wumpuses)

Goal_locations: exit location

Possible_actions: [MOVE, LEFT, RIGHT, SHOOT]

Is_goal_state: agent location is in the exit location

Heuristic_func: best: smart_manhattan(from: agent_location, to: exit_location)

Agent_location: agent_location

Agent_orientation: agent_orientation

Is_arrow_available: if the agent has the arrow available

Wumpus_locations: [possible_wumpus_locations]

Best path to reach the closest safe location

Safe_locations: all locations we are sure to be safe from pits (including the wumpus location if it is known)

Goal_locations: safe locations inside the fringe

Possible_actions: [MOVE, LEFT, RIGHT, SHOOT]

Is_goal_state: agent location is in a safe location inside the fringe

Heuristic_func: minimum smart_manhattan(from: agent_location, to: goal_location) of each goal location

Agent_location: agent_location

Agent_orientation: agent_orientation

Is_arrow_available: if the agent has the arrow available

Wumpus_locations: [possible_wumpus_locations]

Best path to go and kill the most convenient wumpus location

Safe_locations: all locations we are sure to be safe from pits (including the ones with the possible wumpuses we are going to shoot)

Goal_locations: possible wumpuses we are going to shoot

Possible_actions: [MOVE, LEFT, RIGHT, SHOOT]

Is_goal_state: agent location is in one of the possible wumpuses we are going to shoot

Heuristic_func: minimum smart_manhattan(from: agent_location, to: of each possible wumpus location)

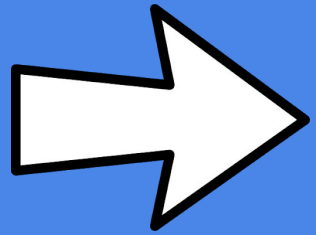
Agent_location: agent_location

Agent_orientation: agent_orientation

Is_arrow_available: if the agent has the arrow available

Wumpus_locations: [the possible wumpuses we are going to shoot]

Tests



Threshold

- Take a risk if probability to be safe > **0.8**
- Chosen according to the **highest average outcome**
- Reduction does not influence strongly the outcome
- The lower is the threshold the longer is the run time (*)

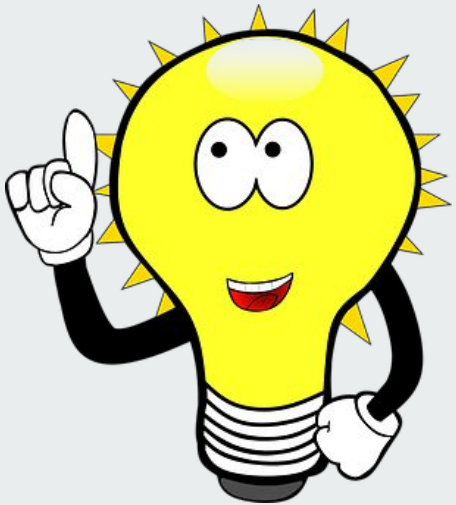
(*) we noticed that when running 200 episodes since the single episode execution only took 0.1 sec

Treshold/Size	4x4	5x5	6x6	7x7	8x8
0.55	avg = 314.135 alive = 96.8%	avg = 250.600 alive = 95.6%	avg = 164.610 alive = 94.8%	avg = 167.325 alive = 94.0%	avg = 86.140 alive = 92.4%
0.60	avg = 330.540 alive = 99.2%	avg = 238.015 alive = 97.6%	avg = 136.360 alive = 91.6%	avg = 167.325 alive = 94.0%	avg = 71.675 alive = 92.0%
0.65	avg = 310.960 alive = 98.8%	avg = 267.075 alive = 98.0%	avg = 149.980 alive = 95.2%	avg = 137.595 alive = 94.0%	avg = 92.400 alive = 92.4%
0.70	avg = 384.300 alive = 98.8%	avg = 206.285 alive = 95.2%	avg = 188.245 alive = 94.8%	avg = 200.990 alive = 95.6%	avg = 66.280 alive = 90.4%
0.75	avg = 376.810 alive = 98.8%	avg = 234.335 alive = 97.6%	avg = 132.640 alive = 91.6%	avg = 148.670 alive = 94.0%	avg = 16.045 alive = 90.4%
0.80	avg = 366.025 alive = 98.4%	avg = 317.870 alive = 97.6%	avg = 234.895 alive = 95.6%	avg = 133.555 alive = 94.8%	avg = 100.130 alive = 92.8%
0.85	avg = 335.865 alive = 99.2%	avg = 274.675 alive = 94.8%	avg = 176.350 alive = 94.8%	avg = 109.655 alive = 92.8%	avg = 58.100 alive = 88.4%

200 worlds tested for each combination of Threshold/Size

Treshold	Average
0.55	192.562
0.60	188.783
0.65	191.602
0.70	209.220
0.75	181.700
0.80	230.495
0.85	190.929

Conclusions



- the performance to get a high reward is influenced by the strategy and the risk threshold
- the calculation of the probability was very slow. SAT Solver showed a reduction in execution time of 5-10 %
- the combination of both the SAT Solver and the Bayesian Network inside the KnowledgeBase class was a good solution

Thank you