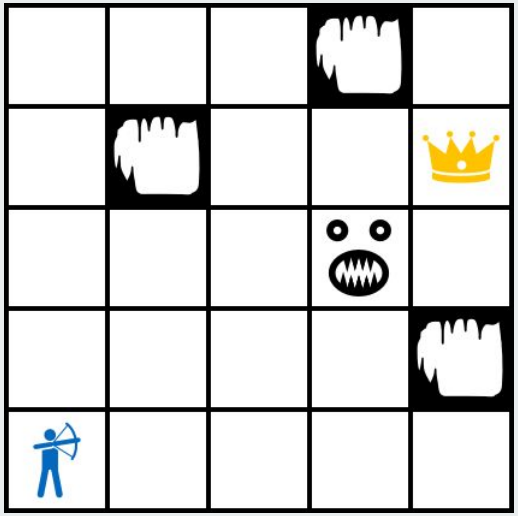
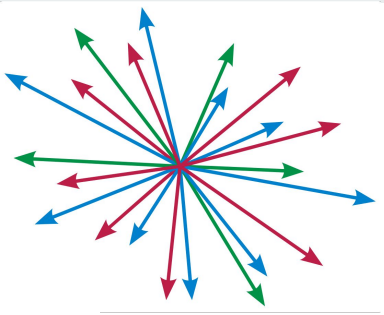
Hunt The Wumpus



Green Team

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

World environment - Linear space



SmartCoordinate

+ x: int
+ y: int

+ __init__()
+ __neg__()
+ __add__(other: SmartCoordinate)
+ __radd__(other: SmartCoordinate)
+ __sub__(other: SmartCoordinate)
+ __hash__()
+ __eq__()
+ __str__()
+ __repr__()

SmartVector

+ x: int
+ y: int

+ __init__()
+ __neg__()
+ __add__(other: SmartVector)
+ __radd__(other: SmartVector)
+ __sub__(other: SmartVector)
+ __mul__(other: SmartVector)
+ __rmul__(other: SmartVector)
+ __hash__()
+ __eq__()
+ __repr__()

+ get_perpendicular_vector_clockwise()
+ get_perpendicular_vectors()

@staticmethod
+ from_coordinate(SmartCoordinate)

Architecture Design

HuntWumpusNode

+ state: HuntWumpusState
+ path_cost: int
+ reward: int
+ previous_action: Hunter.Action
+ parent: HuntWumpusNode

+ __init__()
+ __hash__()
+ __eq__(other: HuntWumpusNode)
+ __lt__(other: HuntWumpusNode)
+ __str__()
+ __repr__()

+ get_cost_heuristic_sum()
+ unwrap_previous_actions()

HuntWumpusState

+ agent_location: SmartCoordinate
+ agent_orientation: SmartVector
+ is_agent_alive: bool
+ is_arrow_available: bool
+ has_agent_climbed_out: bool
+ wumpus_locations: List(SmartCoordinate)
+ gold_locations: List(SmartCoordinate)
+ heuristic_cost: int

@staticproperties
+ world_size: Tuple(int, int)
+ block_locations = List(SmartCoordinate)
+ pit_locations = List(SmartCoordinate)
+ exit_locations = List(SmartCoordinate)

+ __init__()
+ __hash__()
+ __eq__(other: HuntWumpusState)
+ __str__()
+ __repr__()

Architecture Design

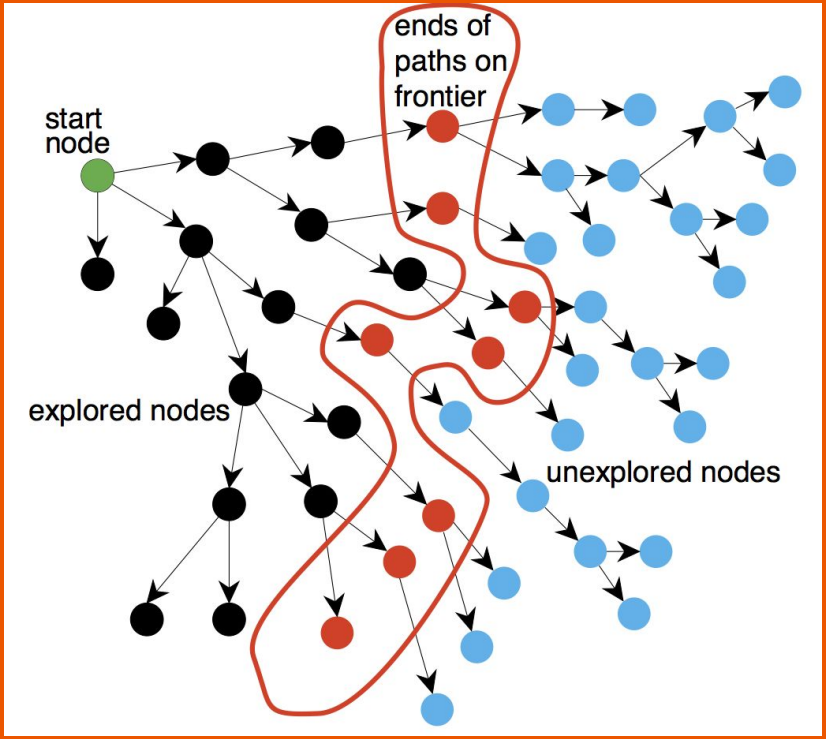
HuntWumpusProblem

+ initialState: HuntWumpusState
+ possible_actions: List(Hunter.Actions)
+ heuristic_func: function
+ action_cost: Dictionary(Hunter.Action: int)
+ action_reward: Dictionary(Hunter.Action: int)

+ __init__()
+ is_legal(location: SmartCoordinate,
*for_state: HuntWumpusState)
+ get_available_actions_for(state: HuntWumpusState)
+ get_effective_actions_for(state: HuntWumpusState)
+ get_best_actions_for(state: HuntWumpusState)
+ get_successor_state_from(state: HuntWumpusState,
*with_action: Hunter.Action)
+ is_goal_state(state: HunterWumpusState)
+ get_child_from(node: HuntWumpusNode,
*with_action: Hunter.Action)
+ unwrap_solution(node: HunterWumpusNode)

Search Algorithms

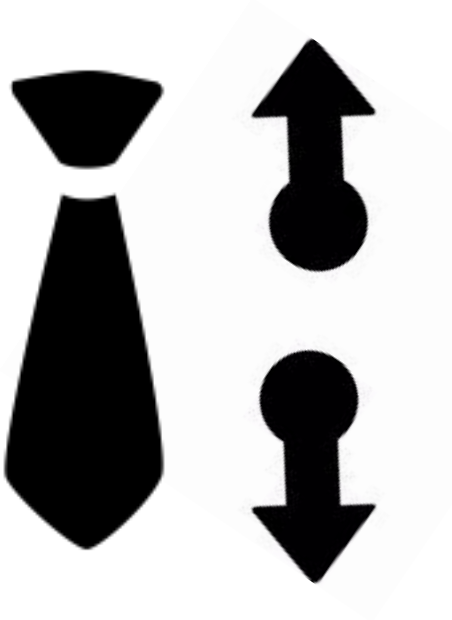
- BFS and IDS
- UCS
- A*
  - Various heuristics



Breaking ties in A* Priority Queue

Order:

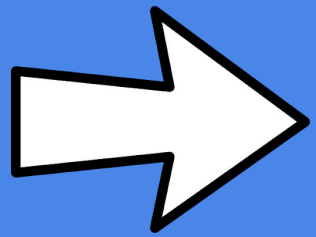
1. Lowest heuristic
2. State that is closer to the goal (Manhattan distance)
3. N > E > W > S

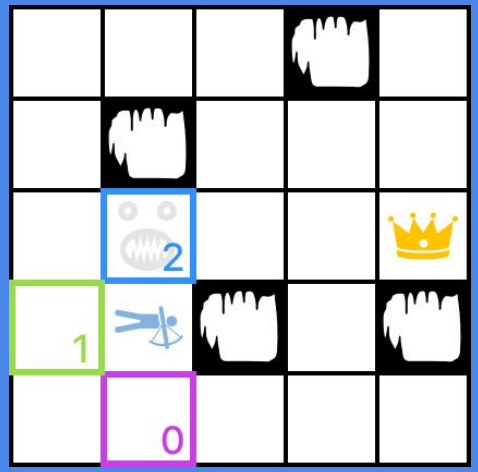


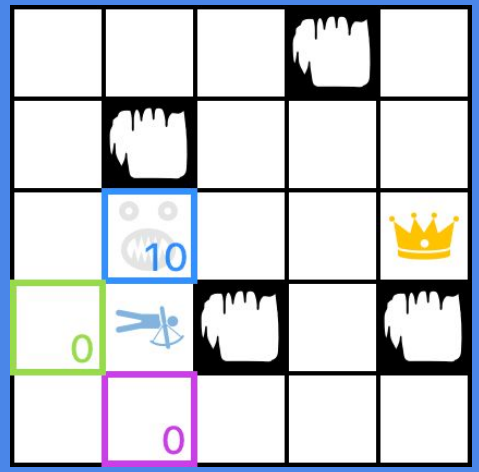
Heuristics

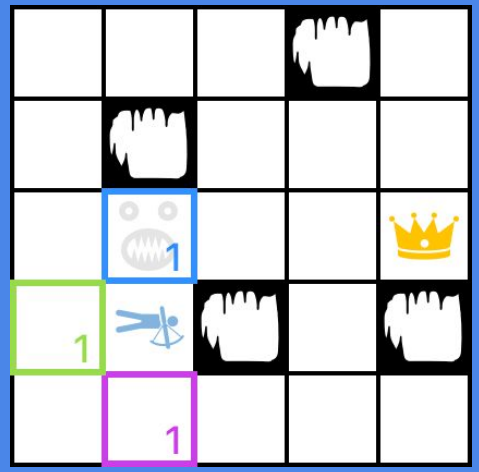
- **heuristic_func_manhattan**
- **heuristic_func_manhattan_with_orientation_overhead**
- **heuristic_func_best_neighbour**
- **heuristic_func_smart_manhattan**
- **heuristic_func_best_neighbour_smart_manhattan**

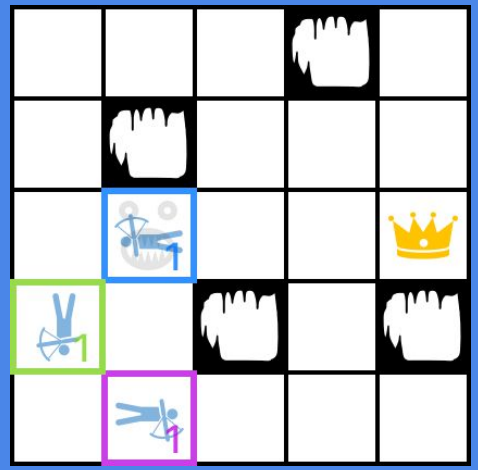
heuristic_func_best_neighbour

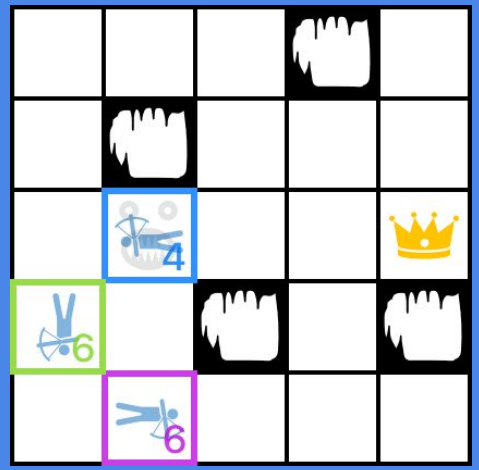


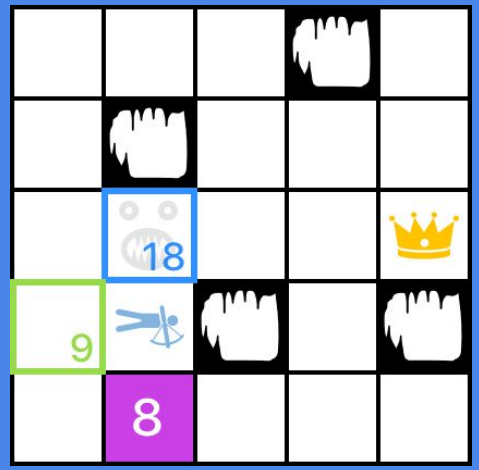

2
1
0


10
0
0

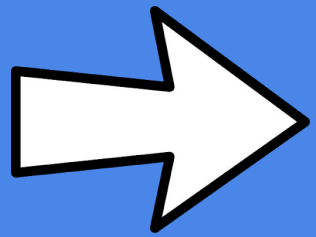

1
1
1

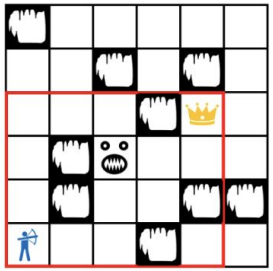

1
1
1
1


4
6
6


18
9
8

heuristic_func_smart_manhattan





(a) *Taking the smallest area containing the agent and the gold*

0	0	0	1	0
0	1	0	0	0
0	1	0	0	1
0	0	0	1	0

(b) *Converting the area in a binary matrix (1 for pit)*

0	0	0	1	0
0	1	0	0	0
0	1	0	0	1
0→	0	0	1	0

(c) *Moving right from initial location*

0	0	0	1	0
0	1	0	0	0
0	1	0	0	1
0	0	0	1	0

(d) *Reachable locations up to this iteration*

0	0	0	1	0
0	1	0	0	0
0	1	0	0	1
0	0	0	1	0

(e) *Moving up from previous row*

0	0	0	1	0
0	1	0	0	0
0→	1	0→	0	1
0	0	0	1	0

(f) *Moving right from reachable locations*

0	0	0	1	0
0	1	0	0	0
0	1	0	0	1
0	0	0	1	0

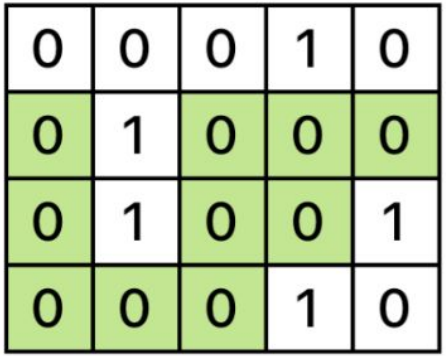
(g) *Reachable locations up to this iteration*

0	0	0	1	0
0	1	0	0	0
0	1	0	0	1
0	0	0	1	0

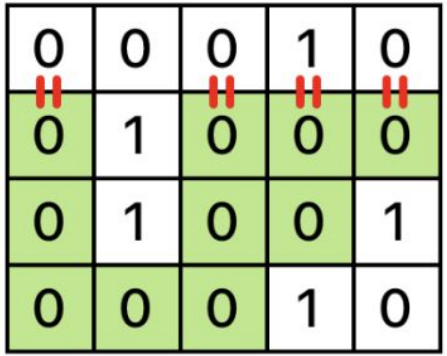
(h) *Moving up from previous row*

0	0	0	1	0
0→	1	0→	0→	0
0	1	0	0	1
0	0	0	1	0

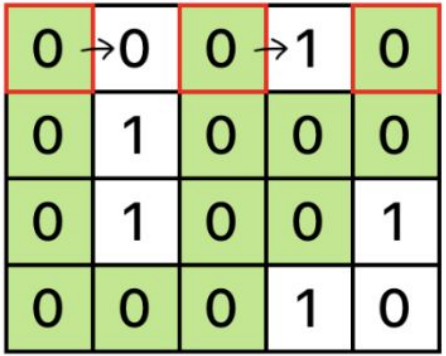
(i) *Moving right from reachable locations*



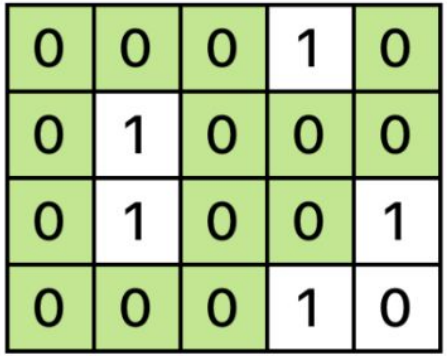
(j) *Reachable locations up to this iteration*



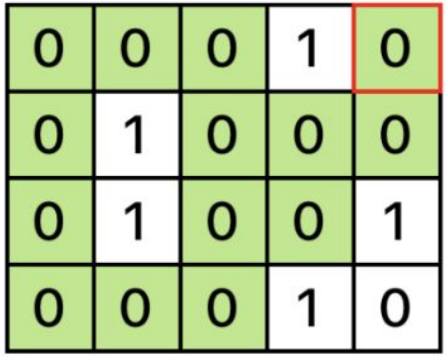
(k) *Moving up from previous row*



(l) *Moving right from reachable locations*

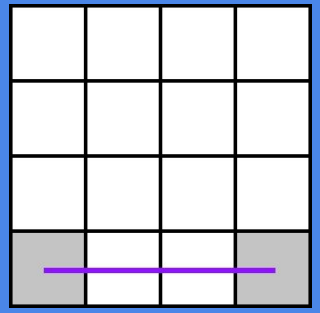


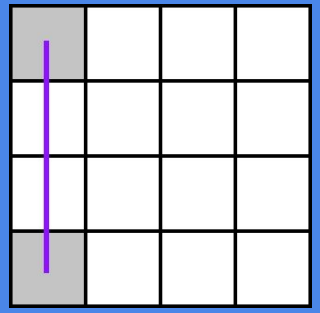
(m) *Reachable locations up to this iteration*



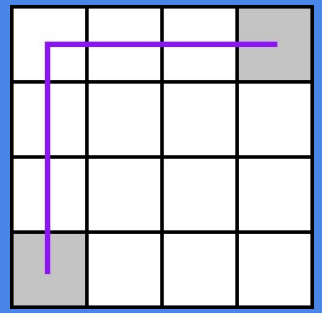
(n) *Resulting matrix*

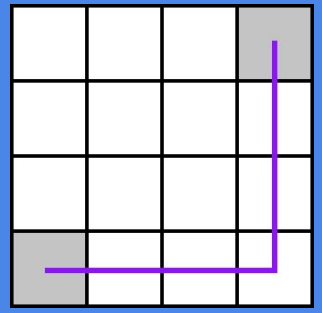
Case 1



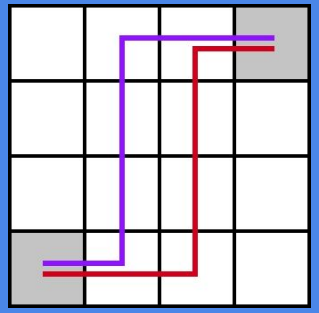


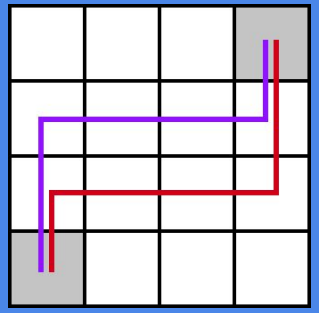
Case 2



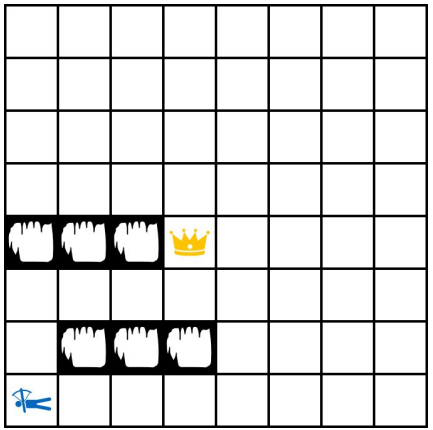


Case 3

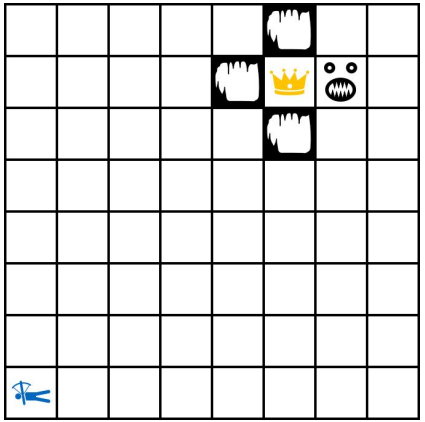




Compare heuristics based on example worlds

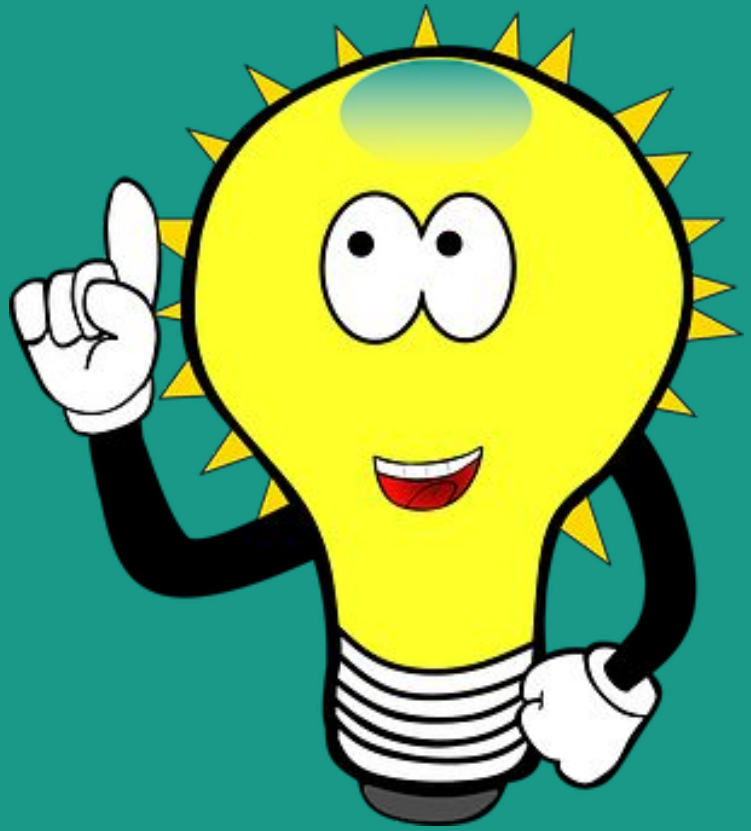


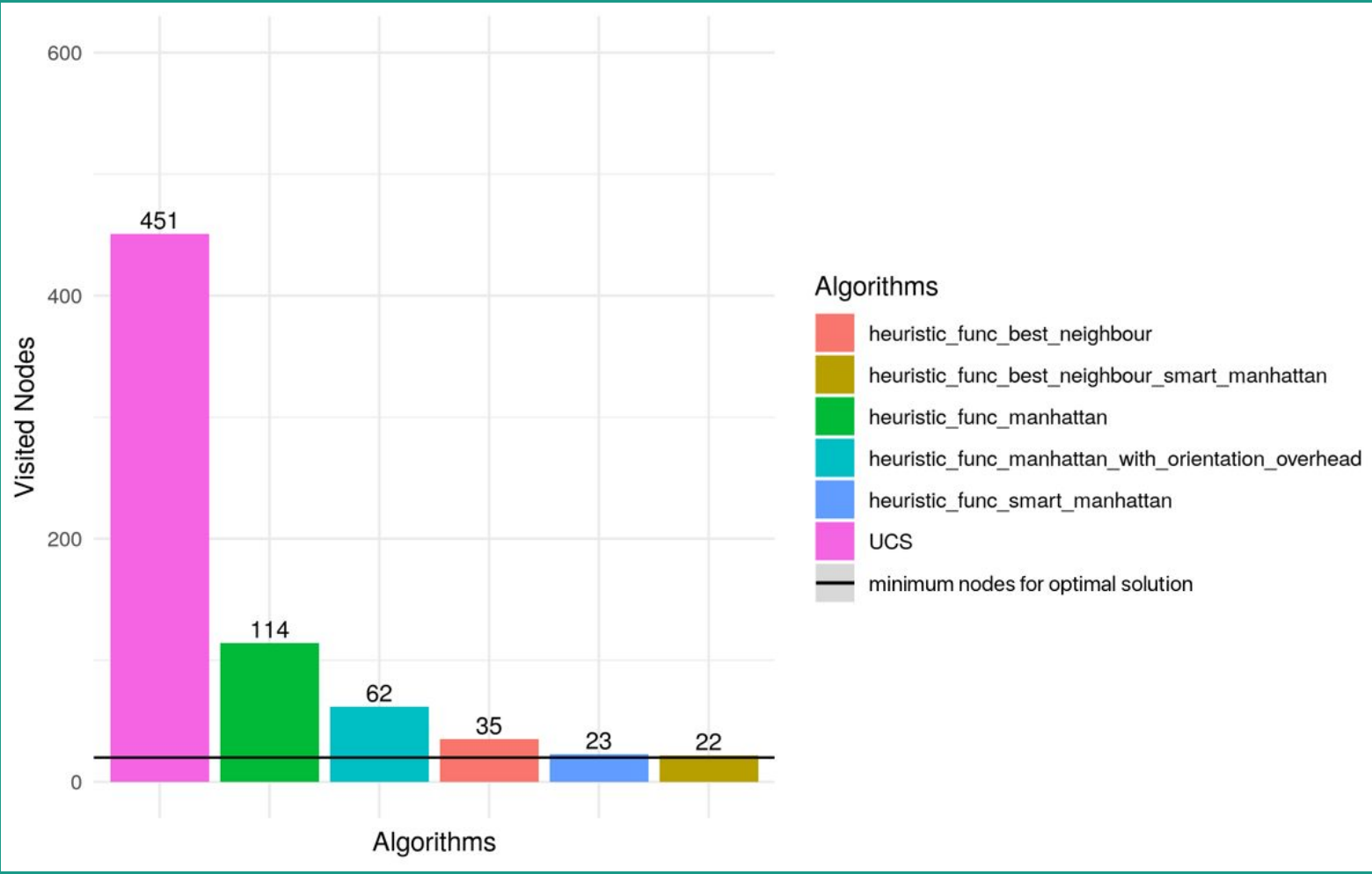
World 4



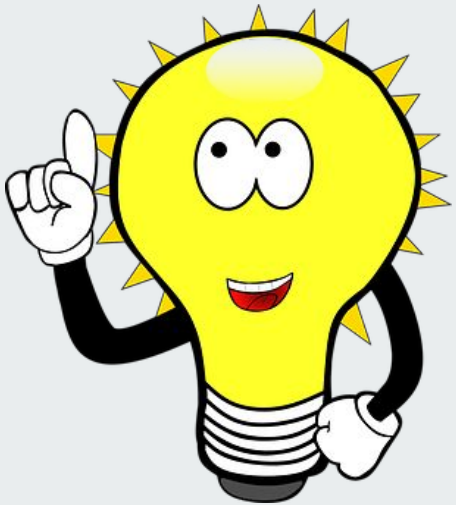
World 6

Results




600
400
200
0
Visited Nodes
451
114
62
35
23
22
Algorithms
Algorithms
heuristic_func_best_neighbour
heuristic_func_best_neighbour_smart_manhattan
heuristic_func_manhattan
heuristic_func_manhattan_with_orientation_overhead
heuristic_func_smart_manhattan
UCS
minimum nodes for optimal solution

Conclusions



- BFS and IDS are not efficient and don’t guarantee optimal solution.
- UCS is optimal, but expensive -> bruteforce.
- A* is optimal and its performance depends on quality of the heuristic function (admissible and consistent)
- The more precise the heuristic function, the better the performance of A*.

Thank you