Informed_Search

May 11, 2024

1 Artificial Intelligence for Robotics 04

1.1 General Information:

Please do not add or delete any cells. Answers belong into the corresponding cells (below the question). If a function is given (either as a signature or a full function), you should not change the name, arguments or return value of the function.

If you encounter empty cells underneath the answer that can not be edited, please ignore them, they are for testing purposes.

When editing an assignment there can be the case that there are variables in the kernel. To make sure your assignment works, please restart the kernel and run all cells before submitting (e.g. via Kernel -> Restart & Run All).

Code cells where you are supposed to give your answer often include the line `raise NotImplementedError`. This makes it easier to automatically grade answers. If you edit the cell please outcomment or delete this line.

1.2 Submission:

Please submit your notebook via the web interface (in the main view -> Assignments -> Submit). The assignments are **due on Monday at 20:00**.

1.3 Group Work:

Please enter your UID (your username here) and those of your group partners into the next cell. We apply plagiarism checking, so do not submit others solutions! If an assignment has a copied solution, the task will be graded with 0 points for all people with the same solution.

1.4 Questions about the Assignment:

If you have questions about the assignment please post them in the LEA forum before the deadline. Don't wait until the last day to post questions!

1.4.1 Please add the usernames of all your team members in the manner member1, member2 in next cell (example given below)

member1 = 'example'

member2 = 'example2'

If you are not working in a group, then please add member2 as none2s

[1]:
```
# YOUR CODE HERE
member1 = 'bghamk2s'

member2 = 'mgadal2s'
```

[2]:
```
# Execute this cell to make sure you correctly filled in the usernames of the␣
 ↪team members

def group_name_test():
    for member_id in [member1, member2]:
        assert isinstance(member_id, str), "Please give your member id as a␣
 ↪string."
        assert len(member_id) > 0, "You need to fill in the member id for both␣
 ↪members"
        assert member_id.endswith("2s"), "The member id should end with 2s␣
 ↪(Your JupyterHub username)"

group_name_test()
print("All tests passed!")
```

```
All tests passed!
```

2 Task 1

[18 Point(s)]

3 A* Theory [20 Points]

During the lecture, you have learned about the A* search algorithm and its propoerties. Please answere the following questions short and consice and be as precise as possible.

3.1 Task 1.1

[2 Point(s)]

3.1.1 Completeness [2 Points]

How is the *completeness* of an algorithm defined? Is A* complete?

An algorithm is complete if it terminates with a solution when one exists given enough resources.

If the search space is not infinite and the heuristic is admissible then it will always return a solution and is complete otherwise if graph or branching factor is infinite and heuristic can be for example negative then it is not complete.

3.2 Task 1.2

[2 Point(s)]

3.2.1 Soundness [2 Points]

When is an algorithm be considered *sound*? Is A* sound?

YOUR ANSWER HERE

3.3 Task 1.3

[2 Point(s)]

3.3.1 Termination [2 Points]

When does A* end the search process?

when: 1- it finds the goal 2- fringe is empty 3- if there is any limitation for resources

3.4 Task 1.4

[5 Point(s)]

3.4.1 Optimality [5 Points]

What does it mean when we say an algorithm is *optimal*? What are the criteria that have to be met in order for A* to be *optimal*? Provide an argument or proof on why this is required.

An algorithm is optimal when it is guarantied to return the least-cost path. A* is optimal if heuristic is: 1- admissible: 2- consistent:

3.5 Task 1.5

[5 Point(s)]

3.5.1 Time Complexity [5 Points]

What is the time complexity of A*? Explain, or derive mathematically! Just writing $O(...)$ is not sufficient!

YOUR ANSWER HERE

3.6 Task 1.6

[2 Point(s)]

3.6.1 Consistency of a Heuristic [2 Points]

How is a consistent heuristic defined? Why is it a critical property in the context of the A* search algorithm?

YOUR ANSWER HERE

4 Task 2

[50 Point(s)]

5 Applying Informed Search [60 Points]

Informed methods help us gain information about solving a problem through its current state space. This keeps a program from blundering around blindly guessing. Informed search strategies make use of information about the path of moves we took to get where we are currently by using an *evaluation function*. This evaluation function often makes use of a *heuristic function*, that provides an estimate of the distance of a node to the goal in a relaxed version of the problem.

6 A Real World Problem

Consider a robot that works in a warehouse. The robot can carry packages on top of itself on a set of omnidirectional conveyors. It can carry a maximum of 8 packages in a 3x3 grid, with one spot left open. There is a slot from which the robot can dispense packages onto shelves. The robot must deposit the packages in order, so they must be lined up in a certain order near the slot. However, workers place the packages onto the robot randomly. Therefore, the robot must first sort the packages before depositing them. It can use the free space and the conveyors to shift the packages horizontally and vertically.

6.1 This becomes what is known as the 8 puzzle.

6.2 Task 2.1

[15 Point(s)]

6.3 The 8 Puzzle [10 Points]

An 8 puzzle is a simple game consisting of a 3 x 3 grid (containing 9 squares). One of the squares is empty. The object is to move the squares around into different positions and having the numbers displayed in the "goal state". The squares can obviously only move along the horizontal and vertical axis and only into the **empty tile**. Therefore the problem can also be viewed as the empty tile "trading" positions with one of its neighbouring tiles.

The image below depicts the goal state of an "3 x 3" 8 puzzle.

The complexity of possible moves toward the final solution in a game like this is large. With an uninformed search strategy, finding a solution to this problem can take a lot of time. The problem is now about finding the shortest path to the goal state within reasonable time, which we will do in this assignment using A* and Greedy Search.

6.3.1 Task

- We will implement the 8 puzzle game as the class `Puzzle`
- Define methods for moving a tile around in the puzzle
- Define a mehod for generating all possible moves from one particular configuration.
- You are already provided with a class and function templates below. Please complete the implementation.

[3]:
```
from typing import List, Tuple
from utils import print_puzzle
```

```
class Puzzle:

    def __init__(self, init_state: List[int], puzzle_type: int = 8):
        """
        Initialize a new 8-puzzle board
        :param init_state: Initial configuration of the board
        """
        self.init_state = init_state
        self.goal_state = [i for i in range(1, 9)] + [0]
        self.explored_set = set()
        self.fringe = []
        self.puzzle_type = puzzle_type

    def goal_test(self, state: List[int]):
        """Test if goal state is reached
        :param state: board configuration to check
        :return: true if the passed configuration is equal to goal configuration
        """
        # YOUR CODE HERE
        return state == self.goal_state

    def is_explored(self, state: List[int]):
        """Check if a particular board configuration has already been explored
        :param state: board configuration to check
        :return: true if a particular configuration has already been explored
        """
        # YOUR CODE HERE
        return state in self.explored_set

    def reset(self):
        # YOUR CODE HERE
        self.explored_set = set()


def move_left(position: int) -> int:
    """Move one position left in 8 puzzle if possible
    :param position: current position of the 0 tile
    :return: new position of the 0 tile after moving to the left
    """
    # YOUR CODE HERE
    return position + 1 if (position % 3) < 2 else position


def move_right(position: int) -> int:
    """Move one position right in 8 puzzle if possible
    :param position: current position of the 0 tile
    :return: new position of the 0 tile after moving to the right
```

```
    """
    # YOUR CODE HERE
    return position - 1 if (position % 3) > 0 else position


def move_up(position: int) -> int:
    """Move one position up in 8 puzzle if possible
    :param position: current position of the 0 tile
    :return: new position of the 0 tile after moving upwards
    """
    # YOUR CODE HERE
    return position + 3 if (position // 3) < 2 else position


def move_down(position: int):
    """Move one position down in 8 puzzle if possible
    :param position: current position of the 0 tile.
    :return: new position of the 0 tile after moving downwards
    """
    # YOUR CODE HERE
    return position - 3 if (position // 3) > 0 else position


def get_possible_moves(state: List[int]) -> List[List[int]]:
    """Check whether a move is possible in left, right, up, down direction and␣
 ↪store it.
    :param state: current configuration of the puzzle as one dimensional list
    :return: list containing the new configurations after applying all possible␣
 ↪moves
    """
    # YOUR CODE HERE
    current_empty_tile = state.index(0)

    def apply_move(move):
        new_empty_tile = move(current_empty_tile)
        if new_empty_tile != current_empty_tile:
            new_state = state.copy()
            new_state[current_empty_tile], new_state[new_empty_tile] =␣
 ↪new_state[new_empty_tile], new_state[
                current_empty_tile]
            return new_state

    move_lists = [apply_move(move_left), apply_move(move_right),␣
 ↪apply_move(move_up), apply_move(move_down)]
    possible_moves = [move for move in move_lists if move is not None]

    return possible_moves
```

```python
# YOUR CODE HERE
class Node:
    def __init__(self, state: List[int], g: int, f: int, parent=None):
        self.state = state
        self.g = g  # Cost to reach this state
        self.f = f  # Estimated total cost. (g + h) for A* and h for best first␣
 ↪search
        self.parent = parent  # Parent node

    def __lt__(self, other):
        return self.f < other.f
```

[ ]:

6.4 Task 2.2

[5 Point(s)]

6.5 Heuristics [5 Points]

In this part, you are tasked to implement two different admissible heuristics for the 8 puzzle: The Hamming distance and the Manhattan Distance.

Hamming Distance
Let A be an alphabet of symbols and C a subset of A^n, the set of worlds of length n over A. Let $u = (u_1, ..., u_n)$ and $v = (v_1, ..., v_n)$. The Hamming distance is deinfed as the number of places in which u and v differ. That is:

$$h(n) = \{i : u_i \neq v_i, i = 1, ..., n\}$$

See also: Encyclopedia of Math

It is clear that this heuristic is admissible since the total number of moves to order the tiles correctly is at least the number of misplaced tiles (each tile not in place must be moved at least once). The cost (number of moves) to the goal (an ordered puzzle) is at least the Hamming distance of the puzzle.

Manhattan Distance
The Manhattan distance between two points $x = (x_1, ..., x_n)$ and $y = (y_1, ..., y_n)$ in n-dimensional space is the sum of the distances in each dimension.

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

See also: Encyclopedia of Machine Learning

The Manhattan distance is an admissible heuristic because every tile will have to be moved at least the number of spots in between itself and its correct position.

6.5.1 Task

- You are provided with the function definitions for the two heuristics below
- Please implement them according to the mathematical definition given above

[4]:
```
from typing import Tuple, List

def hamming_distance(state: List[int]) -> int:
    """Calculate the Hamming distance for a particular configuration
    :param state: current configuration of the puzzle
    :return: the number of misplaced tiles in the given configuration
    """
    # YOUR CODE HERE
    goal_state = [i for i in range(1, 9)] + [0]
    return len([x for (x, y) in zip(state, goal_state) if x != y])


def manhattan_distance(state: List[int]) -> int:
    """Function to calculate the manhattan distance for a
    particular configuration
    :param state: current configuration of the puzzle
    :return: the accumulated manhattan distance between each tile and its goal␣
 ↪position in the given configuration
    """
    # YOUR CODE HERE
    goal_state = [i for i in range(1, 9)] + [0]
    distance = 0
    for i in state:
        goal_i = goal_state.index(i)
        state_i = state.index(i)
        goal_x, goal_y = goal_i // 3, goal_i % 3
        state_x, state_y = state_i // 3, state_i % 3

        distance += abs(goal_x - state_x) + abs(goal_y - state_y)

    return distance
```

[5]:
```
# Run this test cell in order to get an indication of whether your␣
 ↪implementation is working as expected.
assert hamming_distance([0,1,2,3,4,5,8,6,7]) == 9
assert manhattan_distance([1,2,3,4,0,5,6,8,7]) == 8
```

[ ]:
```

```

6.6 Task 2.3

[10 Point(s)]

6.7 A* Search Algorithm [10 Points]

Now you will implement the A* search algorithm. To do that, you can consider the pseudocode from the lecture

```
function A*(problem) returns a solution, or failure
    closed ← an empty set
    fringe ← a list containing Make-Node(Initial-State[problem])
    loop do
        if fringe is empty then return failure
        node ← Remove-Front(fringe)
        if Goal-Test[problem] applied to State(node) succeeds return node
            insert node into closed
        for each node n   Expand(node, problem) do
            if there is a node m   closed   fringe such that
                STATE(m) = STATE(n) and f(m)   f(n)
            then do nothing
            else
                insert n into fringe after the last node m such that f(m)   f(n)
end
```

6.7.1 Task

- You are provided with the function definition for the A* function
- Please implement A* according to the algorithm design discussed in the lectures

In order to get an estimation on whether your solution work as expected, there is a public test available in the test cell. Make sure that all dependent cells are executed first, otherwise some functions or variables might not exist and an error will occur.

[8]:
```python
from typing import Callable, List, Tuple
from heapq import heappop, heappush

def astar_search(board: Puzzle, heuristic: Callable) -> Tuple[List[int], int]:
    """
    :param board: the 8-puzzle to solve
    :param heuristic: the heuristic function to use
    :return: an ordered list with the solution path and the number of total␣
 ↪nodes expanded
    """
    # YOUR CODE HERE
    start_node = Node(board.init_state, 0, heuristic(board.init_state))
    board.fringe = [start_node]

    while board.fringe:
        current_node = heappop(board.fringe)
        current_state = current_node.state

        if board.goal_test(current_state):
```

```
            # Reconstruct the solution path
            solution_path = [current_state]
            while current_node.parent:
                current_node = current_node.parent
                solution_path.append(current_node.state)
            solution_path.reverse()
            return solution_path, len(board.explored_set)

        board.explored_set.add(tuple(current_state))

        for new_state in get_possible_moves(current_state):
            if tuple(new_state) not in board.explored_set:
                new_g = current_node.g + 1
                new_f = new_g + heuristic(new_state)
                new_node = Node(new_state, new_g, new_f, parent=current_node)
                heappush(board.fringe,  new_node)

    return [], len(board.explored_set)
```

[9]:
```
# Run this test cell in order to get an INDICATION of whether your␣
 ↪implementation is working as expected.
import time
t0 = time.time()
p = Puzzle([0,1,2,3,4,5,8,6,7])
path, expanded_nodes = astar_search(p, manhattan_distance)
t1 = time.time()
assert len(path) == 23
# The execution time is limited to 10min, so please keep this in mind when␣
 ↪implementing your solution!
assert t1-t0 <= 600
```

[ ]:
```

```

6.8 Task 2.4

[10 Point(s)]

6.9 Greedy Search [10 Points]

Now we will implement a *greedy search algorithm*. Recall from the lecture, that the evaluation function $f(n)$ is defined as $f(n) = h(n)$, with $h(n)$ being the heuristic function.

6.9.1 Task

- You are provided with the definition of the greedy search function
- Implement a greedy search strategy by completing the function below

In order to get an estimation on whether your solution work as expected, there is a public test

available in the test cell. Make sure that all dependent cells are executed first, otherwise some functions or variables might not exist and an error will occur.

[10]:
```
from typing import List, Tuple, Callable

def greedy_search(board: Puzzle, heuristic: Callable) -> Tuple[List[int], int]:
    """Implementation of the greedy search algorithm.
    :param board: the 8-puzzle to solve
    :param heuristic: the heuristic function to use
    :return: an ordered list with the solution path and the number of total␣
 ↪nodes expanded
    """
    # YOUR CODE HERE
    start_node = Node(board.init_state, 0, heuristic(board.init_state))
    heappush(board.fringe,  start_node)

    while board.fringe:
        current_node = heappop(board.fringe)
        current_state = current_node.state

        if board.goal_test(current_state):
            # Reconstruct the solution path
            solution_path = [current_state]
            while current_node.parent:
                current_node = current_node.parent
                solution_path.append(current_node.state)
            solution_path.reverse()
            return solution_path, len(board.explored_set)

        board.explored_set.add(tuple(current_state))

        for new_state in get_possible_moves(current_state):
            if tuple(new_state) not in board.explored_set:
                new_f = heuristic(new_state)
                new_node = Node(new_state, 0, new_f, parent=current_node)
                heappush(board.fringe, new_node)

    return [], len(board.explored_set)
```

[11]:
```
# Run this test cell in order to get an INDICATION of whether your␣
 ↪implementation is working as expected.
import time
t0 = time.time()
p = Puzzle([0,1,2,3,4,5,8,6,7])
path, expanded_nodes = greedy_search(p, manhattan_distance)
t1 = time.time()
```

```
# The execution time is limited to 10min, so please keep this in mind when␣
 ↪implementing your solution!
assert t1-t0 <= 600
```

[ ]:

6.10 Task 2.5

[5 Point(s)]

6.11 Evaluation A) [5 Points]

Now we want to compare the performance of the different solvers. For that, run every combination of `algorithm` x `heuristic` with the puzzle configurations 1. `[0, 1, 2, 3, 4, 5, 8, 6, 7]` 1. `[8, 7, 6, 5, 1, 4, 2, 0, 3]` 1. `[1, 5, 7, 3, 6, 2, 0, 4, 8]`

for each run, record the `expanded nodes`, `execution time` and the `path cost`.

```
[
  {
    'puzzle': 1,
    'algorithm': 'astar',
    'heuristic': 'manhattan',
    'expanded nodes': 0,
    'execution time': 0.0,
    'path cost': 0
  }
]
```

[20]:
```
evaluation_results = []

# YOUR CODE HERE
def run_evaluation(puzzle_configs, heuristic, algorithm):

    """it solves each configuration with given algorithm and given heuristic
    :param puzzle_configs: dictionary, all the given puzzle configurations.␣
 ↪{index: puzzle}
    :param heuristic: the heuristic function to use.
    :param algorithm: the search algorithm to use.
    :return: a list of recordes. each record is a dictionary.
    """
    records = []
    for i in puzzle_configs.keys():
        t0 = time.time()
        p = Puzzle(puzzle_configs[i])
        path, expanded_nodes = algorithm(p, heuristic)
        t1 = time.time()
```

```
        records.append({
            'puzzle': i,
            'algorithm': algorithm.__name__,
            'heuristic': heuristic.__name__,
            'expanded nodes': expanded_nodes,
            'execution time': t1 - t0,
            'path cost': len(path)})
    return records


def evaluate(puzzle_configs):

    """Runs every combination of algorithm x heuristic with the puzzle␣
 ↪configurations and prints  the results.
    :param puzzle_configs: dictionary, all the given puzzle configurations.␣
 ↪{index: puzzle}
    """
    heuristics = [manhattan_distance, hamming_distance]
    search_algorithms = [astar_search, greedy_search]
    evaluation_results = []

    for heuristic in heuristics:
        for algorithm in search_algorithms:
            eval_records = run_evaluation(puzzle_configs, heuristic, algorithm)
            for record in eval_records:
                evaluation_results.append(record)

    return evaluation_results




puzzle_configs = [[0, 1, 2, 3, 4, 5, 8, 6, 7],
                  [8, 7, 6, 5, 1, 4, 2, 0, 3],
                  [1, 5, 7, 3, 6, 2, 0, 4, 8]]

puzzle_dict = {}

for index, config in enumerate(puzzle_configs):
    puzzle_dict[index] = config

evaluation_results = evaluate(puzzle_dict)
for record in evaluation_results:
    print(record)
```

```
{'puzzle': 0, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1270, 'execution time': 0.027159690856933594, 'path cost': 23}
```

```
{'puzzle': 1, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 9954, 'execution time': 0.24588298797607422, 'path cost': 30}
{'puzzle': 2, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 712, 'execution time': 0.014292716979980469, 'path cost': 23}
{'puzzle': 0, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 192, 'execution time': 0.002690553665161133, 'path cost': 51}
{'puzzle': 1, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 179, 'execution time': 0.002483367919921875, 'path cost': 62}
{'puzzle': 2, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 140, 'execution time': 0.0019690990447998047, 'path cost': 45}
{'puzzle': 0, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 7877, 'execution time': 0.07725334167480469, 'path cost': 23}
{'puzzle': 1, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 79183, 'execution time': 1.5838825702667236, 'path cost': 30}
{'puzzle': 2, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 7550, 'execution time': 0.12472033500671387, 'path cost': 23}
{'puzzle': 0, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 514, 'execution time': 0.004336833953857422, 'path cost': 47}
{'puzzle': 1, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 558, 'execution time': 0.0046999454498291016, 'path cost':
106}
{'puzzle': 2, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 522, 'execution time': 0.0051021575927734375, 'path cost': 85}
```

[ ]:

6.12 Task 2.6

[5 Point(s)]

6.13 Evaluation B) [5 Points]

Lastly, we want to get an estimate on how well the solvers are performing in general. For that, we first have to define a measue of *difficulty* of an initial configuration and whether ther exists a solution at all.

6.13.1 Inversions and Solvability of the 8 and 15 Puzzle

The difficulty of an 8 puzzle configuration can be acessed by counting the number of inversions it contains with respect to the goal configuration. If the square containing the number i appears "before" (reading the squares in the box from left to right and top to bottom) n numbers that are less than i, then call it an inversion of order n, and denote it n_i. Then define

$$N = \sum_{i=1}^{8} n_i = \sum_{i=2}^{8} n_i$$

where the sum only needs to run from 2 to 8, since there are no numbers less than 1 (so n_1 must equal 0). Stated more simply, $N = i(p)$ is the number of *permuted inversions* in the list of numbers.

For the **15 puzzle** there actually exists a proof, which in its principles also holds for the 8 puzzle in modified form:
We additionaly define e to be the row number of the empty square. Then if $N + e$ is even, then the configuration is solvable, otherwise it is not. In other words, if the permutation symbol $(-1)^{i(p)}$ of the list is +1, the configuration is solvable, whereas if the signature is -1, it is not. This can be formally proved using *alternating groups*.

For more details, check out this article by MathWorld

> While odd permutations of the puzzle are impossible to solve (Johnson 1879), all even permutations are solvable (Story 1879).

With those definitions it now becomes possible to evaluate the general performance of the solvers with respect to the complexity of the initial configuration.

1. Compelte the `inversion` function below, such that it calculates the total number of inversions in a puzzle configuration.
2. Use the `inversion` function in order to complete the `is_solvable` function, which should return either `True` or `False` depending on whether a configuration is solvable or not.
3. Generate 100 random **solvable** puzzle configurations
4. Solve each configuration with every combination of `algorithm` and `heuristic` and again record the `expanded nodes`, `execution time` and the `path cost`, as well as the `complexity`, which is the total inversion count of the initial configuration.
5. Plot the metrics over the `complexity` of the puzzle's initial configuration

[38]:
```
import matplotlib.pyplot as plt
import random

def inversion(configuration: List[int]) -> int:
    """
    Finds all inversions in a puzzle configuration and returns their total␣
 ↪number
    :param configuration: the configuration of the 8 puzzle to count its␣
 ↪inversions
    :returns: number of inversions in the puzzle configuration
    """
    # YOUR CODE HERE
    inversion_count = 0
    for i in range(len(configuration)):
        for j in range(i + 1, len(configuration)):
            if configuration[i] != 0 and configuration[j] != 0 and␣
 ↪configuration[i] > configuration[j]:
                inversion_count += 1
    return inversion_count


def is_solvable(configuration: List[int]) -> bool:
    """
    Checks whether a given puzzle configuration is solvable or not
```

```
    :param configuration: the initial configuration of the puzzle to check for␣
 ↪solvability
    :returns: True if the configuration is solvable; False otherwise
    """
    # YOUR CODE HERE
    inversion_count = inversion(configuration)

    return inversion_count % 2 == 0


# YOUR CODE HERE

def run_evaluation(puzzle_configs, heuristic, algorithm):

    records = []
    for i in puzzle_configs.keys():
        t0 = time.time()
        p = Puzzle(puzzle_configs[i])
        path, expanded_nodes = algorithm(p, heuristic)
        t1 = time.time()

        records.append({
            'puzzle': i,
            'algorithm': algorithm.__name__,
            'heuristic': heuristic.__name__,
            'expanded nodes': expanded_nodes,
            'execution time': t1 - t0,
            'path cost': len(path),
            'complexity': inversion(puzzle_configs[i])})
    return records

def generate_puzzle():
    """
    Generate a random solvable puzzle configuration.

    Returns:
    :puzzle: A list representing the puzzle configuration
    """
    puzzle = list(range(9))
    random.shuffle(puzzle)
    while not is_solvable(puzzle):
        random.shuffle(puzzle)
    return puzzle

def generate_solvable_puzzles(total):
    return [generate_puzzle() for i in range(total)]
```

```
puzzle_configs = generate_solvable_puzzles(100)
puzzle_dict = {}

for index, config in enumerate(puzzle_configs):
    puzzle_dict[index] = config

evaluation_results = evaluate(puzzle_dict)
for record in evaluation_results:
    print(record)




# Plotting

# Extracting data for plotting
complexities_astar_man = [result['complexity'] for result in evaluation_results
 ↪if result['algorithm'] == 'astar_search' and result['heuristic'] ==
 ↪'manhattan_distance']
complexities_astar_ham = [result['complexity'] for result in evaluation_results
 ↪if result['algorithm'] == 'astar_search' and result['heuristic'] ==
 ↪'hamming_distance']
complexities_greedy_man = [result['complexity'] for result in
 ↪evaluation_results if result['algorithm'] == 'greedy_search' and
 ↪result['heuristic'] == 'manhattan_distance']
complexities_greedy_ham = [result['complexity'] for result in
 ↪evaluation_results if result['algorithm'] == 'greedy_search' and
 ↪result['heuristic'] == 'hamming_distance']

expanded_nodes_astar_man  = [result['expanded nodes'] for result in
 ↪evaluation_results if result['algorithm'] == 'astar_search' and
 ↪result['heuristic'] == 'manhattan_distance']
expanded_nodes_astar_ham  = [result['expanded nodes'] for result in
 ↪evaluation_results if result['algorithm'] == 'astar_search' and
 ↪result['heuristic'] == 'hamming_distance']
expanded_nodes_greedy_man = [result['expanded nodes'] for result in
 ↪evaluation_results if result['algorithm'] == 'greedy_search' and
 ↪result['heuristic'] == 'manhattan_distance']
expanded_nodes_greedy_ham = [result['expanded nodes'] for result in
 ↪evaluation_results if result['algorithm'] == 'greedy_search' and
 ↪result['heuristic'] == 'hamming_distance']
```

```
execution_times_astar_man  = [result['execution time'] for result in␣
 ↪evaluation_results if result['algorithm'] == 'astar_search' and␣
 ↪result['heuristic'] == 'manhattan_distance']
execution_times_astar_ham  = [result['execution time'] for result in␣
 ↪evaluation_results if result['algorithm'] == 'astar_search' and␣
 ↪result['heuristic'] == 'hamming_distance']
execution_times_greedy_man = [result['execution time'] for result in␣
 ↪evaluation_results if result['algorithm'] == 'greedy_search' and␣
 ↪result['heuristic'] == 'manhattan_distance']
execution_times_greedy_ham = [result['execution time'] for result in␣
 ↪evaluation_results if result['algorithm'] == 'greedy_search' and␣
 ↪result['heuristic'] == 'hamming_distance']

path_costs_times_astar_man  = [result['path cost'] for result in␣
 ↪evaluation_results if result['algorithm'] == 'astar_search' and␣
 ↪result['heuristic'] == 'manhattan_distance']
path_costs_times_astar_ham  = [result['path cost'] for result in␣
 ↪evaluation_results if result['algorithm'] == 'astar_search' and␣
 ↪result['heuristic'] == 'hamming_distance']
path_costs_times_greedy_man = [result['path cost'] for result in␣
 ↪evaluation_results if result['algorithm'] == 'greedy_search' and␣
 ↪result['heuristic'] == 'manhattan_distance']
path_costs_times_greedy_ham = [result['path cost'] for result in␣
 ↪evaluation_results if result['algorithm'] == 'greedy_search' and␣
 ↪result['heuristic'] == 'hamming_distance']

# Plotting
plt.figure(figsize=(15, 20))

plt.subplot(3, 1, 1)
l1 = plt.scatter(complexities_astar_man, expanded_nodes_astar_man, alpha=0.5, s␣
 ↪= 10)
l2 = plt.scatter(complexities_astar_ham, expanded_nodes_astar_ham, alpha=0.5, s␣
 ↪= 10)
l3 = plt.scatter(complexities_greedy_man, expanded_nodes_greedy_man, alpha=0.5,␣
 ↪s = 10)
l4 = plt.scatter(complexities_greedy_ham, expanded_nodes_greedy_ham, alpha=0.5,␣
 ↪s = 10)
plt.legend([l1,l2,l3,l4],["complexities_astar_man", "complexities_astar_ham",␣
 ↪"complexities_greedy_man", "complexities_greedy_ham"])
plt.xlabel('Complexity')
plt.ylabel('Expanded Nodes')

plt.subplot(3, 1, 2)
l1 = plt.scatter(complexities_astar_man, execution_times_astar_man, alpha=0.5,␣
 ↪s = 10)
```

```
l2 = plt.scatter(complexities_astar_ham, execution_times_astar_ham, alpha=0.5,␣
 ↪s = 10)
l3 = plt.scatter(complexities_greedy_man, execution_times_greedy_man, alpha=0.
 ↪5, s = 10)
l4 = plt.scatter(complexities_greedy_ham, execution_times_greedy_ham, alpha=0.
 ↪5, s = 10)
plt.legend([l1,l2,l3,l4],["complexities_astar_man", "complexities_astar_ham",␣
 ↪"complexities_greedy_man", "complexities_greedy_ham"])
plt.xlabel('Complexity')
plt.ylabel('Execution Time')

plt.subplot(3, 1, 3)
l1 = plt.scatter(complexities_astar_man, path_costs_times_astar_man, alpha=0.5,␣
 ↪s = 10)
l2 = plt.scatter(complexities_astar_ham, path_costs_times_astar_ham, alpha=0.5,␣
 ↪s = 10)
l3 = plt.scatter(complexities_greedy_man, path_costs_times_greedy_man, alpha=0.
 ↪5, s = 10)
l4 = plt.scatter(complexities_greedy_ham, path_costs_times_greedy_ham, alpha=0.
 ↪5, s = 10)
plt.legend([l1,l2,l3,l4],["complexities_astar_man", "complexities_astar_ham",␣
 ↪"complexities_greedy_man", "complexities_greedy_ham"])
plt.xlabel('Complexity')
plt.ylabel('Path Cost')

plt.tight_layout()
plt.show()
```

```
{'puzzle': 0, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 587, 'execution time': 0.010325908660888672, 'path cost': 22,
'complexity': 18}
{'puzzle': 1, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1872, 'execution time': 0.02961564064025879, 'path cost': 24,
'complexity': 8}
{'puzzle': 2, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 5043, 'execution time': 0.07601737976074219, 'path cost': 27,
'complexity': 24}
{'puzzle': 3, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1581, 'execution time': 0.024777889251708984, 'path cost': 23,
'complexity': 4}
{'puzzle': 4, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1859, 'execution time': 0.027276992797851562, 'path cost': 24,
'complexity': 10}
{'puzzle': 5, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1816, 'execution time': 0.02700638771057129, 'path cost': 24,
'complexity': 18}
{'puzzle': 6, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
```

'expanded nodes': 653, 'execution time': 0.009658575057983398, 'path cost': 21,
'complexity': 12}
{'puzzle': 7, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 3556, 'execution time': 0.054041385650634766, 'path cost': 26,
'complexity': 12}
{'puzzle': 8, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1949, 'execution time': 0.02967691421508789, 'path cost': 26,
'complexity': 16}
{'puzzle': 9, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 2484, 'execution time': 0.13313603401184082, 'path cost': 27,
'complexity': 12}
{'puzzle': 10, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 913, 'execution time': 0.013367176055908203, 'path cost': 23,
'complexity': 12}
{'puzzle': 11, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1620, 'execution time': 0.02247762680053711, 'path cost': 25,
'complexity': 18}
{'puzzle': 12, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 4241, 'execution time': 0.059326887130737305, 'path cost': 26,
'complexity': 8}
{'puzzle': 13, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 740, 'execution time': 0.011401891708374023, 'path cost': 23,
'complexity': 16}
{'puzzle': 14, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 2749, 'execution time': 0.0380861759185791, 'path cost': 25,
'complexity': 14}
{'puzzle': 15, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1218, 'execution time': 0.017003774642944336, 'path cost': 25,
'complexity': 14}
{'puzzle': 16, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 775, 'execution time': 0.010607481002807617, 'path cost': 22,
'complexity': 16}
{'puzzle': 17, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1869, 'execution time': 0.02645087242126465, 'path cost': 24,
'complexity': 12}
{'puzzle': 18, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 4866, 'execution time': 0.06751632690429688, 'path cost': 27,
'complexity': 14}
{'puzzle': 19, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1596, 'execution time': 0.023438692092895508, 'path cost': 27,
'complexity': 18}
{'puzzle': 20, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1578, 'execution time': 0.02153635025024414, 'path cost': 24,
'complexity': 18}
{'puzzle': 21, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1669, 'execution time': 0.1074528694152832, 'path cost': 26,
'complexity': 16}
{'puzzle': 22, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',

'expanded nodes': 267, 'execution time': 0.003887176513671875, 'path cost': 19,
'complexity': 14}
{'puzzle': 23, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1892, 'execution time': 0.02620720863342285, 'path cost': 23,
'complexity': 12}
{'puzzle': 24, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1044, 'execution time': 0.013586282730102539, 'path cost': 25,
'complexity': 16}
{'puzzle': 25, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 145, 'execution time': 0.0020797252655029297, 'path cost': 21,
'complexity': 14}
{'puzzle': 26, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 4367, 'execution time': 0.05860447883605957, 'path cost': 27,
'complexity': 12}
{'puzzle': 27, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1806, 'execution time': 0.02576279640197754, 'path cost': 24,
'complexity': 6}
{'puzzle': 28, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 4189, 'execution time': 0.05935072898864746, 'path cost': 27,
'complexity': 12}
{'puzzle': 29, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 2070, 'execution time': 0.03490400314331055, 'path cost': 26,
'complexity': 16}
{'puzzle': 30, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 791, 'execution time': 0.011392831802368164, 'path cost': 23,
'complexity': 20}
{'puzzle': 31, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 101, 'execution time': 0.0015444755554199219, 'path cost': 17,
'complexity': 12}
{'puzzle': 32, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 463, 'execution time': 0.005908966064453125, 'path cost': 25,
'complexity': 20}
{'puzzle': 33, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 350, 'execution time': 0.004686117172241211, 'path cost': 20,
'complexity': 14}
{'puzzle': 34, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 728, 'execution time': 0.009428262710571289, 'path cost': 23,
'complexity': 14}
{'puzzle': 35, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 3128, 'execution time': 0.12674593925476074, 'path cost': 27,
'complexity': 16}
{'puzzle': 36, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 599, 'execution time': 0.008975744247436523, 'path cost': 23,
'complexity': 8}
{'puzzle': 37, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 277, 'execution time': 0.003729104995727539, 'path cost': 18,
'complexity': 8}
{'puzzle': 38, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',

'expanded nodes': 1130, 'execution time': 0.014800786972045898, 'path cost': 20,
'complexity': 10}
{'puzzle': 39, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1142, 'execution time': 0.014786958694458008, 'path cost': 24,
'complexity': 20}
{'puzzle': 40, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 44, 'execution time': 0.0009412765502929688, 'path cost': 15,
'complexity': 12}
{'puzzle': 41, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 63, 'execution time': 0.0007886886596679688, 'path cost': 13,
'complexity': 8}
{'puzzle': 42, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 216, 'execution time': 0.002716064453125, 'path cost': 16,
'complexity': 12}
{'puzzle': 43, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 591, 'execution time': 0.007370710372924805, 'path cost': 21,
'complexity': 12}
{'puzzle': 44, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 2975, 'execution time': 0.04099750518798828, 'path cost': 26,
'complexity': 16}
{'puzzle': 45, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 940, 'execution time': 0.013731718063354492, 'path cost': 22,
'complexity': 14}
{'puzzle': 46, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 178, 'execution time': 0.002666950225830078, 'path cost': 15,
'complexity': 10}
{'puzzle': 47, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 2497, 'execution time': 0.03398466110229492, 'path cost': 26,
'complexity': 12}
{'puzzle': 48, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 3191, 'execution time': 0.04528689384460449, 'path cost': 26,
'complexity': 14}
{'puzzle': 49, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 96, 'execution time': 0.0024003982543945312, 'path cost': 17,
'complexity': 12}
{'puzzle': 50, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1690, 'execution time': 0.022457599639892578, 'path cost': 25,
'complexity': 16}
{'puzzle': 51, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 3166, 'execution time': 0.044270992279052734, 'path cost': 27,
'complexity': 14}
{'puzzle': 52, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 5449, 'execution time': 0.17555785179138184, 'path cost': 27,
'complexity': 14}
{'puzzle': 53, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1595, 'execution time': 0.0252993106842041, 'path cost': 23,
'complexity': 14}
{'puzzle': 54, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',

'expanded nodes': 836, 'execution time': 0.011404037475585938, 'path cost': 22,
'complexity': 12}
{'puzzle': 55, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1340, 'execution time': 0.01818990707397461, 'path cost': 25,
'complexity': 14}
{'puzzle': 56, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 37, 'execution time': 0.0008957386016845703, 'path cost': 15,
'complexity': 12}
{'puzzle': 57, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1632, 'execution time': 0.02178645133972168, 'path cost': 25,
'complexity': 16}
{'puzzle': 58, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 122, 'execution time': 0.002086162567138672, 'path cost': 16,
'complexity': 10}
{'puzzle': 59, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 6206, 'execution time': 0.08688139915466309, 'path cost': 27,
'complexity': 14}
{'puzzle': 60, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 2043, 'execution time': 0.030822038650512695, 'path cost': 23,
'complexity': 14}
{'puzzle': 61, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 2093, 'execution time': 0.028253793716430664, 'path cost': 25,
'complexity': 14}
{'puzzle': 62, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1358, 'execution time': 0.018152475357055664, 'path cost': 25,
'complexity': 16}
{'puzzle': 63, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 828, 'execution time': 0.010834217071533203, 'path cost': 24,
'complexity': 18}
{'puzzle': 64, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 447, 'execution time': 0.006036520004272461, 'path cost': 20,
'complexity': 12}
{'puzzle': 65, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 416, 'execution time': 0.005410671234130859, 'path cost': 19,
'complexity': 16}
{'puzzle': 66, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 327, 'execution time': 0.0048329830169677734, 'path cost': 19,
'complexity': 8}
{'puzzle': 67, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 2783, 'execution time': 0.04012632369995117, 'path cost': 27,
'complexity': 16}
{'puzzle': 68, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 510, 'execution time': 0.00771641731262207, 'path cost': 19,
'complexity': 10}
{'puzzle': 69, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1798, 'execution time': 0.025032520294189453, 'path cost': 24,
'complexity': 12}
{'puzzle': 70, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',

'expanded nodes': 2398, 'execution time': 0.03245115280151367, 'path cost': 27,
'complexity': 24}
{'puzzle': 71, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1308, 'execution time': 0.017880678176879883, 'path cost': 24,
'complexity': 16}
{'puzzle': 72, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1260, 'execution time': 0.016402244567871094, 'path cost': 24,
'complexity': 20}
{'puzzle': 73, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 2460, 'execution time': 0.11731123924255371, 'path cost': 26,
'complexity': 18}
{'puzzle': 74, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 266, 'execution time': 0.004289388656616211, 'path cost': 18,
'complexity': 12}
{'puzzle': 75, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1161, 'execution time': 0.014748573303222656, 'path cost': 23,
'complexity': 14}
{'puzzle': 76, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 9399, 'execution time': 0.13419342041015625, 'path cost': 28,
'complexity': 14}
{'puzzle': 77, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 6464, 'execution time': 0.09354710578918457, 'path cost': 29,
'complexity': 20}
{'puzzle': 78, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 838, 'execution time': 0.01336526870727539, 'path cost': 25,
'complexity': 20}
{'puzzle': 79, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 311, 'execution time': 0.004324674606323242, 'path cost': 19,
'complexity': 12}
{'puzzle': 80, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 12739, 'execution time': 0.2648279666900635, 'path cost': 29,
'complexity': 8}
{'puzzle': 81, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 557, 'execution time': 0.013150930404663086, 'path cost': 21,
'complexity': 14}
{'puzzle': 82, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 76, 'execution time': 0.001142263412475586, 'path cost': 17,
'complexity': 16}
{'puzzle': 83, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 67, 'execution time': 0.0008957386016845703, 'path cost': 15,
'complexity': 10}
{'puzzle': 84, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 2011, 'execution time': 0.025960683822631836, 'path cost': 25,
'complexity': 18}
{'puzzle': 85, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 644, 'execution time': 0.008568763732910156, 'path cost': 21,
'complexity': 14}
{'puzzle': 86, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',

'expanded nodes': 1227, 'execution time': 0.01615762710571289, 'path cost': 26,
'complexity': 20}
{'puzzle': 87, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 516, 'execution time': 0.008325338363647461, 'path cost': 23,
'complexity': 16}
{'puzzle': 88, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 735, 'execution time': 0.010186910629272461, 'path cost': 23,
'complexity': 20}
{'puzzle': 89, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 3662, 'execution time': 0.14939212799072266, 'path cost': 26,
'complexity': 20}
{'puzzle': 90, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 3441, 'execution time': 0.0472569465637207, 'path cost': 27,
'complexity': 22}
{'puzzle': 91, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1566, 'execution time': 0.022740840911865234, 'path cost': 24,
'complexity': 18}
{'puzzle': 92, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 6093, 'execution time': 0.0832982063293457, 'path cost': 28,
'complexity': 22}
{'puzzle': 93, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 6375, 'execution time': 0.1791553497314453, 'path cost': 30,
'complexity': 20}
{'puzzle': 94, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1814, 'execution time': 0.028297901153564453, 'path cost': 25,
'complexity': 12}
{'puzzle': 95, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 46, 'execution time': 0.0011157989501953125, 'path cost': 14,
'complexity': 10}
{'puzzle': 96, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 4314, 'execution time': 0.06310868263244629, 'path cost': 26,
'complexity': 8}
{'puzzle': 97, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 340, 'execution time': 0.009217500686645508, 'path cost': 21,
'complexity': 14}
{'puzzle': 98, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 1736, 'execution time': 0.02480149269104004, 'path cost': 26,
'complexity': 14}
{'puzzle': 99, 'algorithm': 'astar_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 943, 'execution time': 0.013737916946411133, 'path cost': 22,
'complexity': 12}
{'puzzle': 0, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 406, 'execution time': 0.00523066520690918, 'path cost': 116,
'complexity': 18}
{'puzzle': 1, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 241, 'execution time': 0.0032224655151367188, 'path cost': 68,
'complexity': 8}
{'puzzle': 2, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',

'expanded nodes': 351, 'execution time': 0.004614353179931641, 'path cost': 75,
'complexity': 24}
{'puzzle': 3, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 46, 'execution time': 0.0006613731384277344, 'path cost': 31,
'complexity': 4}
{'puzzle': 4, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 480, 'execution time': 0.007195711135864258, 'path cost': 90,
'complexity': 10}
{'puzzle': 5, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 138, 'execution time': 0.0018761157989501953, 'path cost': 48,
'complexity': 18}
{'puzzle': 6, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 728, 'execution time': 0.009465456008911133, 'path cost': 91,
'complexity': 12}
{'puzzle': 7, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 618, 'execution time': 0.008053779602050781, 'path cost': 116,
'complexity': 12}
{'puzzle': 8, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 101, 'execution time': 0.0013949871063232422, 'path cost': 38,
'complexity': 16}
{'puzzle': 9, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 363, 'execution time': 0.004697322845458984, 'path cost': 75,
'complexity': 12}
{'puzzle': 10, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 328, 'execution time': 0.006321430206298828, 'path cost': 95,
'complexity': 12}
{'puzzle': 11, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 81, 'execution time': 0.0010776519775390625, 'path cost': 33,
'complexity': 18}
{'puzzle': 12, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 184, 'execution time': 0.002499103546142578, 'path cost': 72,
'complexity': 8}
{'puzzle': 13, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 154, 'execution time': 0.0021677017211914062, 'path cost': 51,
'complexity': 16}
{'puzzle': 14, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 873, 'execution time': 0.013523101806640625, 'path cost': 85,
'complexity': 14}
{'puzzle': 15, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 163, 'execution time': 0.002698183059692383, 'path cost': 71,
'complexity': 14}
{'puzzle': 16, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 497, 'execution time': 0.006471395492553711, 'path cost': 96,
'complexity': 16}
{'puzzle': 17, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 331, 'execution time': 0.004485607147216797, 'path cost': 66,
'complexity': 12}
{'puzzle': 18, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',

'expanded nodes': 290, 'execution time': 0.003833770751953125, 'path cost': 93,
'complexity': 14}
{'puzzle': 19, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 412, 'execution time': 0.0055272579193115234, 'path cost': 89,
'complexity': 18}
{'puzzle': 20, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 526, 'execution time': 0.007555961608886719, 'path cost': 122,
'complexity': 18}
{'puzzle': 21, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 126, 'execution time': 0.0016825199127197266, 'path cost': 56,
'complexity': 16}
{'puzzle': 22, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 285, 'execution time': 0.0036339759826660156, 'path cost': 73,
'complexity': 14}
{'puzzle': 23, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 576, 'execution time': 0.007589101791381836, 'path cost': 53,
'complexity': 12}
{'puzzle': 24, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 149, 'execution time': 0.001972675323486328, 'path cost': 55,
'complexity': 16}
{'puzzle': 25, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 335, 'execution time': 0.004234790802001953, 'path cost': 73,
'complexity': 14}
{'puzzle': 26, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 291, 'execution time': 0.003796815872192383, 'path cost': 67,
'complexity': 12}
{'puzzle': 27, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 411, 'execution time': 0.00561833381652832, 'path cost': 82,
'complexity': 6}
{'puzzle': 28, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 563, 'execution time': 0.007369518280029297, 'path cost': 91,
'complexity': 12}
{'puzzle': 29, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 110, 'execution time': 0.0015239715576171875, 'path cost': 36,
'complexity': 16}
{'puzzle': 30, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 216, 'execution time': 0.002835988998413086, 'path cost': 47,
'complexity': 20}
{'puzzle': 31, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 151, 'execution time': 0.0019559860229492188, 'path cost': 69,
'complexity': 12}
{'puzzle': 32, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 607, 'execution time': 0.007956981658935547, 'path cost': 137,
'complexity': 20}
{'puzzle': 33, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 896, 'execution time': 0.011632204055786133, 'path cost': 124,
'complexity': 14}
{'puzzle': 34, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',

'expanded nodes': 30, 'execution time': 0.0006918907165527344, 'path cost': 23,
'complexity': 14}
{'puzzle': 35, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 415, 'execution time': 0.005584478378295984, 'path cost': 75,
'complexity': 16}
{'puzzle': 36, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 44, 'execution time': 0.0006821155548095703, 'path cost': 31,
'complexity': 8}
{'puzzle': 37, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 68, 'execution time': 0.0008666515350341797, 'path cost': 40,
'complexity': 8}
{'puzzle': 38, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 143, 'execution time': 0.0018036365509033203, 'path cost': 32,
'complexity': 10}
{'puzzle': 39, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 383, 'execution time': 0.00504302978515625, 'path cost': 90,
'complexity': 20}
{'puzzle': 40, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 154, 'execution time': 0.0020601749420166016, 'path cost': 59,
'complexity': 12}
{'puzzle': 41, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 374, 'execution time': 0.00473785400390625, 'path cost': 57,
'complexity': 8}
{'puzzle': 42, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 95, 'execution time': 0.0012695789337158203, 'path cost': 22,
'complexity': 12}
{'puzzle': 43, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 329, 'execution time': 0.004167079925537109, 'path cost': 89,
'complexity': 12}
{'puzzle': 44, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 271, 'execution time': 0.003483295440673828, 'path cost': 56,
'complexity': 16}
{'puzzle': 45, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 63, 'execution time': 0.0008716583251953125, 'path cost': 40,
'complexity': 14}
{'puzzle': 46, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 476, 'execution time': 0.006102561950683594, 'path cost': 89,
'complexity': 10}
{'puzzle': 47, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 295, 'execution time': 0.003805398941040039, 'path cost': 76,
'complexity': 12}
{'puzzle': 48, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 137, 'execution time': 0.0018200874328613281, 'path cost': 48,
'complexity': 14}
{'puzzle': 49, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 84, 'execution time': 0.0011043548583984375, 'path cost': 37,
'complexity': 12}
{'puzzle': 50, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',

'expanded nodes': 221, 'execution time': 0.0028276443481445312, 'path cost': 51,
'complexity': 16}
{'puzzle': 51, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 461, 'execution time': 0.006125450134277344, 'path cost': 109,
'complexity': 14}
{'puzzle': 52, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 624, 'execution time': 0.008077621459960938, 'path cost': 85,
'complexity': 14}
{'puzzle': 53, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 376, 'execution time': 0.0051364898681640625, 'path cost': 99,
'complexity': 14}
{'puzzle': 54, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 284, 'execution time': 0.0038230419158935547, 'path cost': 64,
'complexity': 12}
{'puzzle': 55, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 79, 'execution time': 0.0010614395141601562, 'path cost': 29,
'complexity': 14}
{'puzzle': 56, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 43, 'execution time': 0.0005640983581542969, 'path cost': 27,
'complexity': 12}
{'puzzle': 57, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 631, 'execution time': 0.008209705352783203, 'path cost': 107,
'complexity': 16}
{'puzzle': 58, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 31, 'execution time': 0.0005681514739990234, 'path cost': 22,
'complexity': 10}
{'puzzle': 59, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 211, 'execution time': 0.0027170181274414062, 'path cost': 61,
'complexity': 14}
{'puzzle': 60, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 430, 'execution time': 0.005437850952148437 5, 'path cost':
101, 'complexity': 14}
{'puzzle': 61, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 145, 'execution time': 0.0018448829650878906, 'path cost': 43,
'complexity': 14}
{'puzzle': 62, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 181, 'execution time': 0.0022749900817871094, 'path cost': 39,
'complexity': 16}
{'puzzle': 63, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 401, 'execution time': 0.005087137222290039, 'path cost': 90,
'complexity': 18}
{'puzzle': 64, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 357, 'execution time': 0.004889726638793945, 'path cost': 74,
'complexity': 12}
{'puzzle': 65, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 82, 'execution time': 0.0011413097381591797, 'path cost': 43,
'complexity': 16}
{'puzzle': 66, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',

'expanded nodes': 71, 'execution time': 0.0009014606475830078, 'path cost': 31,
'complexity': 8}
{'puzzle': 67, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 539, 'execution time': 0.007335186004638672, 'path cost': 91,
'complexity': 16}
{'puzzle': 68, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 505, 'execution time': 0.006964921951293945, 'path cost': 89,
'complexity': 10}
{'puzzle': 69, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 350, 'execution time': 0.00480961799621582, 'path cost': 72,
'complexity': 12}
{'puzzle': 70, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 329, 'execution time': 0.004238128662109375, 'path cost': 111,
'complexity': 24}
{'puzzle': 71, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 257, 'execution time': 0.003470182418823242, 'path cost': 62,
'complexity': 16}
{'puzzle': 72, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 278, 'execution time': 0.0037386417388916016, 'path cost': 36,
'complexity': 20}
{'puzzle': 73, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 427, 'execution time': 0.005532503128051758, 'path cost': 56,
'complexity': 18}
{'puzzle': 74, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 435, 'execution time': 0.005558490753173828, 'path cost': 66,
'complexity': 12}
{'puzzle': 75, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 185, 'execution time': 0.002408266067504883, 'path cost': 65,
'complexity': 14}
{'puzzle': 76, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 709, 'execution time': 0.009398460388183594, 'path cost': 64,
'complexity': 14}
{'puzzle': 77, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 114, 'execution time': 0.0016112327575683594, 'path cost': 47,
'complexity': 20}
{'puzzle': 78, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 137, 'execution time': 0.0017633438110351562, 'path cost': 57,
'complexity': 20}
{'puzzle': 79, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 176, 'execution time': 0.002391338348388672, 'path cost': 49,
'complexity': 12}
{'puzzle': 80, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 575, 'execution time': 0.00841665267944336, 'path cost': 67,
'complexity': 8}
{'puzzle': 81, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 438, 'execution time': 0.005813121795654297, 'path cost': 105,
'complexity': 14}
{'puzzle': 82, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',

'expanded nodes': 115, 'execution time': 0.001592397689819336, 'path cost': 47,
'complexity': 16}
{'puzzle': 83, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 104, 'execution time': 0.0013611316680908203, 'path cost': 29,
'complexity': 10}
{'puzzle': 84, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 336, 'execution time': 0.0042798519134521484, 'path cost': 87,
'complexity': 18}
{'puzzle': 85, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 172, 'execution time': 0.002286195755004883, 'path cost': 61,
'complexity': 14}
{'puzzle': 86, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 232, 'execution time': 0.003359079360961914, 'path cost': 88,
'complexity': 20}
{'puzzle': 87, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 82, 'execution time': 0.0011324882507324219, 'path cost': 37,
'complexity': 16}
{'puzzle': 88, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 306, 'execution time': 0.0044782161712646484, 'path cost': 81,
'complexity': 20}
{'puzzle': 89, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 160, 'execution time': 0.0021440982818603516, 'path cost': 54,
'complexity': 20}
{'puzzle': 90, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 266, 'execution time': 0.0034677982330322266, 'path cost': 37,
'complexity': 22}
{'puzzle': 91, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 280, 'execution time': 0.0036385059356689453, 'path cost': 44,
'complexity': 18}
{'puzzle': 92, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 244, 'execution time': 0.0030641555786132812, 'path cost': 48,
'complexity': 22}
{'puzzle': 93, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 491, 'execution time': 0.00634312629699707, 'path cost': 144,
'complexity': 20}
{'puzzle': 94, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 339, 'execution time': 0.00447392463684082, 'path cost': 87,
'complexity': 12}
{'puzzle': 95, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 144, 'execution time': 0.0018923282623291016, 'path cost': 40,
'complexity': 10}
{'puzzle': 96, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 518, 'execution time': 0.006687164306640625, 'path cost': 104,
'complexity': 8}
{'puzzle': 97, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 353, 'execution time': 0.004631757736206055, 'path cost': 87,
'complexity': 14}
{'puzzle': 98, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',

'expanded nodes': 274, 'execution time': 0.0035903453826904297, 'path cost': 96,
'complexity': 14}
{'puzzle': 99, 'algorithm': 'greedy_search', 'heuristic': 'manhattan_distance',
'expanded nodes': 241, 'execution time': 0.003185272216796875, 'path cost': 36,
'complexity': 12}
{'puzzle': 0, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 4963, 'execution time': 0.04238462448120117, 'path cost': 22,
'complexity': 18}
{'puzzle': 1, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 10638, 'execution time': 0.1887664794921875, 'path cost': 24,
'complexity': 8}
{'puzzle': 2, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 28434, 'execution time': 0.5073256492614746, 'path cost': 27,
'complexity': 24}
{'puzzle': 3, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 8329, 'execution time': 0.0906364917755127, 'path cost': 23,
'complexity': 4}
{'puzzle': 4, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 11635, 'execution time': 0.20616579055786133, 'path cost': 24,
'complexity': 10}
{'puzzle': 5, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 10690, 'execution time': 0.10313153266906738, 'path cost': 24,
'complexity': 18}
{'puzzle': 6, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 2539, 'execution time': 0.026595592498779297, 'path cost': 21,
'complexity': 12}
{'puzzle': 7, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 24856, 'execution time': 0.44478774070739746, 'path cost': 26,
'complexity': 12}
{'puzzle': 8, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 26111, 'execution time': 0.37719058990478516, 'path cost': 26,
'complexity': 16}
{'puzzle': 9, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 37484, 'execution time': 0.6210942268371582, 'path cost': 27,
'complexity': 12}
{'puzzle': 10, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 6756, 'execution time': 0.08174872398376465, 'path cost': 23,
'complexity': 12}
{'puzzle': 11, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 13558, 'execution time': 0.22249412536621094, 'path cost': 25,
'complexity': 18}
{'puzzle': 12, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 28164, 'execution time': 0.38297557830810547, 'path cost': 26,
'complexity': 8}
{'puzzle': 13, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 7507, 'execution time': 0.16963434219360352, 'path cost': 23,
'complexity': 16}
{'puzzle': 14, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',

'expanded nodes': 13441, 'execution time': 0.22347187995910645, 'path cost': 25,
'complexity': 14}
{'puzzle': 15, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 13112, 'execution time': 0.12352538108825684, 'path cost': 25,
'complexity': 14}
{'puzzle': 16, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 4753, 'execution time': 0.13361334800720215, 'path cost': 22,
'complexity': 16}
{'puzzle': 17, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 5284, 'execution time': 0.05234122276306152, 'path cost': 22,
'complexity': 12}
{'puzzle': 18, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 28562, 'execution time': 0.5141377449035645, 'path cost': 27,
'complexity': 14}
{'puzzle': 19, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 27508, 'execution time': 0.39087772369384766, 'path cost': 27,
'complexity': 18}
{'puzzle': 20, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 10937, 'execution time': 0.2236320972442627, 'path cost': 24,
'complexity': 18}
{'puzzle': 21, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 26487, 'execution time': 0.3734159469604492, 'path cost': 26,
'complexity': 16}
{'puzzle': 22, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1199, 'execution time': 0.024762868881225586, 'path cost': 19,
'complexity': 14}
{'puzzle': 23, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 9244, 'execution time': 0.08355307579040527, 'path cost': 23,
'complexity': 12}
{'puzzle': 24, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 18487, 'execution time': 0.4080216884613037, 'path cost': 25,
'complexity': 16}
{'puzzle': 25, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 2721, 'execution time': 0.0356900691986084, 'path cost': 21,
'complexity': 14}
{'puzzle': 26, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 28511, 'execution time': 0.41001224517822266, 'path cost': 27,
'complexity': 12}
{'puzzle': 27, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 11001, 'execution time': 0.21765875816345215, 'path cost': 24,
'complexity': 6}
{'puzzle': 28, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 41293, 'execution time': 0.6760637760162354, 'path cost': 27,
'complexity': 12}
{'puzzle': 29, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 23340, 'execution time': 0.3945772647857666, 'path cost': 26,
'complexity': 16}
{'puzzle': 30, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',

'expanded nodes': 6089, 'execution time': 0.0648810863494873, 'path cost': 23,
'complexity': 20}
{'puzzle': 31, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 390, 'execution time': 0.005625247955322266, 'path cost': 17,
'complexity': 12}
{'puzzle': 32, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 12788, 'execution time': 0.20714330673217773, 'path cost': 25,
'complexity': 20}
{'puzzle': 33, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 2421, 'execution time': 0.026439666748046875, 'path cost': 20,
'complexity': 14}
{'puzzle': 34, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 7527, 'execution time': 0.06892061233520508, 'path cost': 23,
'complexity': 14}
{'puzzle': 35, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 29532, 'execution time': 0.4939439296722412, 'path cost': 27,
'complexity': 16}
{'puzzle': 36, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 8084, 'execution time': 0.09021592140197754, 'path cost': 23,
'complexity': 8}
{'puzzle': 37, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1014, 'execution time': 0.01201319694519043, 'path cost': 18,
'complexity': 8}
{'puzzle': 38, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 2831, 'execution time': 0.11343646049499512, 'path cost': 20,
'complexity': 10}
{'puzzle': 39, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 11671, 'execution time': 0.10657286643981934, 'path cost': 24,
'complexity': 20}
{'puzzle': 40, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 155, 'execution time': 0.006446123123168945, 'path cost': 15,
'complexity': 12}
{'puzzle': 41, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 83, 'execution time': 0.0006382465362548828, 'path cost': 13,
'complexity': 8}
{'puzzle': 42, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 373, 'execution time': 0.002997875213623047, 'path cost': 16,
'complexity': 12}
{'puzzle': 43, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 2543, 'execution time': 0.021674633026123047, 'path cost': 21,
'complexity': 12}
{'puzzle': 44, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 24411, 'execution time': 0.44265007972717285, 'path cost': 26,
'complexity': 16}
{'puzzle': 45, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 4903, 'execution time': 0.055708885192871094, 'path cost': 22,
'complexity': 14}
{'puzzle': 46, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',

'expanded nodes': 255, 'execution time': 0.004042625427246094, 'path cost': 15,
'complexity': 10}
{'puzzle': 47, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 26883, 'execution time': 0.35653233528137207, 'path cost': 26,
'complexity': 12}
{'puzzle': 48, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 23409, 'execution time': 0.43747782707214355, 'path cost': 26,
'complexity': 14}
{'puzzle': 49, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 481, 'execution time': 0.016369104385375977, 'path cost': 17,
'complexity': 12}
{'puzzle': 50, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 17774, 'execution time': 0.27893567085266113, 'path cost': 25,
'complexity': 16}
{'puzzle': 51, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 38364, 'execution time': 0.652125358581543, 'path cost': 27,
'complexity': 14}
{'puzzle': 52, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 28271, 'execution time': 0.41325974464416504, 'path cost': 27,
'complexity': 14}
{'puzzle': 53, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 6370, 'execution time': 0.0726780891418457, 'path cost': 23,
'complexity': 14}
{'puzzle': 54, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 4816, 'execution time': 0.04304957389831543, 'path cost': 22,
'complexity': 12}
{'puzzle': 55, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 17600, 'execution time': 0.25999903678894043, 'path cost': 25,
'complexity': 14}
{'puzzle': 56, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 221, 'execution time': 0.010733366012573242, 'path cost': 15,
'complexity': 12}
{'puzzle': 57, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 20521, 'execution time': 0.2885255813598633, 'path cost': 25,
'complexity': 16}
{'puzzle': 58, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 473, 'execution time': 0.015256166458129883, 'path cost': 16,
'complexity': 10}
{'puzzle': 59, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 30253, 'execution time': 0.49256086349487305, 'path cost': 27,
'complexity': 14}
{'puzzle': 60, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 10126, 'execution time': 0.20427393913269043, 'path cost': 23,
'complexity': 14}
{'puzzle': 61, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 15033, 'execution time': 0.2526557445526123, 'path cost': 25,
'complexity': 14}
{'puzzle': 62, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',

'expanded nodes': 13512, 'execution time': 0.15169215202331543, 'path cost': 25,
'complexity': 16}
{'puzzle': 63, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 11041, 'execution time': 0.19979214668273926, 'path cost': 24,
'complexity': 18}
{'puzzle': 64, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1667, 'execution time': 0.018362760543823242, 'path cost': 20,
'complexity': 12}
{'puzzle': 65, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1463, 'execution time': 0.012485027313232422, 'path cost': 19,
'complexity': 16}
{'puzzle': 66, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1307, 'execution time': 0.011234283447265625, 'path cost': 19,
'complexity': 8}
{'puzzle': 67, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 37420, 'execution time': 0.6014070510864258, 'path cost': 27,
'complexity': 16}
{'puzzle': 68, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1221, 'execution time': 0.034062862396240234, 'path cost': 19,
'complexity': 10}
{'puzzle': 69, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 11320, 'execution time': 0.19829583168029785, 'path cost': 24,
'complexity': 12}
{'puzzle': 70, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 41896, 'execution time': 0.7108860015869141, 'path cost': 27,
'complexity': 24}
{'puzzle': 71, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 10562, 'execution time': 0.12481045722961426, 'path cost': 24,
'complexity': 16}
{'puzzle': 72, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 12691, 'execution time': 0.21930313110351562, 'path cost': 24,
'complexity': 20}
{'puzzle': 73, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 23626, 'execution time': 0.3331160545349121, 'path cost': 26,
'complexity': 18}
{'puzzle': 74, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 965, 'execution time': 0.021233081817626953, 'path cost': 18,
'complexity': 12}
{'puzzle': 75, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 9806, 'execution time': 0.2091834545135498, 'path cost': 23,
'complexity': 14}
{'puzzle': 76, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 50246, 'execution time': 0.7976658344268799, 'path cost': 28,
'complexity': 14}
{'puzzle': 77, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 58202, 'execution time': 1.053152084350586, 'path cost': 29,
'complexity': 20}
{'puzzle': 78, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',

'expanded nodes': 13139, 'execution time': 0.1567518711090088, 'path cost': 25,
'complexity': 20}
{'puzzle': 79, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1165, 'execution time': 0.016170978546142578, 'path cost': 19,
'complexity': 12}
{'puzzle': 80, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 40424, 'execution time': 0.6391825675964355, 'path cost': 27,
'complexity': 8}
{'puzzle': 81, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 2513, 'execution time': 0.046654701232910156, 'path cost': 21,
'complexity': 14}
{'puzzle': 82, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 488, 'execution time': 0.009750127792358398, 'path cost': 17,
'complexity': 16}
{'puzzle': 83, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 235, 'execution time': 0.003590106964111328, 'path cost': 15,
'complexity': 10}
{'puzzle': 84, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 18831, 'execution time': 0.28522706031799316, 'path cost': 25,
'complexity': 18}
{'puzzle': 85, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 3677, 'execution time': 0.041635751724243164, 'path cost': 21,
'complexity': 14}
{'puzzle': 86, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 24047, 'execution time': 0.4578266143798828, 'path cost': 26,
'complexity': 20}
{'puzzle': 87, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 8855, 'execution time': 0.09503793716430664, 'path cost': 23,
'complexity': 16}
{'puzzle': 88, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 8641, 'execution time': 0.17316937446594238, 'path cost': 23,
'complexity': 20}
{'puzzle': 89, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 25845, 'execution time': 0.3957529067993164, 'path cost': 26,
'complexity': 20}
{'puzzle': 90, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 28114, 'execution time': 0.5121207237243652, 'path cost': 27,
'complexity': 22}
{'puzzle': 91, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 12431, 'execution time': 0.13129234313964844, 'path cost': 24,
'complexity': 18}
{'puzzle': 92, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 54490, 'execution time': 0.9622504711151123, 'path cost': 28,
'complexity': 22}
{'puzzle': 93, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 55957, 'execution time': 1.0346555709838867, 'path cost': 28,
'complexity': 20}
{'puzzle': 94, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',

'expanded nodes': 17957, 'execution time': 0.3150303363800049, 'path cost': 25,
'complexity': 12}
{'puzzle': 95, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 143, 'execution time': 0.009967565536499023, 'path cost': 14,
'complexity': 10}
{'puzzle': 96, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 24077, 'execution time': 0.3556935787200928, 'path cost': 26,
'complexity': 8}
{'puzzle': 97, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 2871, 'execution time': 0.03801918029785156, 'path cost': 21,
'complexity': 14}
{'puzzle': 98, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 23632, 'execution time': 0.3344442844390869, 'path cost': 26,
'complexity': 14}
{'puzzle': 99, 'algorithm': 'astar_search', 'heuristic': 'hamming_distance',
'expanded nodes': 4683, 'execution time': 0.05453848838806152, 'path cost': 22,
'complexity': 12}
{'puzzle': 0, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 218, 'execution time': 0.0017588138580322266, 'path cost': 64,
'complexity': 18}
{'puzzle': 1, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 342, 'execution time': 0.002851247787475586, 'path cost': 78,
'complexity': 8}
{'puzzle': 2, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 862, 'execution time': 0.007213592529296875, 'path cost': 99,
'complexity': 24}
{'puzzle': 3, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1085, 'execution time': 0.009540557861328125, 'path cost':
103, 'complexity': 4}
{'puzzle': 4, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 289, 'execution time': 0.0028166770935058594, 'path cost': 76,
'complexity': 10}
{'puzzle': 5, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 292, 'execution time': 0.0023005008697509766, 'path cost': 74,
'complexity': 18}
{'puzzle': 6, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 672, 'execution time': 0.0053348541259765625, 'path cost': 71,
'complexity': 12}
{'puzzle': 7, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 523, 'execution time': 0.0043065547943115234, 'path cost': 86,
'complexity': 12}
{'puzzle': 8, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1304, 'execution time': 0.010708332061767578, 'path cost':
180, 'complexity': 16}
{'puzzle': 9, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 913, 'execution time': 0.09381556510925293, 'path cost': 155,
'complexity': 12}
{'puzzle': 10, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',

'expanded nodes': 1169, 'execution time': 0.009715795516967773, 'path cost':
169, 'complexity': 12}
{'puzzle': 11, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 439, 'execution time': 0.004145145416259766, 'path cost': 111,
'complexity': 18}
{'puzzle': 12, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1319, 'execution time': 0.01125645637512207, 'path cost': 176,
'complexity': 8}
{'puzzle': 13, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 435, 'execution time': 0.0038766860961914062, 'path cost': 75,
'complexity': 16}
{'puzzle': 14, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 355, 'execution time': 0.0028390884399414062, 'path cost': 59,
'complexity': 14}
{'puzzle': 15, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1576, 'execution time': 0.013931751251220703, 'path cost':
173, 'complexity': 14}
{'puzzle': 16, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 525, 'execution time': 0.0046825408935546875, 'path cost': 88,
'complexity': 16}
{'puzzle': 17, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1414, 'execution time': 0.012508630752563477, 'path cost':
184, 'complexity': 12}
{'puzzle': 18, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 513, 'execution time': 0.00442194938659668, 'path cost': 79,
'complexity': 14}
{'puzzle': 19, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 490, 'execution time': 0.003976345062255859, 'path cost': 53,
'complexity': 18}
{'puzzle': 20, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 815, 'execution time': 0.00695037841796875, 'path cost': 134,
'complexity': 18}
{'puzzle': 21, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 348, 'execution time': 0.0029799938201904297, 'path cost': 94,
'complexity': 16}
{'puzzle': 22, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1043, 'execution time': 0.008375167846679688, 'path cost': 97,
'complexity': 14}
{'puzzle': 23, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1266, 'execution time': 0.010496377944946289, 'path cost': 91,
'complexity': 12}
{'puzzle': 24, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 575, 'execution time': 0.0049896240234375, 'path cost': 101,
'complexity': 16}
{'puzzle': 25, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1003, 'execution time': 0.008238554000854492, 'path cost': 75,
'complexity': 14}
{'puzzle': 26, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',

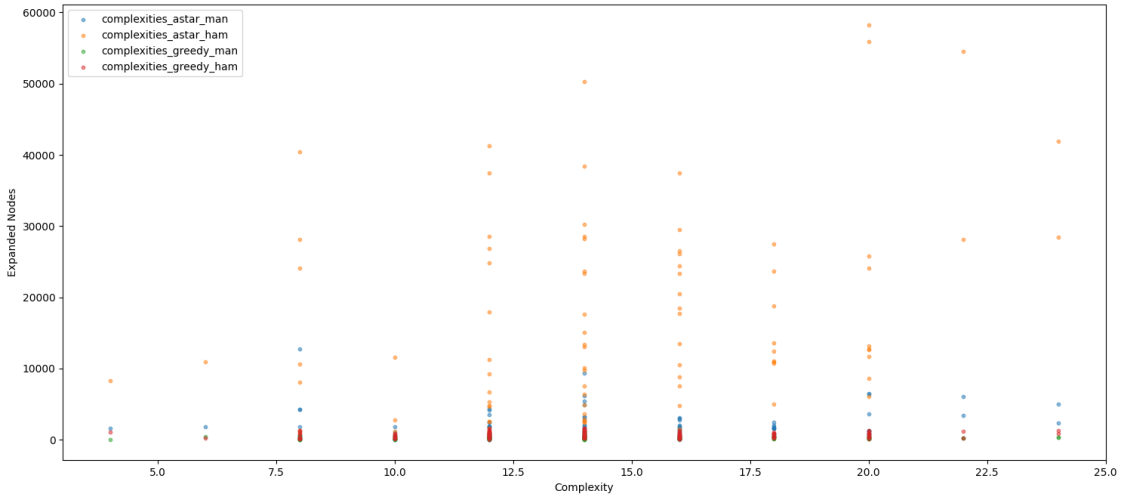
'expanded nodes': 525, 'execution time': 0.0043315887451171875, 'path cost':
117, 'complexity': 12}
{'puzzle': 27, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 279, 'execution time': 0.0022325515747070312, 'path cost': 78,
'complexity': 6}
{'puzzle': 28, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 716, 'execution time': 0.006018400192260742, 'path cost': 151,
'complexity': 12}
{'puzzle': 29, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 416, 'execution time': 0.0034422874450683594, 'path cost': 96,
'complexity': 16}
{'puzzle': 30, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 301, 'execution time': 0.0026009082794189453, 'path cost': 49,
'complexity': 20}
{'puzzle': 31, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 648, 'execution time': 0.005084514617919922, 'path cost': 43,
'complexity': 12}
{'puzzle': 32, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 449, 'execution time': 0.003694295883178711, 'path cost': 95,
'complexity': 20}
{'puzzle': 33, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1253, 'execution time': 0.010383367538452148, 'path cost':
112, 'complexity': 14}
{'puzzle': 34, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 363, 'execution time': 0.003221750259399414, 'path cost': 87,
'complexity': 14}
{'puzzle': 35, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 801, 'execution time': 0.006667375564575195, 'path cost': 99,
'complexity': 16}
{'puzzle': 36, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 132, 'execution time': 0.0011684894561767578, 'path cost': 37,
'complexity': 8}
{'puzzle': 37, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 163, 'execution time': 0.0011873245239257812, 'path cost': 56,
'complexity': 8}
{'puzzle': 38, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 874, 'execution time': 0.0070416927337646484, 'path cost':
100, 'complexity': 10}
{'puzzle': 39, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 492, 'execution time': 0.0044248104095458984, 'path cost': 98,
'complexity': 20}
{'puzzle': 40, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 373, 'execution time': 0.003289937973022461, 'path cost': 93,
'complexity': 12}
{'puzzle': 41, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1176, 'execution time': 0.010747194290161133, 'path cost':
147, 'complexity': 8}
{'puzzle': 42, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',

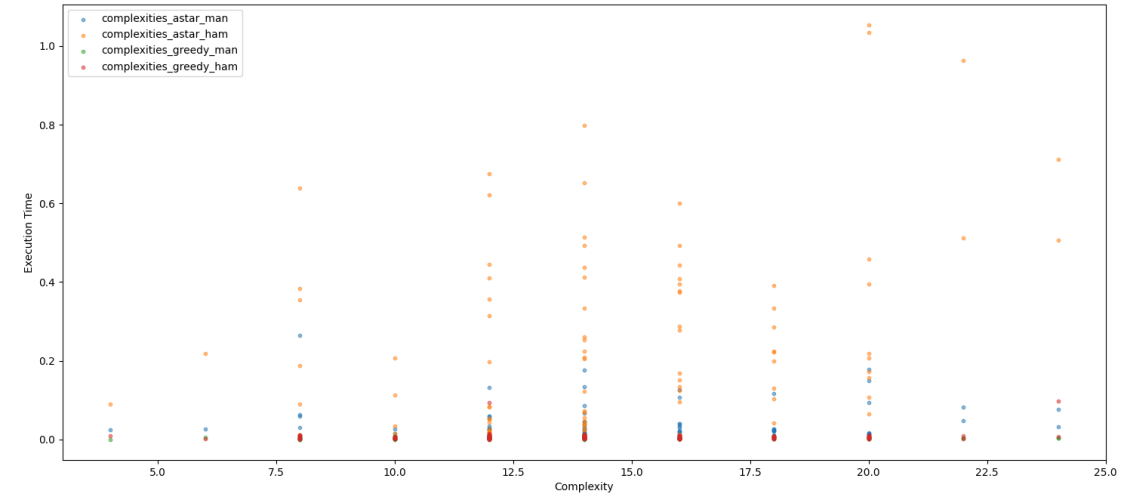
'expanded nodes': 953, 'execution time': 0.00839853286743164, 'path cost': 124,
'complexity': 12}
{'puzzle': 43, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1646, 'execution time': 0.015170812606811523, 'path cost':
115, 'complexity': 12}
{'puzzle': 44, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1047, 'execution time': 0.008637189865112305, 'path cost':
156, 'complexity': 16}
{'puzzle': 45, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 866, 'execution time': 0.007432222366333008, 'path cost': 80,
'complexity': 14}
{'puzzle': 46, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 741, 'execution time': 0.0060999393463134766, 'path cost': 65,
'complexity': 10}
{'puzzle': 47, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 702, 'execution time': 0.00569462776184082, 'path cost': 128,
'complexity': 12}
{'puzzle': 48, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 370, 'execution time': 0.0031023025512695312, 'path cost': 56,
'complexity': 14}
{'puzzle': 49, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 383, 'execution time': 0.003287792205810547, 'path cost': 83,
'complexity': 12}
{'puzzle': 50, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 926, 'execution time': 0.0073544979095458984, 'path cost': 81,
'complexity': 16}
{'puzzle': 51, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1390, 'execution time': 0.011767148971557617, 'path cost':
181, 'complexity': 14}
{'puzzle': 52, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 687, 'execution time': 0.005789756774902344, 'path cost': 123,
'complexity': 14}
{'puzzle': 53, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1380, 'execution time': 0.011240243911743164, 'path cost':
171, 'complexity': 14}
{'puzzle': 54, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 268, 'execution time': 0.0025069713592529297, 'path cost': 62,
'complexity': 12}
{'puzzle': 55, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 669, 'execution time': 0.005560159683227539, 'path cost': 107,
'complexity': 14}
{'puzzle': 56, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 65, 'execution time': 0.0007276535034179688, 'path cost': 27,
'complexity': 12}
{'puzzle': 57, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 464, 'execution time': 0.0036139488220214844, 'path cost': 63,
'complexity': 16}
{'puzzle': 58, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',

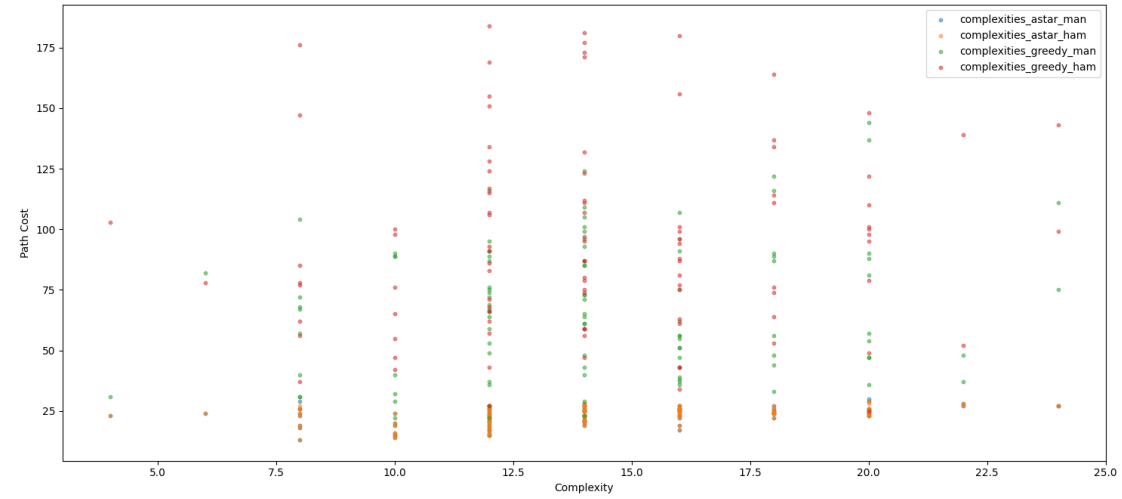
'expanded nodes': 170, 'execution time': 0.0014445781707763672, 'path cost': 42,
'complexity': 10}
{'puzzle': 59, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1614, 'execution time': 0.013759613037109375, 'path cost':
177, 'complexity': 14}
{'puzzle': 60, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 236, 'execution time': 0.002267599105834961, 'path cost': 59,
'complexity': 14}
{'puzzle': 61, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 608, 'execution time': 0.004888772964477539, 'path cost': 87,
'complexity': 14}
{'puzzle': 62, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 954, 'execution time': 0.007947206497192383, 'path cost': 87,
'complexity': 16}
{'puzzle': 63, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 828, 'execution time': 0.006860017776489258, 'path cost': 114,
'complexity': 18}
{'puzzle': 64, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 823, 'execution time': 0.007138490676879883, 'path cost': 68,
'complexity': 12}
{'puzzle': 65, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 114, 'execution time': 0.001047372817993164, 'path cost': 43,
'complexity': 16}
{'puzzle': 66, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 792, 'execution time': 0.006565093994140625, 'path cost': 77,
'complexity': 8}
{'puzzle': 67, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 523, 'execution time': 0.0043065547943115234, 'path cost': 77,
'complexity': 16}
{'puzzle': 68, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 318, 'execution time': 0.0026988983154296875, 'path cost': 55,
'complexity': 10}
{'puzzle': 69, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1120, 'execution time': 0.009361743927001953, 'path cost':
134, 'complexity': 12}
{'puzzle': 70, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1323, 'execution time': 0.09856438636779785, 'path cost': 143,
'complexity': 24}
{'puzzle': 71, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 84, 'execution time': 0.0012106895446777344, 'path cost': 34,
'complexity': 16}
{'puzzle': 72, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1191, 'execution time': 0.009827852249145508, 'path cost':
122, 'complexity': 20}
{'puzzle': 73, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 798, 'execution time': 0.006829500198364258, 'path cost': 164,
'complexity': 18}
{'puzzle': 74, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',

'expanded nodes': 983, 'execution time': 0.008123636245727539, 'path cost': 106,
'complexity': 12}
{'puzzle': 75, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 558, 'execution time': 0.0048558712005615234, 'path cost': 95,
'complexity': 14}
{'puzzle': 76, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 807, 'execution time': 0.006718873977661133, 'path cost': 74,
'complexity': 14}
{'puzzle': 77, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1336, 'execution time': 0.011390924453735352, 'path cost':
101, 'complexity': 20}
{'puzzle': 78, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 102, 'execution time': 0.0013060569763183594, 'path cost': 25,
'complexity': 20}
{'puzzle': 79, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 253, 'execution time': 0.0020513534545898438, 'path cost': 57,
'complexity': 12}
{'puzzle': 80, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 278, 'execution time': 0.0022242069244384766, 'path cost': 85,
'complexity': 8}
{'puzzle': 81, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 263, 'execution time': 0.002053976058959961, 'path cost': 59,
'complexity': 14}
{'puzzle': 82, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 112, 'execution time': 0.0009403228759765625, 'path cost': 43,
'complexity': 16}
{'puzzle': 83, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 436, 'execution time': 0.0034246444702148438, 'path cost': 47,
'complexity': 10}
{'puzzle': 84, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1088, 'execution time': 0.010148286819458008, 'path cost':
137, 'complexity': 18}
{'puzzle': 85, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 190, 'execution time': 0.0019845962524414062, 'path cost': 47,
'complexity': 14}
{'puzzle': 86, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 860, 'execution time': 0.007077932357788086, 'path cost': 110,
'complexity': 20}
{'puzzle': 87, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 215, 'execution time': 0.0019364356994628906, 'path cost': 61,
'complexity': 16}
{'puzzle': 88, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 866, 'execution time': 0.006934642791748047, 'path cost': 79,
'complexity': 20}
{'puzzle': 89, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 659, 'execution time': 0.0055806636810302734, 'path cost':
100, 'complexity': 20}
{'puzzle': 90, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',

'expanded nodes': 1146, 'execution time': 0.009779691696166992, 'path cost':
139, 'complexity': 22}
{'puzzle': 91, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 850, 'execution time': 0.007109642028808594, 'path cost': 76,
'complexity': 18}
{'puzzle': 92, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 202, 'execution time': 0.0018875598907470703, 'path cost': 52,
'complexity': 22}
{'puzzle': 93, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 831, 'execution time': 0.006554603576660156, 'path cost': 148,
'complexity': 20}
{'puzzle': 94, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 885, 'execution time': 0.007346630096435547, 'path cost': 107,
'complexity': 12}
{'puzzle': 95, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 699, 'execution time': 0.005907773971557617, 'path cost': 98,
'complexity': 10}
{'puzzle': 96, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 856, 'execution time': 0.0068302154541015625, 'path cost': 62,
'complexity': 8}
{'puzzle': 97, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 955, 'execution time': 0.008291482925415039, 'path cost': 111,
'complexity': 14}
{'puzzle': 98, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 1096, 'execution time': 0.009649038314819336, 'path cost':
132, 'complexity': 14}
{'puzzle': 99, 'algorithm': 'greedy_search', 'heuristic': 'hamming_distance',
'expanded nodes': 516, 'execution time': 0.004603385925292969, 'path cost': 66,
'complexity': 12}


complexities_astar_man
complexities_astar_ham
complexities_greedy_man
complexities_greedy_ham
Expanded Nodes
Complexity


complexities_astar_man
complexities_astar_ham
complexities_greedy_man
complexities_greedy_ham
Execution Time
Complexity


complexities_astar_man
complexities_astar_ham
complexities_greedy_man
complexities_greedy_ham
Path Cost
Complexity

[ ]: