CS 486/686 Assignment 1 (140 marks)

Sample Solutions

Alice Gao

Due Date: 11:59 pm on Wednesday, June 9, 2021

Changes

- Q1: Changed the descriptions to use wolf and sheep instead of missionaries and cannibals. Changed the formulation to use S_j instead of M_j for a sheep and W_j instead of C_j for a wolf. This change does not affect the problem solution at all.

Academic Integrity Statement

If your written submission on Learn does not include this academic integrity statement with your signature (typed name), we will deduct 5 marks from your final assignment mark.

I declare the following statements to be true:

- The work I submit here is entirely my own.
- I have not shared and will not share any of my code with anyone at any point.
- I have not posted and will not post my code on any public or private forum or website.
- I have not discussed and will not discuss the contents of this assessment with anyone at any point.
- I have not posted and will not post the contents of this assessment and its solutions on any public or private forum or website.
- I will not search for assessment solutions online.
- I am aware that misconduct related to assessments can result in significant penalties, possibly including failure in the course and suspension. This is covered in Policy 71: https://uwaterloo.ca/secretariat/policies-procedures-guidelines/policy-71.

By typing or writing my full legal name below, I confirm that I have read and understood the academic integrity statement above.

Instructions

- Submit any written solutions in a file named `writeup.pdf` to the A1 Dropbox on `Learn`. Submit any code to `Marmoset` at https://marmoset.student.cs.uwaterloo.ca/. No late assignment will be accepted. This assignment is to be done individually.
- I strongly encourage you to complete your write-up in Latex, using this source file. If you do, in your submission, please replace the author with your name and student number. Please also remove the due date, the Instructions section, and the Learning goals section. Thanks!
- Lead TAs:
  - Xinda Li (xinda.li@uwaterloo.ca)
  - Josh Jung (j35jung@uwaterloo.ca)

  The TAs' office hours will be posted on MS Teams.

Learning goals

Uninformed and Heuristic Search

- Formulate a given problem as a heuristic search problem. Define the states, the initial state, the goal states, the action, the successor function, and the cost function.
- Trace the execution of Breadth-First Search and Depth-First Search.
- Define an admissible heuristic by solving a relaxed problem optimally. Explain why a heuristic is admissible.
- Trace the execution of the A* search algorithm with an admissible heuristic.
- Implement the A* search algorithm.

1 The Wolves and Sheep Problem (42 marks)

There are three wolves, three sheep, and a boat on the left side of the river. Let's assume that the wolves and sheep know how to row the boat. The boat can hold one or two animals. Every time the boat crosses the river, we count it as one boat trip. Our goal is to find a way to move all the animals from the left side to the right side of the river using the smallest number of boat trips subject to the constraints below.

- The boat can only cross the river if there is at least one animal in it.
- The sheep must stay alive.

  At any moment in time, on either side of the river, if there is at least one sheep and the number of wolves is greater than the number of sheep, the wolves will eat the sheep.

You will apply uninformed and informed search algorithms to solve this problem.

Please complete the following tasks.

(a) Formulate this problem as a search problem. Make sure you define the states, the initial state, the goal state, the successor function, and the cost function.

In parts b and c, we will provide you with some other formulations. After that, you will have a chance to revise and improve your formulation in part d. Come up with your own formulation first before looking at the next two parts. Don't miss this valuable learning opportunity.

Marking Scheme: (2 marks) We will mark this part for completion only. As long as you provide a state definition and a successor function, you will get the 2 marks.

- (1 mark) for stating a state definition.
- (1 mark) for stating a successor function.

Solutions:

See part d.

(b) Your friend Taylor came up with the following formulation. Taylor's formulation is correct, which means applying a (optimal) search algorithm on this formulation will allow us to find a (optimal) solution. Unfortunately, Taylor's formulation has a problem, which significantly affects the performance of a search algorithm on this problem.

Describe the problem with Taylor's formulation in no more than 3 sentences. Then, discuss how this problem affects the performance of a search algorithm in no more than a paragraph.

Hint: If you have trouble identifying the problem, draw the search graph and look at the states in the search graph. How many states are there? Why are these states in the graph?

Taylor's formulation:

- **State:** Each state is given by $S_1S_2S_3W_1W_2W_3B$. S_i and W_j are 1 if the sheep/wolf is on the left bank and 0 otherwise. $B = d$ if the boat is on the left bank (departure side) and $B = a$ if the boat is on the right bank (arrival side).

  A state is valid if and only if it satisfies the following two constraints.

  If there is at least one sheep on the left bank, then the number of wolves on the left bank must be less than or equal to the number of sheep on the left bank.

  If there is at least one sheep on the right bank, then the number of wolves on the right bank must be less than or equal to the number of sheep on the right bank.
- **Initial state:** $111111d$.
- **Goal states:** Any state where every animal is on the right bank is a goal state. $000000x$ where x can be a or d.
- **Successor function:** Assume that A and B are two states satisfying the state definition. B is a successor of A if and only if we can start from A, move the boat with one or two animals from one side of the river to the other side of the river, and arrive at state B.
- **Cost function:** The cost of each boat trip is one.

Marking Scheme: (4 marks)

- (2 mark) Describe the problem with Taylor's formulation.
- (2 marks) Describe how the problem affects the performance of a search algorithm.

Solutions:

The problem with Taylor's formulation is that each state is keeping track of a lot of unnecessary information.

- Each state is keeping track of the individual identities of the sheep and the wolves. We only need to keep track of the number of the sheep and the wolves. For instance, 110000d should be the same state as 101000d.

- Each state is keeping track of the sheep and the wolves on both sides of the river. We only need to keep track of the number of sheep and wolves on one side. For instance, if we know that there are 2 sheep on the left side, then we already know that there is 1 sheep on the right side.

This problem has a few consequences.

- The large total number of states results in a worse time complexity for most search algorithms.
- The branching factor is large, which causes search algorithms such as Breadth-First Search to be very slow.
- With the redundant information, many states are equivalent. Therefore, many paths in the search graph represent the same solution, which many cause us to explore more paths than necessary.

Having 000000d as a goal state is not a problem of the formulation. 000000d is a valid goal state because we do not care about the location of the boat in the goal state. In a goal state, we only require that all the animals are on the right side of the river and 000000d satisfies this requirement.

Another way to think about the goal states is that we can encode the goal states using a goal test. The goal test says that a state is a goal state as long as all the animals are on the right side of the river. Given this goal test, we have two goal states satisfying it.

(c) Your other friend Avery proposed a formulation that is similar to Taylor's. However, Taylor included the constraints in the state definition, whereas Avery put the constraints in the successor function. See the state and the successor function in the two formulations below.

Taylor and Avery argued for a while but couldn't figure out which choice is better. Could you help them settle the argument? *Which of the two formulations is better? State your answer and justify your answer in no more than a paragraph.*

Taylor's formulation:

- **State:** Each state consists of $(S_1, S_2, S_3, W_1, W_2, W_3, B)$. S_i and W_j are 1 if the sheep/wolf is on the left bank and 0 otherwise. $B = d$ if the boat is on the left bank (departure side) and $B = a$ if the boat is on the right bank (arrival side).

  A state is invalid if it violates either constraint below.

- If there is at least one sheep on the left bank, then the number of wolves on the left bank must be less than or equal to the number of sheep on the left bank.
- If there is at least one sheep on the right bank, then the number of wolves on the right bank must be less than or equal to the number of sheep on the right bank.

- **Successor function:** Assume that A and B are two states satisfying the state definition. B is a successor of A if and only if we can start from A, move the boat with one or two animals from one side of the river to the other side of the river, and arrive at state B.

Avery's formulation:

- **State:** Each state consists of $(S_1, S_2, S_3, W_1, W_2, W_3, B)$. S_i and W_j are 1 if the sheep/wolf is on the left bank and 0 otherwise. $B = d$ if the boat is on the left bank (departure side) and $B = a$ if the boat is on the right bank (arrival side).
- **Successor function:** Consider a state A that satisfies the state definition. A does not have any successor if it violates either constraint below.
  - If there is at least one sheep on the left bank, then the number of wolves on the left bank must be less than or equal to the number of sheep on the left bank.
  - If there is at least one sheep on the right bank, then the number of wolves on the right bank must be less than or equal to the number of sheep on the right bank.

  If state A satisfies both constraints above, then we will determine A's successor states as follows. Assume that B is a state satisfying the state definition. B is a successor of A if and only if we can start from A, move the boat with one or two animals from one side of the river to the other side of the river, and arrive at state B.

Marking Scheme: (4 marks)

- (2 marks) Correct answer
- (2 marks) A reasonable justification

Solutions:

Taylor's formulation (putting the constraints in the state definition) is better.

Here is the main difference between the two formulations.

- In Taylor's formulation, a valid state has to satisfy all the constraints. This ensures that the total number of states is as small as possible. After all, it is unnecessary for us to consider states that do not satisfy the constraints.
- In Avery's formulation, the most important sentence is that "A state does not have any successors if it (i.e. the state) violates either constraint below." This means that, if a state violates a constraint, it is still a valid state and will appear in the search graph. However, since the state has no successors, we cannot get to any other state from such a state. As a result, Avery's formulation has many more states than that of Taylor's formulation.

Putting the constraints in the successor function (as in Avery's formulation) will increase the total number of states. Cypress's formulation has more states than Taylor's formulation and some states in Avery's formulation violate the constraints. By using Avery's formulation, we might end up exploring some states that violate the constraints, which is unnecessary for solving this problem.

(d) Based on the previous parts, come up with the best formulation for this problem. Make sure you define the states, the initial state, the goal state, and the successor function. Assume the cost function is the same as the one in Taylor's formulation.

We will mark your formulation on its correctness and its quality.

Marking Scheme: (10 marks)

- (4 marks) The quality of your formulation.
- (2 marks) State definition.
- (2 marks) Constraints.
- (1 mark) Initial state and goal states.
- (1 mark) Successor function.

Solutions:

The best solution is as follows.

- State: Each state is given by MCB. M is the number of sheep on the left bank $M \in \{0, 1, 2, 3\}$. C is the number of wolves on the left bank $C \in \{0, 1, 2, 3\}$. $B = d$ if the boat is on the left bank (departure side) and $B = a$ if the boat is on the right bank (arrival side).

  A state is valid if and only if it satisfies the following two constraints.

  If there is at least one sheep on the left bank, then the number of wolves on the left bank must be less than or equal to the number of sheep on the left bank. Formally, if $M \geq 1$, then $C \leq M$.

If there is at least one sheep on the right bank, then the number of wolves on the right bank must be less than or equal to the number of sheep on the right bank. Formally, if $(3-M) \geq 1$, then $(3-C) \leq (3-M)$. After simplifying, this condition becomes: if $M \leq 2$, then $C \geq M$.

To summarize: If $M = 1$, then $C = M = 1$. If $M = 2$, then $C = M = 2$. If $M = 0$ or $M = 3$, then C can be any value.

- Initial state: $33d$.
- Goal states: $00a$ and $00d$.
  Our goal is to move all the animals to the right bank. We don't care where the boat is at the end. Therefore, $00d$ is a goal state even though it is unreachable.
- Successor function: Given any state, we can generate a successor state by moving the boat with one or two animals from one side of the river to the other side of the river.
- Cost function: The cost of each boat trip is one.

(e) Draw the search graph. The search graph should contain the states and the arcs in your problem formulation in part d. Highlight the start state and the goal states in your graph.

If you search graph has more than 35 nodes in it, go back to part d and rethink your formulation.

Marking Scheme: (6 marks)

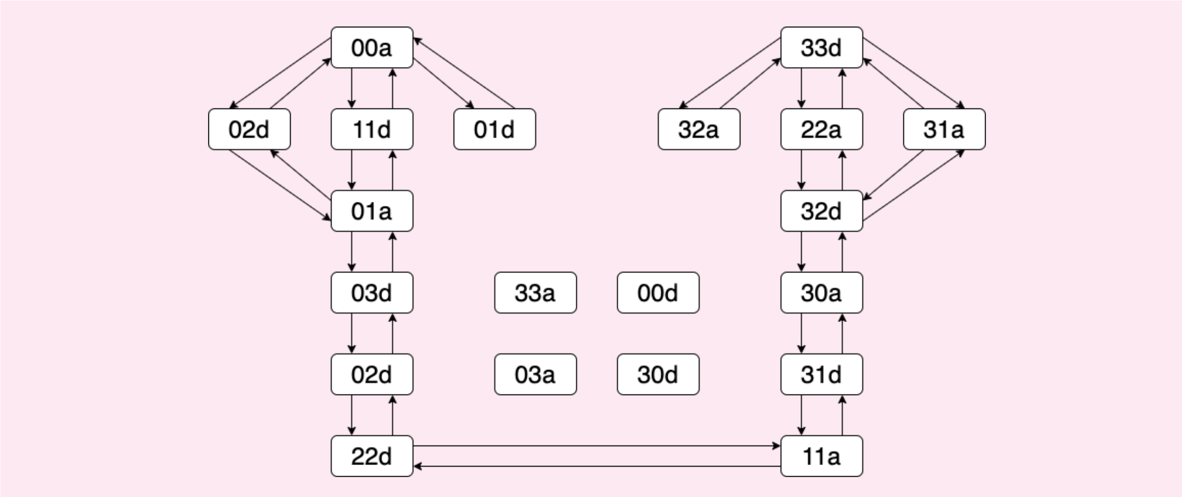
- (4 marks) includes all the states (and handles the constraints correctly).
- (2 marks) draws the arcs correctly (and handles the constraints correctly).

Solutions:

Common mistakes:

- Does not include unreachable states.
- Include duplicate states.
- The search graph will be much more complicated if the state definition does not include the constraints.

Search graph for the best solution:



(f) If your goal is to find a solution quickly (while minimizing the number of nodes expanded), which **uninformed search algorithm** would you use to solve this problem?

You can choose among breadth first search, depth first search, iterative-deepening search, and lowest-cost-first search. You can also use any pruning strategy discussed in lectures. Since you already drew the search graph, you can assume that it is given.

State the algorithm of your choice and justify your answer in at most 5 sentences.

Marking Scheme:

(3 marks) Provide a good reason for choosing a particular uninformed algorithm.

Solutions:

The best uninformed search algorithm is depth-first search with cycle pruning. The search graph contains a main path without many alternative paths. Going deep into the search tree is the best way to find a solution quickly.

(g) Propose an admissible heuristic function h. Describe the heuristic function in detail. Then, explain why the heuristic function is admissible. Some marks will be assigned to the quality of the heuristic function.

Hint: A heuristic function must specify a value for every state in the search graph.

Hint: If you relax the problem by removing a constraint, then the optimal solution to the relaxed problem is guaranteed to be an admissible heuristic function. One way to justify why your heuristic function is admissible is to explain how you relaxed the problem and solved the relaxed problem optimally.

Marking Scheme: (10 marks)

- (6 marks) Describe the heuristic function.
- (2 marks) Describe why the heuristic function is admissible.
- (2 marks) The quality of your heuristic function.

Solutions:

Common mistakes:

- Does not account for both the left and the right banks.
- Does not take care of special cases such as when there are 1 or 2 animals on the left bank.
- (2 marks) The quality of the heuristic function: 2 marks for heuristic 1 or a better heuristic. 1 mark for heuristic 2. 0 marks for a heuristic function, which is worse than heuristic 2.

We derived two possible heuristic functions.

Heuristic #1:

We relax the problem as follows.

- Discard the constraint that the sheep cannot be outnumbered by the wolves at either bank at any time.

We can solve this relaxed problem as follows.

- If there are at most 2 animals on the left bank, we can move them over in 1 trip.
- If there are more than 2 animals on the left bank, we can move 1 animal over in each trip by moving 2 animals to the right bank and then moving 1 animal back. We will keep doing this until there are 2 animals on the left bank. Then, we can move both animals to the right bank in 1 trip.

Given this, the optimal solution to this relaxed problem is as follows. Let there be n animals on the left bank.

If the boat is on the left bank:

- If $n = 0$, $h(n) = 0$.
- If $1 \leq n \leq 2$, $h(n) = 1$.
- If $n \geq 3$, $h(n) = (n - 2) * 2 + 1 = 2n - 3$.

If the boat is on the right bank and $n > 0$, we need to bring 1 animal back to the left bank using the boat. Thus, this is the same as the scenario where there are $n + 1$ animals on the left bank.

- $h(n) = 2n$.

Heuristic #2:

We relax the problem as follows.

- Discard the constraint that the sheep cannot be outnumbered by the wolves at either bank at any time.
- The boat can travel by itself with no animal in it.

We can solve this relaxed problem as follows.

- If there are at most 2 animals on the left bank, we can move them over in 1 trip.
- If there are more than 2 animals on the left bank, we can move 2 animals over in each trip. We will keep doing this until there are at most 2 animals on the left bank. Then, we can move the 1-2 animals to the right bank in 1 trip.

Given this, the optimal solution to this relaxed problem is as follows. Let there be n animals on the left bank.

If the boat is on the left bank:

- If $n = 0$, $h(n) = 0$.
- If $n \leq 2$, $h(n) = 1$.
- If $n \geq 3$, $h(n) = 2\lfloor \frac{n-1}{2} \rfloor + 1$.

If the boat is on the right bank and $n > 0$, we need to bring the boat back to the left bank first. This adds one trip to the solutions above.

- If $n = 0$, $h(n) = 0$.
- If $n \leq 2$, $h(n) = 2$.
- If $n \geq 3$, $h(n) = 2\lfloor \frac{n-1}{2} \rfloor + 2$.

(h) Your friend is considering using **the A* search algorithm** instead of using an uninformed search algorithm to solve the sheep and wolves problem. Would the A* search algorithm be a better choice than an uninformed search algorithm? Why or why not? State your answer and justify it in at most 5 sentences.

Tip: Do not trace the algorithm on the problem. We are not looking for the exact number of states expanded for either algorithm. Instead, formulate your answer by looking at the search graph and reasoning about the behaviour of the search algorithms.

Marking Scheme: (3 marks) A good justification for your answer.

Solutions:

A* search is not a better choice than an uninformed search algorithm.

The branching factor of the search graph is very small (almost 1 everywhere except when we are 1 step from the start or the goal node). There are not a lot of alternative paths to explore. If we use an uninformed search algorithm, then the order in which it adds nodes to the frontier will affect how many nodes we explore. However, in general, there are not a lot of opportunities for an uninformed search algorithm to explore unnecessary paths.

2 The Rush Hour Sliding Puzzle (98 marks)

In this programming question, you will solve the Rush Hour puzzle using the A* search and the depth-first search algorithms.

Take a look at an example of a Rush Hour puzzle below. The puzzle is on a 6 by 6 grid. We will number the rows as 0 to 5 from the top, and the columns as 0 to 5 from the left. In row 2, a horizontal car of length 2, called the goal car, is trying to escape through the exit on the right. There are horizontal and vertical cars of various lengths in the grid. A horizontal car can only move horizontally, whereas a vertical car can only move vertically. Each car may move more than one square in one step, but it cannot move over or through other cars. The goal is to move the cars around until the goal car reaches the exit, i.e. until the goal car is in the columns 4 and 5 in row 2.

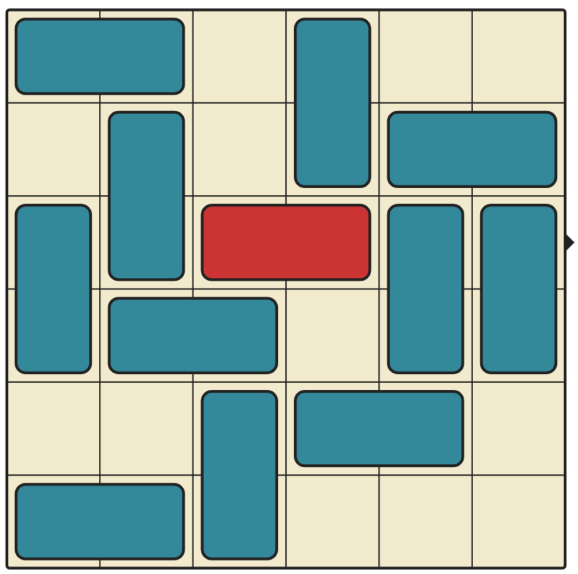


Figure 1: An example of a Rush Hour puzzle from https://www.michaelfogleman.com/rush/

You can make the following assumptions for this question.

- The puzzle must be on a 6 x 6 grid.
- The goal car is in row 2 and it has a length of 2.
- Besides the goal car, there is no other horizontal car in row 2.

Information on the Provided Code

We have provided three files `board.py`, `solve.py`, and `jams_posted.txt` on Learn. `board.py` contains code for handling input/output, representing states, etc. Your main task is to fill in the empty functions in `solve.py`.

Submit the `solve.py` to Marmoset only. We will use our version of `board.py` to test your code. Do not modify any provided function signatures in `solve.py`. Doing so will cause you to fail our tests. Feel free to add any new code to `solve.py`.

Input Format

The file `jams_posted.txt` contains 40 puzzles. You can use these puzzles to test your program. We will test your program using a different set of puzzles.

Below is an example of an input describing a puzzle.

```
example
6
1 2 h 2
2 0 v 2
4 0 h 2
3 1 v 3
4 1 v 2
4 3 h 2
.
```

- The first line assigns a name to the puzzle. In this case, the name is "example."
- The next line specifies the size of the grid. We only use 6 by 6 grid. So this number is always 6.
- The next line "`1 2 h 2`" gives a description of the goal car. The first two numbers $(1, 2)$ gives the (x, y) coordinates of the upper left corner of the car. The next letter "h" indicates that the car is horizontal ("v" would indicate that the car is vertical). The last number "2" indicates that the car has size 2.
- Each subsequent line, except the last line, describe a car in the puzzle, using the same format.
- The last line consists of a single period, indicating the end of the puzzle.

You can include multiple puzzles consecutively in the same file using the above format.

The Heuristic Functions for A* Search:

We have provided the implementation of the zero Heuristic function, which assigns a heuristic value of 0 to every state.

You must implement two other heuristic functions for A* search.

- Blocking Heuristic: The heuristic value for any goal state is zero. For any non-goal state, the heuristic value is one plus the number of cars blocking the path to the exit. For example, for the state in Figure 1, the blocking heuristic value is 3.
- Advanced Heuristic: Implement a third advanced heuristic of your choosing and invention. Your advanced heuristic should be consistent and should dominate the blocking heuristic.

Testing Your Program

Debugging and testing are essential skills for computer scientists. For this question, debugging your program may be especially challenging because of ties. Among "correct" implementations, the number of nodes expanded may vary widely due to how we handle the nodes with the same heuristic value on the frontier.

Please test your code using **Python 3.8.5**.

We rely on Python's hashing to generate a state's ID for breaking ties (see the Breaking Ties section below). However, Python' hashing function is not deterministic across different sessions. For example, you may get different hashing values of the same object for running your program multiple times. **Please set the environment variable PYTHONHASHSEED to 486 BEFORE running the Python script.** Note that setting the variable in the code/program will not work.

Implement **multi-path pruning for both DFS and A***. When there are multiple paths to a state, multi-path pruning explores the first path to the state and discards any subsequent path to the same state. Use an explored set to keep track of the states that have been expanded by the algorithm. When you **remove** a state from the frontier, check whether the state is in the explored set or not. If the state is in the explored set, then do nothing. Otherwise, add the state to the explored set and continue with the algorithm. Note that we perform pruning after we **remove** a state from the frontier, not before we **add** a state to the frontier.

DFS's behaviour depends on the order of adding a state's successors to the frontier. We will break ties by using the states' ID values. At each step, DFS will add the successors to the frontier in **decreasing** order of their IDs. In other words, DFS will expand the state with the smallest ID value among the successors.

A* search will also break ties using the states' ID values. Among several states with the same f value, A* will expand the state with the smallest ID value. If two states have the same ID value, A* will break ties using the states' parents — expanding the state whose parent has the smaller ID value.

Please complete the following tasks:

Submit your solutions to part (a) on Marmoset and submit your solutions to parts (b) and (c) on Learn.

(a) Complete the empty functions in `solve.py` and submit `solve.py` on Marmoset. Marmoset will evaluate your program for its correctness and efficiency.

For correctness, we have written unit tests for these functions: `get_path`, `is_goal`, `blocking_heuristic`, `get_successors`, `dfs`, `a_star`.

For each function, Marmoset provides one public test, which tests the function in a trivial scenario. There are also several secret tests. Before the deadline, you can only view the results of the public tests. After the deadline, Marmoset will run all the tests and calculate your marks.

Each test runs the function up to a predefined time limit. If the test passes if and only if the function terminates within the time limit and returns the expected result. Each test is all or nothing — there are no partial marks available.

Marking Scheme: (88 marks)

Unit tests on `get_path`, `is_goal`, `blocking_heuristic`, `get_successors`, `dfs`, and `a_star`.

- `get_path`: (1 public test + 2 secret tests) * 1 mark = 3 marks.
- `is_goal`: (1 public test + 4 secret tests) * 1 mark = 5 marks.
- `blocking_heuristic`: (1 public test + 9 secret tests) * 2 marks = 20 marks.
- `get_successors`: (1 public test + 9 secret tests) * 2 marks = 20 marks.
- `dfs`: (1 public test + 9 secret tests) * 2 marks = 20 marks.
- `a_star`: (1 public test + 9 secret tests) * 2 marks = 20 marks.

(b) Prove that the blocking heuristic is consistent using the definition of a consistent heuristic.

Marking Scheme: (3 marks) Proof is correct and easy to understand.

Solutions:

Let $h(x)$ denote the blocking heuristic function. We need to prove that $h(m) - h(n) \leq cost(m, n)$ for any two states m and n in the search graph.

Consider two states m and n. Without loss of generality, assume that $h(m) \geq h(n)$. By definition, $h(n)$ is the number of cars blocking the goal car plus one. Since

$h(m) \geq h(n)$, the number of cars blocking the goal car in stte m is greater than or equal to that of state n. For convenience, let $k = h(m) - h(n)$. k is the difference between the number of cards blokcing the goal car in the two states m and n.

By definition, $cost(m, n)$ is the number of steps we need to transfer state m to state n. Since $h(m) - h(n) = k$, we need to move each of the k cars for at least one step so that they do not block the goal car anymore. When moving each of the k cars, it is possible that we may have to move other cars that are blocking one of the k cars. Thus, the number of steps to transfer state m to state n must be greater than or equal to k.

Therefore, $h(m) - h(n) \leq cost(m, n)$ and the blocking heuristic is consistent.

(c) Design and implement an advanced heuristic of your own invention. Your advanced heuristic should be consistent and dominate the blocking heuristic.

Prove that your advanced heuristic is consistent and dominates the blocking heuristic.

Implement your advanced heuristic. Show that A* search with the advanced heuristic expands fewer nodes than A* search with the blocking heuristic on all the 40 provided puzzles.

Marking Scheme: (7 marks)

- (3 marks) Prove that your advanced heuristic is consistent.
- (2 marks) Prove that your advanced heuristic dominates the blocking heuristic.
- (2 marks) Show program output to support that the advanced heuristic dominates the blocking heuristic.

Solutions:

To derive the blocking heuristic, we relax the constraint that any car cannot overlap with any other car. For the relaxed problem, assume that the goal car must satisfy this constraint, but other cars do not need to satisfy this constraint. We can solve this relaxed problem optimally in two steps. (1) For any car directly blocking the goal car, move the car out of the steps of the goal car in one step. This move is allowed since these cars can overlap with other cars. (2) Move the goal car to the exit in one step. The cost of this optimal solution is the number of cards directly blocking the goal car plus one.

To derive an advanced heuristic that dominates the blocking heuristic, we can consider the same constraint. Assume that the goal car and any car directly blocking the goal car must satisfy the constraint. Other cars do not need to satisfy this constraint. This problem is less relaxed than the problem described above. It will allow us to derive an advanced heuristic that dominates the blocking heuristic.