Assignment One: Search
Due: 1:00pm, Wednesday 7 October 2015. Submit solution to handin.

Some of the following problems may be easier to do if you use the AIspace.org applets and/or the Python code.

This can be done in groups of size 1, 2 or 3. Working alone is not recommended. A group of size $n$ can choose any $n + 1$ questions out of questions 1 to 5. Every group should do question 6. All members of the group need to be able to explain the group’s answer. Please look at all of the questions, as the exam will assume that you have thought about all of the questions. Please post questions to the Piazza web site.

Use handin to submit your answers. You must submit a written response to the questions; it is this written response that will be marked. Use proper sentences in your answer. You should also submit your code and/or AIspace representations, in case the markers want to look at it.

Ask questions on Piazza. Feel free to answer them too.

Question One
You are working at a popular marina where a number of yachts are parked in berths. Each berth can either be vacant or have exactly one yacht parked in it. Any yacht can be parked in any berth. There is only one vacant berth in the marina.

Your marina has 10 berths numbered 1 to 10 and houses 9 yachts labeled y1 to y9. Table 1 shows the current state of the marina in which 9 is the initial vacant berth.

| y2 ▶ 1 | 10 ◄ y5 |
| y4 ▶ 2 | 9 |
| y3 ▶ 3 | 8 ◄ y7 |
| y1 ▶ 4 | 7 ◄ y8 |
| y9 ▶ 5 | 6 ◄ y6 |

Table 1: Current state of your marina

The yachts were originally assigned to the berth with the same number, y1 to berth 1, y2 to berth 2 etc. Your customers are clearly misbehaving! The marina manager has asked you to put each yacht back to its originally assigned berth.

You will only be able to move one boat at a time and you are not allowed to take yachts out of the marina. Also, because of some strange liability issues, you can only move boats from a berth $b$ to a vacant berth $b_v$ if one is a multiple of the other or if both are prime.

Examples of legal moves

- Yachts from berths 1, 3, 4, 5, 6, 7, 8 and 10 can be moved into berth 2.
- Yachts from all other berths can be moved into berth 1.
- Yachts from berths 1, 2 and 5 can be moved into berth 10.
- Yachts from berths 1, 2, 3 and 5 can be moved into berth 7.
**Your Problem**: You want to find a sequence of yacht-moves such that each yacht is back in its originally assigned berth.

(a) [6 marks] Represent the marina problem as a search problem.
   - [2 marks] How would you represent a node/state?
   - [1 marks] How many states are there?
   - [1 marks] What is the goal node?
   - [2 marks] What are the arcs? (How can they be specified so that given a node and an arc, the resulting state can be determined?)

(b) [4 marks] Write out the first two levels (counting the root as level 1) of the search tree. Also write down one subtree at level three. Only label the arcs; labeling the nodes would be too much work.

(c) [2 marks] What kind of uninformed search strategy would you use for this problem assuming that: all moves have the same cost, you want to minimize your work and your computer does not have much memory? Justify your answer.

(d) [2 marks] What is the max branching factor for this search problem?

(e) [2 marks bonus] Assuming that the length of the shortest path to the goal is $m$, what are the time and space complexity of the strategy you chose, as a function of $m$ only?

**Question Two**

Consider the grid world in the following figure:

![Grid World Diagram](image)

Assume that the robot can be in any of the white squares, and can do one step up, down, left, or right at each time. It cannot step into one of the black squares or outside of the boundary. The cost of a path is the number of steps in the path. At the squares marked with $C_i$ are coffee shops where, if the robot goes to the square, it will be given coffee. Suppose the robot starts at the square marked as $S$, without coffee, and must end up at the square marked $G$ carrying coffee.
(a) [2 marks] What is the state space? How many states are there?

(b) [2 marks] Give some depiction of the search space that makes the states explicit (either so that a human can understand it, or so that the (Python or AIspace) searching algorithms provided can use it). [For the human understandable representation, you can use whatever notation you like as long as you explain it (the simpler the better). Hint: You do not need to draw it as a graph; try to use the grid, but explain your notation; the arcs can be implicit as long as you explain what is going on. For the computer-based representations, you need to explain it so the marker can understand it.]

(c) [4 marks] Give an admissible heuristic. Your heuristic can take into account where the coffee shops are, but not where the blocked squares are. Hint: a useful concept is the Manhattan distance between two points \((x_1, y_1)\) and \((x_2, y_2)\) which is \(|x_1 - x_2| + |y_1 - y_2|\). [The trivial heuristic that is the same for all nodes will get you 1 mark out of 4.]

(d) [4 marks] Using your depiction of the search space given in part (b), number the nodes in the order they are expanded in an \(A^*\) search with multiple-path pruning. This should either be in the human-understandable form or as output of the program (make sure that you explain the output so that the TA can understand your answer). [Note that the provided Python code does not implement multiple-path pruning.]

(e) [2 marks] What is the path that your algorithm found? Would it have been different if there was loop detection instead of multiple-path pruning?

(f) [2 marks, bonus] How will your heuristic and \(A^*\) change if we assume that there is a hill going down towards the right so that steps to the right cost 0.5 and steps to the left cost 1.5, and steps up and down cost 1 each.

**Question Three**

For the Python implementation for \(A^*\) search:

(a) [6 marks] Implement multiple-path pruning. Compare multiple path pruning and the original algorithm (in terms of the paths found and the number of nodes expanded) for the cyclic delivery problem.

(b) [4 marks] Implement loop-detection. How does it compare in terms of the number of nodes expanded to multiple-path pruning for the cyclic delivery problem?

(c) [4 marks] Change the code so that it implements (i) best-first search and (ii) least-cost-first search. For each of these methods compare it to \(A^*\) in terms of the number of nodes expanded, and the path found.

(d) [2 marks, bonus] Make up a graph that highlights the difference between the methods compared in this question. [Imagine you wanted to explain how these were different to your peers and want a reasonably-simple example to highlight the differences.] You need to explain the example so another student can understand it, as well as submit the working representation of the graph.

You need to hand in a writeup that explains how the code is modified from the original as well as the code that runs.

**Question Four**

For each of the following, give a graph that is a tree (there is at most one arc into any node), contains at most 15 nodes, and has at most two arcs out of any node. Explain why the graph you give is a solution to the problem.
(a) [3 marks] Give a graph where depth-first search is much more efficient (expands fewer nodes) than breadth-first search.

(b) [3 marks] Give a graph where breadth-first search is much better than depth-first search.

(c) [3 marks] Give a graph where A* search is more efficient than either depth-first search or breadth-first search.

(d) [3 marks] Give a graph where depth-first search and breadth-first search are both more efficient than A* search.

(e) [2 marks] Give a graph where A* search is much better in the backwards direction than in the forward direction.

(f) [2 marks, bonus] Explain how Zeno’s paradox is (or could be) related to A* search. This should be written for your peers who have not read the textbook.

You must draw the graph and show the order of the neighbors (this is needed for the depth-first search). Either give the arc costs and heuristic function or state explicitly that you are drawing the graph to scale and are using Euclidean distance for the arc costs and the heuristic function.

**Question Five**

(a) [6 marks] Implement loop-detection for the Python implementation of branch-and-bound search. Your write-up needs to explain what modifications you made.

(b) [3 marks] When there are multiple paths with the same value, which one is chosen in A* search? If all nodes have the same value, which search method does this act like? Change the A* searcher so that it will act like a different search method when the values are equal. (Hint: this only requires a change in the Frontier class, and can be done with a minimal change to the program.)

(c) [5 marks] After the branch-and-bound search found a solution, Sam ran the search again, and noticed a different count of the number of nodes expanded. Sam hypothesized that this count was related to the number of nodes that an A* search would use (either expand or add to the frontier). Or maybe, Sam thought, the count for a number of nodes expanded when the bound is slightly above the length of the optimal path is related to how A* would work. Is there a relationship between these counts? Try to find the most specific statement that is true, and explain why it is true. [Sam also wondered about whether it might depend on the structure of the search space (e.g., if the graph is a tree).]

   Note that this an more of an open-ended question that is more like how research works. Play and find something that might be true, and then try to prove it (or find a counterexample). You can get full marks for giving a non-trivial statement that is always true and explaining why it is true. You can get 2 bonus marks for proving it.

**Question Six**

[2 marks per question] For each question, specify how long you spend on it, and what you learned. How was the work in the team allocated? Was the question reasonable? (This questions is worth marks, so please do it!)