Problem 1 Define “Intelligence”

No standard answer.

Problem 2 Towers of Hanoi

2.a i. We can represent each state as a three-tuple of lists. Each component of the tuple contains a list of indices of the discs that are on the respective peg – hence there are three components in the tuple, one for each peg from left to right. Also, the discs are numbered from 1 to n in the order of increasing size. In the list, the discs are listed from top to bottom.

   ii. The initial state can be represented as ([1, 2, 3, ..., n], [], []). The goal state can be represented as ([], [], [1, 2, 3, ..., n]).

   iii. The operators represent the movement of any one disc from the very top of a peg \( x \), to the very top of another peg, \( y \) such that \( x \neq y \). If a peg is empty, we cannot move anything from it to another peg. Also, we cannot move a disc from \( x \) to \( y \) if the index of the top disc of \( x \) exceeds the index of the top disc of \( y \). More formally, we have the following 6 possible operations at each state, of which only a subset are valid at any given state: (their respective notation is shown alongside) Move from 1 to 2 (12), move from 1 to 3 (13), move from 2 to 1 (21), move from 2 to 3 (23), move from 3 to 1 (31), move from 3 to 2 (32).

   iv. To derive the branching factor, we need to consider the maximum number of possible operators at any given state. Consider the case where all the pegs are non-empty. In that case, we can choose any of the 3 pegs to move the top disc from. For each of those cases, we could move the top disc to any of the other two pegs. However, not all of these operations are valid. Out of the three top discs, the smallest one can move to any of the other two pegs. The next smallest one can move to just one other peg. The largest of the three top discs cannot move to any other peg, since it would violate the invariant of the problem. Thus, at maximum, we can have 3 possible branches from a given state.

b-e. Shown (and explained) below. Every node in the graph represents a state - and the nodes are annotated with their respective states. They are also annotated with the numbers and letters from parts c, d and e, representing the different traversals. The edges are all two-way edges, and are labeled with tuples. In each of these tuples, the first component indicates the operator
Problem 3 Water Jugs

3.a  
   i. We can represent each state as a two-tuple of integers, where the first integer represents the amount of water currently in the larger jug, and the second integer represents the amount of water currently in the smaller jug. Note that according to the problem, if \((x, y)\) is a state, then \(x\) and \(y\) are integers and \(0 \leq x \leq 10\) and \(0 \leq y \leq 6\).

   ii. The initial state can be represented as \((0, 0)\), since both jugs are empty. The goal states can be represented as \((8, y)\), where \(0 \leq y \leq 6\) - i.e., the larger jug has 8 liters of water.

   iii. There are six possible operators/actions at each step. We could fill the larger jug till the top (represented by LF), fill the smaller jug till the top (represented by SF), pour from the larger to the smaller (LS), or from the smaller to the larger (SL), empty the larger jug (LE) or empty the smaller jug (SE). Note that we will only allow non-empty jugs to be emptied, or poured into other jugs.

   iv. To derive the branching factor, we need to consider the maximum number of possible operators at any given state. Note that at any given state, it must be the case that at least one jug is either full or empty, since every operator results in such a state. That being said, there are at maximum 4 possible valid operators for a given state. This is because the
operator also results in a state where at least one jug is either full or empty. This can also be seen from the diagram below.

3.b The diagram of the search space is shown below. Every state is represented as a tuple, as explained above. All the state transitions are represented as directed arrows, with the corresponding operators labeled along each edge (see above for the meaning of each operator). The two goal states, (8, 6) and (8, 0) are boxed.

3.c The shortest path to the goal state from the initial state is highlighted in yellow in the diagram below.

Problem 4 3D Integer Grid

4.a At each step $k$, values $x$, $y$, and $z$ can each be increased or decreased by 1, giving us a branching factor of 6.
4.b This problem is an example of an Octahedral Number. To solve this, we want to identify a pattern with the number of possible states. If the \( x \) coordinate is incremented by \( k \) steps, then there is 1 possible combination for \( y \) and \( z \): \((0, 0)\). If \( x \) is incremented by \( k - 1 \) steps, then there are 4 possible options for \( y \) and \( z \): \((0, 1), (1, 0), (0, -1), (-1, 0)\).

Following this series, if \( x \) is incremented by 0, we see that there are \((k + 1)^2\) options for \( y \) and \( z \). If we sum up the entire series, we get the equation: \( \sum_{i=0}^{k+1} i^2 \). Since the number of options is the same for when \( x \) is decremented, we take two times this sum and subtract the case when \( x = 0 \) so it is not double-counted:

\[
2 \left( \sum_{i=0}^{k+1} i^2 \right) - (k + 1)^2 = 2 \left[ \frac{(k+1)(k+2)(2k+3)}{6} \right] - (k + 1)^2 \tag{1}
\]

Recall from piazza the following equation:

\[
\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \tag{2}
\]

Using this, we can expand our equation to come up with the number of distinct states at step \( k \):

\[
2 \left[ \frac{(k+1)(k+2)(2k+3)}{6} \right] - (k^2 + 2k + 1) = \frac{2k^3 + 9k^2 + 13k + 6}{3} - \frac{3k^2 + 6k + 3}{3} = \frac{2}{3}k^3 + 2k^2 + \frac{7}{3}k + 1 \tag{3}
\]

As mentioned on Piazza, you could also have answered the question “How many distinct states are there that are \( k \) or fewer steps from the initial state?” This alternate problem is an example of a Centered Octahedral Number.

To solve this, we can look at the 2D case and expand it to 3D. If we draw out the 2D case, we note that at \( k \) steps away, there are exactly \( 4k \) nodes. We can sum this over all \( k \) steps to find the total number of distinct states that are \( k \) steps or fewer away (adding 1 for the initial state):

\[
1 + \sum_{i=1}^{k} 4i = 1 + 4 \sum_{i=1}^{k} i = 1 + 4 \sum_{i=1}^{k} i \tag{6}
\]

Recall from piazza the following equation:

\[
\sum_{i=0}^{n} i = \frac{n(n+1)}{2} \tag{7}
\]

Using this, we can solve for the 2-dimensional answer:

\[
1 + 4 \left[ \frac{k(k+1)}{2} \right] = 2k^2 + 2k + 1 \tag{8}
\]
Expanding into 3 dimensions, we can simply sum this equation over all values of \( z \) to find the number of unique nodes exactly \( k \) steps away from the initial state. Since \( z \) goes from \(-k\) to \( k\), we add the summation twice, and subtract the initial case so it is not counted twice:

\[
2 \left[ \sum_{z=0}^{k} (2z^2 + 2z + 1) \right] - (2k^2 + 2k + 1)
\]

(10)

\[
2 \left[ \sum_{z=0}^{k} z^2 \right] + 2 \left[ \sum_{z=0}^{k} z \right] + \left[ \sum_{z=0}^{k} 1 \right] - (2k^2 + 2k + 1)
\]

(11)

\[
2 \left[ \frac{k(k+1)(2k+1)}{6} \right] + 2 \left[ \frac{k(k+1)}{2} \right] + [k + 1] - (2k^2 + 2k + 1)
\]

(12)

\[
\frac{4}{3} k^3 + 2k^2 + \frac{8}{3} k + 1
\]

(13)

4.c Yes. Since only one digit can be changed by +1/-1 at a time, the minimum cost to reach the goal state is the sum across all dimensions of the number of steps each value is from the goal. In other words, \( h \) is the Manhattan Distance for \( n = 3 \), and since we only move horizontally/vertically and never diagonally, this represents the minimum cost to reach the goal state. And sum of absolute values is always greater than equal to zero; therefore, \( 0 \leq h(x, y, z) \leq h^*(x, y, z) \) is satisfied.

4.d Yes. Adding this restriction cannot decrease the minimum number of steps required to reach the goal state. The minimum number of steps to reach goal is either increased or remains the same. Since we proved in the previous part that \( h \) is admissible for the original problem, it will also be admissible for this added variation.

4.e No. Since \( h \) is dependent on the values being incremented or decremented by 1, being able to change \( x \) by 3 when \( x \) is odd can cause \( h \) to overestimate the actual cost to the goal for certain states. For example, if we have a goal state of \((4, 0, 0)\), then \( h(0, 0, 0) = 4 \). The first step will bring us from \((0, 0, 0)\) to \((1, 0, 0)\) with \( h(1, 0, 0) = 3 \). Since \( x \) is now odd, we can increment by 3, bringing us to the goal state in 1 step, which is less than the value from \( h \). Therefore, \( h \) is not admissible for this problem.

4.f Assuming using \( \text{argmin} \) to find the next best successor state, the heuristic from part (c) will still find an optimal solution for this non-negative variant. You can also simplify the function to remove the absolute values:

\[
f(x, y, z) = (a - x) + (b - y) + (c - z)
\]

\[
= (a + b + c) - (x + y + z)
\]

Another option is to use the euclidean distance function:

\[
f(x, y, z) = \sqrt{(x - a)^2 + (b - y)^2 + (c - z)^2}
\]

Using \( \text{argmax} \), you can negate any of the functions mentioned above.
Problem 5 Admissibility

5.a No, $h_1(s)$ can overestimate the cost.

5.b Yes, $0 \leq h_1(s) \leq h(s) \leq h^*(s)$.

5.c No, $h_1(s) \leq 0$.

5.d $a > 0$ and $0 \leq \frac{b}{a} \leq 1$. $a > 0$ so that $a \times g(s)$ remains positive. We transform the equation into $f(s) = a\left(g(s) + \frac{b}{a}h(s)\right)$. Let $f'(s) = g(s) + \frac{b}{a}h(s)$, then $f(s) = af'(s)$ where $a > 0$. A* search guaranteed to find an optimal solution by $f(s)$ if and only if $f'(s)$ has an admissible heuristic. Since $h(s)$ is admissible, then $0 \leq \frac{b}{a} \leq 1$. 
