Assignment 1

Zahin Mohammad

June 2020

1 Question 1

For the given problem, a state will be represented as a *tuple*, where the first element represents the amount of water in the 4 gallon jug, and the second element represents the amount of water in the 3 gallon jug.

The initial state will be represented as $(0, 0)$, indicating an empty 4 gallon jug and an empty 3 gallon jug.

The goal state will be represented as $(, 2)$ indicating that the 3 gallon jug has 2 gallons of water, and the 4 gallon jug has anywhere from 0-4 gallons of water.

At any state, the given actions are:

- Pour all water from the 4 gallon jug to the 3 gallon jug
- Pour all water from the 3 gallon jug to the 4 gallon jug
- Pour all water from the 4 gallon jug to the ground
- Pour all water from the 3 gallon jug to the ground
- Fill the 4 gallon jug with 4 gallons of water via the pump
- Fill the 3 gallon jug with 3 gallons of water via the pump

The condition for the actions is as follows:

- If jug x is pouring water into jug y then the water in jug $x > 0$
- If jug x is getting filled with water via the pump, then the water in jug $x <$ max capacity of jug x

2 Question 2

For the given problem, a state will be represented as 3 *arrays*, one per pole. The items in the array represent discs on a pole, with element 0 being the base of the pole (largest diameter), and the end of the array being the top most disc on the pole (smallest diameter). The discs are represented as their numeric diameter.

The initial state will be represented as $[x, y, z...], [], []$ indicating that the first pole has all the discs in descending order, and the other two poles are empty. Here x, y, z represent arbitrary disc diameters where the diameters follow as $x > y > z$.

The goal state will be represented as $[], [x, y, z], []$ or $[], [], [x, y, z]$ indicating that the first pole has no discs, and either the second or third pole has all the discs in descending order of disc diameter.

At any state, the given actions are:

- Move a disc from pole a to pole b where $a! = b$

The condition for the actions is as follows:

- If a disc is being moved from pole a to pole b, then there must exist at least 1 disc in pole a
- If a disc is being moved from pole a to pole b and b is not empty, then the disc at the top of pole b must be less than the disc at the top of pole a

3 Question 3

3.1 Breadth-First Search

In Table 1 the steps of the breadth first search algorithm are shown. The open-queue is a FIFO queue, with the next element being the left-most element in the array. By step 10, we are processing the goal state I. This gives us a final path of

$$A \to E \to I, \tag{1}$$

with a cost of 2.

Table 1: Steps for Breadth First Search

Step	Current	Goal	Open Queue	Closed Queue
1		I	$[A^0]$	[]
2	A^0	I	$[B^1, D^1, E^1]$	[]
3	B^1	I	$[D^1, E^1, G^2]$	$[A^0]$
4	D^1	I	$[E^1, G^2, C^2, F^2]$	$[A^0, B^1]$
5	E^1	I	$[G^2, C^2, F^2, H^2, I^2]$	$[A^0, B^1, D^1]$
6	G^2	I	$[C^2, F^2, H^2, I^2]$	$[A^0, B^1, D^1, E^1]$
7	C^2	I	$[F^2, H^2, I^2]$	$[A^0, B^1, D^1, E^1, G^2]$
8	F^2	I	$[H^2, I^2]$	$[A^0, B^1, D^1, E^1, G^2, C^2]$
9	H^2	I	$[I^2]$	$[A^0, B^1, D^1, E^1, G^2, C^2, F^2]$
10	I^2	I	[]	$[A^0, B^1, D^1, E^1, G^2, C^2, F^2, H^2]$

3.2 Depth-First Search

In Table 2 the steps of the depth first search algorithm are shown. The open-queue is a LIFO queue, with the next element being the left-most element in the array. By step 4, we are processing the goal state I. This gives us a final path of

$$A \to E \to I, \tag{2}$$

with a cost of 2.

Table 2: Steps for Depth First Search

Step	Current	Goal	Open Queue	Closed Queue
1		I	$[A^0]$	[]
2	A^0	I	$[E^1, D^1, B^1]$	[]
3	E^1	I	$[I^2, H^2, G^2, D^1, B^1]$	$[A^0]$
4	I^2	I	$[H^2, G^2, D^1, B^1]$	$[A^0, E^1]$

4 Question 4

NOTE: For this question, it is assumed that states can be repeated as it was not stated in the document.

4.1 Breadth-First Search

In Table 3 the steps of the breadth first search algorithm are shown. The open-queue is a FIFO queue, with the next element being the left-most element in the array. By step 14, we are processing the goal state G. This gives us a final path of

$$S \to A \to B \to H \to G, \tag{3}$$

with a cost of 12.

4.2 Depth-First Search

In Table 4 the steps of the depth first search algorithm are shown. The open-queue is a LIFO queue, with the next element being the left-most element in the array. By step 7, we are processing the goal state G. This gives us a final path of

$$S \to D \to F \to H \to G, \tag{4}$$

with a cost of 16.

Table 3: Steps for Breadth First Search

Step	Current	Goal	Open Queue	Closed Queue
1		G	$[S^0]$	[]
2	S^0	G	$[A^3, D^4]$	[]
3	A^3	G	$[D^4, B^7, D^8]$	$[S^0]$
4	D^4	G	$[B^7, D^8, E^6]$	$[S^0, A^3]$
5	B^7	G	$[D^8, E^6, C^{10}, H^{11}]$	$[S^0, A^3, D^3]$
6	D^8	G	$[E^6, C^{10}, H^{11}, E^{10}]$	$[S^0, A^3, D^3, B^7]$
7	E^6	G	$[C^{10}, H^{11}, E^{10}, B^{11}, C^{10}, F^{10}]$	$[S^0, A^3, D^3, B^7, D^8]$
8	C^{10}	G	$[H^{11}, E^{10}, B^{11}, C^{10}, F^{10}]$	$[S^0, A^3, D^3, B^7, D^8, E^6]$
9	H^{11}	G	$[E^{10}, B^{11}, C^{10}, F^{10}, G^{12}]$	$[S^0, A^3, D^3, B^7, D^8, E^6, C^{10}]$
10	E^{10}	G	$[B^{11}, C^{10}, F^{10}, G^{12}, B^{15}, C^{14}, F^{14}]$	$[S^0, A^3, D^3, B^7, D^8, E^6, C^{10}, H^{11}]$
11	B^{11}	G	$[C^{10}, F^{10}, G^{12}, B^{15}, C^{14}, F^{14}, C^{14}, H^{15}]$	$[S^0, A^3, D^3, B^7, D^8, E^6, C^{10}, H^{11}, E^{10}]$
12	C^{10}	G	$[F^{10}, G^{12}, B^{15}, C^{14}, F^{14}, C^{14}, H^{15}]$	$[S^0, A^3, D^3, B^7, D^8, E^6, C^{10}, H^{11}, E^{10}, B^{11}]$
13	F^{10}	G	$[G^{12}, B^{15}, C^{14}, F^{14}, C^{14}, H^{15}, H^{15}]$	$[S^0, A^3, D^3, B^7, D^8, E^6, C^{10}, H^{11}, E^{10}, B^{11}, C^{10}]$
14	G^{12}	G	$[B^{15}, C^{14}, F^{14}, C^{14}, H^{15}, H^{15}]$	$[S^0, A^3, D^3, B^7, D^8, E^6, C^{10}, H^{11}, E^{10}, B^{11}, C^{10}, F^{10}]$

Table 4: Steps for Depth First Search

Step	Current	Goal	Open Queue	Closed Queue
1		G	$[S^0]$	[]
2	S^0	G	$[D^4, A^3]$	[]
3	D^4	G	$[E^6, A^3]$	$[S^0]$
4	E^6	G	$[F^{10}, C^{10}, B^{11}, A^3]$	$[S^0, D^4]$
5	F^{10}	G	$[H^{15}, C^{10}, B^{11}, A^3]$	$[S^0, D^4, E^6]$
6	H^{15}	G	$[G^{16}, C^{10}, B^{11}, A^3]$	$[S^0, D^4, E^6, F^{10}]$
7	G^{16}	G	$[C^{10}, B^{11}, A^3]$	$[S^0, D^4, E^6, F^{10}, H^{15}]$

4.3 Uniform Cost Search

In Table 5 the steps of the uniform cost search algorithm are shown. By step 13, we are processing the goal state G. This gives us a final path of

$$S \to A \to B \to H \to G, \tag{5}$$

with a cost of 12.

Table 5: Steps for Uniform Cost Search

Step	Current	Goal	Open Queue	Closed Queue
1		G	$[S^0]$	[]
2	S^0	G	$[A^3, D^4]$	[]
3	A^3	G	$[D^4, B^7, D^8]$	$[S^0]$
4	D^4	G	$[E^6, B^7, D^8]$	$[S^0, A^3]$
5	E^6	G	$[B^7, D^8, C^{10}, F^{10}, B^{11},]$	$[S^0, A^3, D^4]$
6	B^7	G	$[D^8, C^{10}, C^{10}, F^{10}, H^{11}, B^{11}]$	$[S^0, A^3, D^4, E^6]$
7	D^8	G	$[C^{10}, C^{10}, E^{10}, F^{10}, H^{11}, B^{11}]$	$[S^0, A^3, D^4, E^6, B^7]$
8	C^{10}	G	$[C^{10}, E^{10}, F^{10}, H^{11}, B^{11}]$	$[S^0, A^3, D^4, E^6, B^7, D^8]$
9	C^{10}	G	$[E^{10}, F^{10}, H^{11}, B^{11}]$	$[S^0, A^3, D^4, E^6, B^7, D^8, C^{10}]$
10	E^{10}	G	$[F^{10}, H^{11}, B^{11}, C^{14}, F^{14}, B^{15}]$	$[S^0, A^3, D^4, E^6, B^7, D^8, C^{10}, C^{10}]$
11	F^{10}	G	$[H^{11}, B^{11}, C^{14}, F^{14}, B^{15}, H^{15}]$	$[S^0, A^3, D^4, E^6, B^7, D^8, C^{10}, C^{10}, E^{10}]$
12	H^{11}	G	$[G^{12}, B^{11}, C^{14}, F^{14}, B^{15}, H^{15}]$	$[S^0, A^3, D^4, E^6, B^7, D^8, C^{10}, C^{10}, E^{10}, F^{10}]$
13	G^{12}	G	$[G^{12}, B^{11}, C^{14}, F^{14}, B^{15}, H^{15}]$	$[S^0, A^3, D^4, E^6, B^7, D^8, C^{10}, C^{10}, E^{10}, F^{10}, H^{11}]$

5 Question 5

5.1 Uniform Cost Search

In Table 6 the steps of the uniform cost search algorithm are shown. By step 11, we are processing the goal state 7. This gives us a final path of

$$1 \to 8 \to 10 \to 9 \to 7, \tag{6}$$

with a cost of 100.

Table 6: Steps for Uniform Cost Search

Step	Current	Goal	Open Queue	Closed Queue
1		7	$[1^{0}]$	[]
2	1^{0}	7	$[5^{5}, 8^{24}]$	[]
3	5^{5}	7	$[8^{24}, 6^{40}]$	$[1^{0}]$
4	8^{24}	7	$[10^{39}, 6^{40}, 3^{47}]$	$[1^{0}, 5^{5}]$
5	10^{39}	7	$[6^{40}, 3^{47}, 9^{65}]$	$[1^{0}, 5^{5}, 8^{24}]$
6	6^{40}	7	$[3^{47}, 9^{65}, 2^{78}]$	$[1^{0}, 5^{5}, 8^{24}, 10^{39}]$
7	3^{47}	7	$[4^{54}, 9^{65}, 2^{78}]$	$[1^{0}, 5^{5}, 8^{24}, 10^{39}, 6^{40}]$
8	4^{54}	7	$[9^{65}, 2^{78}]$	$[1^{0}, 5^{5}, 8^{24}, 10^{39}, 6^{40}, 3^{47}]$
9	9^{65}	7	$[2^{78}, 7^{100}]$	$[1^{0}, 5^{5}, 8^{24}, 10^{39}, 6^{40}, 3^{47}, 4^{54}]$
10	2^{78}	7	$[7^{100}]$	$[1^{0}, 5^{5}, 8^{24}, 10^{39}, 6^{40}, 3^{47}, 4^{54}, 9^{65}]$
11	7^{100}	7	[]	$[1^{0}, 5^{5}, 8^{24}, 10^{39}, 6^{40}, 3^{47}, 4^{54}, 9^{65}, 2^{78}]$

5.2 Greedy Best First Search

In Table 7 the steps of the greedy best first search algorithm are shown. In the table, a state is represented as $state^{h(n),f(n)}$, where $h(n)$ is the heuristic, and $f(n)$ is the cost to get to that state. By step 7, we are processing the goal state 7. This gives us a final path of

$$1 \to 8 \to 3 \to 4 \to 9 \to 7, \tag{7}$$

with a cost of 107.

Table 7: Steps for Greedy Best First Search

Step	Current	Goal	Open Queue	Closed Queue
1		7	$[1^{78,0}]$	[]
2	$1^{78,0}$	7	$[8^{60,24}, 5^{75,5}]$	[]
3	$8^{60,24}$	7	$[3^{37,47}, 10^{57,39}, 5^{75,5}]$	$[1^{78,0}]$
4	$3^{37,47}$	7	$[4^{30,77}, 10^{57,39}, 5^{75,5}]$	$[1^{78,0}, 8^{60,24}]$
5	$4^{30,54}$	7	$[9^{35,72}, 10^{57,39}, 5^{75,5}]$	$[1^{78,0}, 8^{60,24}, 3^{37,47}]$
6	$9^{35,72}$	7	$[7^{0,107}, 2^{32,98}, 10^{57,39}, 6^{60,98}, 5^{75,5}]$	$[1^{78,0}, 8^{60,24}, 3^{37,47}, 4^{30,54}]$
7	$7^{0,107}$	7	$[2^{32,98}, 10^{57,39}, 6^{60,98}, 5^{75,5}]$	$[1^{78,0}, 8^{60,24}, 3^{37,47}, 4^{30,54}, 9^{35,72}]$

5.3 A* Search

In Table 8 the steps of the A* search algorithm are shown. In the table, a state is represented as $state^{h(n),f(n)}$, where $h(n)$ is the heuristic + the cost get to the state, and $f(n)$ is the cost to get to that state. By step 11, we are processing the goal state 7. This gives us a final path of

$$1 \rightarrow 8 \rightarrow 10 \rightarrow 9 \rightarrow 7, \tag{8}$$

with a cost of 100.

Table 8: Steps for Greedy Best First Search

Step	Current Node	Goal Node	Open Queue	Closed Queue
1		7	$[1^{78,0}]$	[]
2	$1^{78,0}$	7	$[5^{80,5}, 8^{84,24},]$	[]
3	$5^{80,5}$	7	$[8^{84,24}, 6^{100,40}]$	$[1^{78,0}]$
4	$8^{84,24}$	7	$[3^{84,47}, 10^{96,39}, 6^{100,40}]$	$[1^{78,0}, 5^{80,5}]$
6	$3^{84,47}$	7	$[4^{84,54}, 10^{96,39}, 6^{100,40}]$	$[1^{78,0}, 5^{80,5}, 8^{84,24}]$
7	$4^{84,54}$	7	$[10^{96,39}, 6^{100,40}, 9^{107,72}]$	$[1^{78,0}, 5^{80,5}, 8^{84,24}, 3^{84,47}]$
8	$10^{96,39}$	7	$[6^{100,40}, 9^{100,65}]$	$[1^{78,0}, 5^{80,5}, 8^{84,24}, 3^{84,47}, 4^{84,54}]$
9	$6^{100,40}$	7	$[9^{100,65}, 2^{110,78}]$	$[1^{78,0}, 5^{80,5}, 8^{84,24}, 3^{84,47}, 4^{84,54}, 10^{96,39}]$
10	$9^{100,65}$	7	$[7^{100,100}, 2^{110,78}]$	$[1^{78,0}, 5^{80,5}, 8^{84,24}, 3^{84,47}, 4^{84,54}, 10^{96,39}, 6^{100,40}]$
11	$7^{100,100}$	7	$[2^{110,78}]$	$[1^{78,0}, 5^{80,5}, 8^{84,24}, 3^{84,47}, 4^{84,54}, 10^{96,39}, 6^{100,40}, 9^{100,65}]$

6 Question 6

For all the search techniques in this question, the output path can be seen in a colored graph from the console output of the python script *q6.py*.

6.1 Breadth First Search

In Table 9 the results of the breadth first search are shown. The complete path of the search is available in file $q6 - output.txt$.

The breadth first search is implemented in an iterative fashion by maintaining a FIFO queue. States are expanded in the order of UP, DOWN, LEFT, RIGHT,

and appended to the queue in that order. In every iteration, the front first element in the queue is removed and expanded. This repeats until there are no states left in the stack. A *hashmap* is used to store a mapping from child to parent, and once the goal state is found, this mapping of child to parent is traversed from goal state to initial state, which becomes the final path.

Table 9: Search Summary for Breadth First Search

Initial State	Goal State	Visited States	Expanded States	Path Cost
$(0, 0)$	$(24, 24)$	449	450	49
$(2, 11)$	$(2, 21)$	269	252	21
$(2, 11)$	$(23, 19)$	386	378	30

6.2 Depth First Search

In Table 10 the results of the depth first search are shown. The complete path of the search is available in file $q6 - output.txt$.

The depth first search is implemented in the same way as breadth first search except for a single key difference. In each iteration, instead of choosing the first element in the queue, depth first search chooses the last element in the queue. This turns the queue into a LIFO queue. The same method for expanding nodes, and deriving the final path is used as in the breadth first search case.

Table 10: Search Summary for Depth First Search

Initial State	Goal State	Visited States	Expanded States	Path Cost
$(0, 0)$	$(24, 24)$	426	360	95
$(2, 11)$	$(2, 21)$	312	221	123
$(2, 11)$	$(23, 19)$	161	92	90

6.3 A* Search

In Table 11 the results of the A* search are shown. The complete path of the search is available in file $q6 - output.txt$.

The heuristic used in this search is $\Delta x^2 + \Delta y^2$, a value proportional to the euclidean distance. As the goal is to get from state x to state y with minimum cost, it is reasonable to assume that points closer to y will likely have a lower path cost. The actual euclidean distance was not used as the value was only ever used to maintain a priority queue.

The a* search is implemented in the same way as breadth first search except for two key differences. In each iteration, instead of choosing the first element in the queue, a* search chooses the last element in the queue. Additionally, after all children have been added to the queue, the queue is sorted in descending order based on the heuristic value of each state. This ensures that in each iteration, the state with the smallest heuristic value is explored. The same method for expanding nodes, and deriving the final path is used as in the breadth first search case.

Table 11: Search Summary for A* Search

Initial State	Goal State	Visited States	Expanded States	Path Cost
$(0, 0)$	$(24, 24)$	91	50	49
$(2, 11)$	$(2, 21)$	51	29	21
$(2, 11)$	$(23, 19)$	75	34	32