Several problems of interest can be modeled as a state-space search problem. This is how it is done: choose a problem, determine what constitutes a STATE (a written representation of the state-of-existence), identify the START STATE and the GOAL STATE(S), identify the operations/actions/moves that cause a STATE to change, then write down the STATE TRANSITIONS as a function (or program module) that maps a STATE to a set of NEIGHBORING STATES, a.k.a. MoveGen or neighborhood function.

In state space search a solution is found by exploring the state space with help of neighbourhood function: begin at the start state and keep expanding until a goal state is found.

State spaces are used to represent two kinds of problems. In configuration problems the task is to find a goal state that satisfies some properties. In planning problems the task is to find a path to a goal state. The sequence of moves in the path is a plan.

State spaces have properties:

- **Extent**: state spaces may be finite or infinite.
- **Exponential Growth**: finite state spaces may be very very large — exponential and beyond. And the search space associated with a search algorithm may increase exponentially with depth.
- **Branching Factor**: may be constant or bounded or finite or large-and-finite.
- **Reversible**: some state spaces are reversible, i.e., every STATE TRANSITION is REVERSIBLE. Most real world problems do not have this property, but have regions that are reversible, and regions that are not, and in sum total it is not reversible.
- **Connectedness**: All kinds of connectivity from graph theory apply here. The whole state space may be completely connected, or may have several connected components which are mutually disjoint. A search algorithm can only explore the connected component in which the start node lies. The 8-puzzle, for example, has two disjoint connected components, and the Rubik's cube has twelve.
- **Metric Spaces**: States (and/or transitions) may provide a metric (say Euclidean distance, Manhattan distance, etc.) as a measure of fitness or direction or distance to a goal state.

**Water jug puzzle**: A seven litre jug is filled with water, you are required to divide it into 1 + 5 + 1 litres. You may use two empty jugs of size 5 and 2 litres for this purpose. There is no other way of measuring water except by the size of the jugs, i.e., one can either empty a jug into another or fill another jug to its brim.

**ATTENTION**: Answers to Q1 thru Q7 are integers. DO NOT include spaces, tabs, periods or non-numeric characters in the text box.
1) Starting from (7,0,0), what is the least number of transfers/moves required to reach (1,5,1)?

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: Numeric) 6

2) Starting from (1,5,1), what is the least number of transfers/moves required to reach (7,0,0)?

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: Numeric) 2

3) How many paths from (1,5,1) to (7,0,0) have the minimum length found in question 2?

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: Numeric) 2

4) What is the size of the state-space (number of states) in 7-5-2 water jug puzzle?

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: Numeric) 14
(Type: Numeric) 18

5) Starting from (7,0,0), what is the least number of transfers required to measure 6 litres?

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: Numeric) 5
(Type: Numeric) 4

6) Starting from (7,0,0), what is the least number of transfers required to measure 4 litres?

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: Numeric) 3
(Type: Numeric) 2

7) Starting from (7,0,0), what is the least number of transfers required to measure 3 litres?

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: Numeric) 2

8) Which of the following is true about the state-space of water jug puzzle?

- [ ] State-space is reversible
- [ ] Every state is reachable from every other state
- [ ] All seven volumes from 1L to 7L are measurable
- [ ] There is at least one state that has no outgoing edge
- [ ] There is at least one state that has no incoming edge

No, the answer is incorrect.
Score: 0
Accepted Answers:
Every state is reachable from every other state
All seven volumes from 1L to 7L are measurable
9) Four state spaces and one MoveGen is shown below.  

Here (as well as in all state space diagrams in this assignment), to reduce clutter, two-way edges are shown without arrowheads and one-way edges are shown with arrowheads.

![State Space Diagrams]

**MoveGen (N)**

<table>
<thead>
<tr>
<th>S</th>
<th>Move Gen (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>[A, B]</td>
</tr>
<tr>
<td>B</td>
<td>[C, D, G, S]</td>
</tr>
<tr>
<td>C</td>
<td>[A, B]</td>
</tr>
<tr>
<td>D</td>
<td>[A, B, G, S]</td>
</tr>
<tr>
<td>G</td>
<td>[A, B, C, D]</td>
</tr>
</tbody>
</table>

Mark the correct options.

- The MoveGen describes State Space 1
- The MoveGen describes State Space 2
- The MoveGen describes State Space 3
- The MoveGen describes State Space 4
- None of these.

No, the answer is incorrect.

Score: 0

Accepted Answers:

The MoveGen describes State Space 2
The MoveGen describes State Space 3

10) A state space and four MoveGen functions are given below

![State Space Diagram]

**MoveGen-1 (N)**

<table>
<thead>
<tr>
<th>N</th>
<th>MoveGen-1 (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>[A, B]</td>
</tr>
<tr>
<td>A</td>
<td>[C, G, S]</td>
</tr>
<tr>
<td>B</td>
<td>[C, D, G, S]</td>
</tr>
<tr>
<td>C</td>
<td>[A, B]</td>
</tr>
<tr>
<td>D</td>
<td>[A, B, G, S]</td>
</tr>
<tr>
<td>G</td>
<td>[A, B, C, D]</td>
</tr>
</tbody>
</table>

**MoveGen-2 (N)**

<table>
<thead>
<tr>
<th>N</th>
<th>MoveGen-2 (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>[B, A]</td>
</tr>
<tr>
<td>A</td>
<td>[C, S, D]</td>
</tr>
<tr>
<td>B</td>
<td>[G, B, C]</td>
</tr>
<tr>
<td>C</td>
<td>[A]</td>
</tr>
<tr>
<td>D</td>
<td>[A, B]</td>
</tr>
<tr>
<td>G</td>
<td>[A, B]</td>
</tr>
</tbody>
</table>

**MoveGen-3 (N)**

<table>
<thead>
<tr>
<th>N</th>
<th>MoveGen-3 (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>[B]</td>
</tr>
<tr>
<td>A</td>
<td>[C, D, G, S]</td>
</tr>
<tr>
<td>B</td>
<td>[C, G, S]</td>
</tr>
<tr>
<td>C</td>
<td>[A, B]</td>
</tr>
<tr>
<td>D</td>
<td>[A, B, G, S]</td>
</tr>
<tr>
<td>G</td>
<td>[A, B, C, G]</td>
</tr>
</tbody>
</table>

**MoveGen-4 (N)**

<table>
<thead>
<tr>
<th>N</th>
<th>MoveGen-4 (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>[A, B]</td>
</tr>
<tr>
<td>A</td>
<td>[B, A]</td>
</tr>
<tr>
<td>B</td>
<td>[A]</td>
</tr>
<tr>
<td>C</td>
<td>[A, B]</td>
</tr>
<tr>
<td>D</td>
<td>[A, B, C, D]</td>
</tr>
<tr>
<td>G</td>
<td>[A, B]</td>
</tr>
</tbody>
</table>

Mark the correct options.

- The state space is described by MoveGen-1
- The state space is described by MoveGen-2
- The state space is described by MoveGen-3
- The state space is described by MoveGen-4
- None of these
BEGIN GROUP: Q11 – Q15

Use the given DFS and BFS algorithms to answer questions in this group. Observe that in these algorithms a new node is added to OPEN only if it is not present in either OPEN or CLOSED.

DFS

1. OPEN ← {S, null} : []
2. CLOSED ← empty list
3. while OPEN is not empty
4. nodePair ← head OPEN
5. (N, _) ← nodePair
6. if GOALTEST(N) = true
7. return RECONSTRUCTPATH(nodePair, CLOSED)
8. else CLOSED ← nodePair : CLOSED
9. children ← MOVEGEN(N)
10. newNodes ← REMOVE Thương (children, OPEN, CLOSED)
11. newPairs ← MAKEPAIRS(newNodes, N)
12. OPEN ← newPairs ++ (tail OPEN)
13. return empty list

BFS

1. OPEN ← {S, null} : []
2. CLOSED ← empty list
3. while OPEN is not empty
4. nodePair ← head OPEN
5. (N, _) ← nodePair
6. if GOALTEST(N) = true
7. return RECONSTRUCTPATH(nodePair, CLOSED)
8. else CLOSED ← nodePair : CLOSED
9. children ← MOVEGEN(N)
10. newNodes ← REMOVE Thương (children, OPEN, CLOSED)
11. newPairs ← MAKEPAIRS(newNodes, N)
12. OPEN ← (tail OPEN) ++ newPairs
13. return empty list

A MoveGen and its graphical representation is given below. Here S is the start node and G is the goal node. Now, run DFS and separately BFS until the goal is found, then answer the questions in this group.

Note: The given DFS and BFS algorithms maintain node-pairs in OPEN and CLOSED lists. Therefore, when typing the contents of OPEN or CLOSED list in the textbox, type the list as comma separated node-pairs (without using spaces, tabs, period or other symbols), for example: (A,S),(B,S),(S,null)

Note: In your answer the order of nodePairs in OPEN/CLOSED should be as constructed, i.e., the rightmost entry is the first nodePair added, and the leftmost is the last. This is consistent with how the Cons operator "::" constructs lists.

Note: If OPEN or CLOSED list is empty then enter NA in the textbox.

11. When DFS finds G, the CLOSED list is

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: String) (D,A),(C,A),(A,S),(S,null)

12. When DFS finds G, the OPEN list is

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: String) (G,D),(B,S)

13. When BFS finds G, the CLOSED list is

...
Artificial Intelligence Search Methods For Problem Solving - - Unit 5 - Week 2

14) When BFS finds G, the OPEN list is _____

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: String) (D,A), (C,A), (B,S), (A,S), (S,null)

15) Based on your responses to Q11-14, which of the following statements are true?

- When DFS finds G, B is in CLOSED
- When DFS finds G, B is still in OPEN
- When BFS finds G, B is in CLOSED
- When BFS finds G, B is still in OPEN

No, the answer is incorrect.
Score: 0
Accepted Answers:
When DFS finds G, B is still in OPEN
When BFS finds G, B is in CLOSED

END GROUP: Q11 – Q15

BEGIN GROUP: Q16 – Q20

Consider two variations of DEPTH FIRST ITERATIVE DEEPENING (DFID) where DFID-1 opens only new nodes (nodes not in OPEN and CLOSED are opened), and DFID-2 allows CLOSED nodes to be reopened and does not reopen the OPEN nodes. Run the two variations on the following state space which has S as the start node and G as the goal node.

16) What is the path found by DFID-1?

In the textbox, enter the path as a comma separated list starting from node S, enter only the node labels, do not enter node pairs, for example: S,A,B,C (and DO NOT use spaces, tabs, period or other characters in the path).

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: String) S,A,C,E,G

17) What is the path found by DFID-2?

In the textbox, enter the path as a comma separated list starting from node S, enter only the node labels, do not enter node pairs, for example: S,A,B,C (and DO NOT use spaces, tabs, period or other characters in the path).

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: String) S,B,D,G
18) For depth bound 3, DFID-1 invokes GoalTest ______ times

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: Numeric) 6
(Type: Numeric) 15

19) For depth bound 3, DFID-2 invokes GoalTest ______ times

No, the answer is incorrect.
Score: 0
Accepted Answers:
(Type: Numeric) 16
(Type: Numeric) 15
(Type: Numeric) 26
(Type: Numeric) 27

20) Select the correct statements

☐ If there is a path to the goal, DFID-1 always finds it
☐ If there is a path to the goal, DFID-1 returns the shortest path
☐ If there is a path to the goal, DFID-2 always finds it
☐ If there is a path to the goal, DFID-2 returns the shortest path.
☐ None of these

No, the answer is incorrect.
Score: 0
Accepted Answers:
If there is a path to the goal, DFID-1 always finds it
If there is a path to the goal, DFID-2 always finds it
If there is a path to the goal, DFID-2 returns the shortest path.