Formulate problem for 8-Puzzle.

Initial state: 

```
3 7 1
4 5 8
2 6
```

Actions - move blank U, D, L, R

State space - all possible configurations of 8 tiles and 1 blank

Transition model:

```
3 7 1
4 5 8
2 6
```

Goal: 

```
1 2
3 4 5
6 7 8
```

Cost: 1 (for each action)
Given all of this information, how do we search for a solution?

Start at initial state, continuously try all valid actions, get to next state until goal.

Terminology:
expand - apply each legal action to current state to generate new states.

Visualize with tree:

```
  3 7 1
  4 5 8
  2 6
  B      C
     /\   /\               \      \      \  
    D   A   E A D G A     B     A
  3 7 1
  4 5
  2 6
```

Search Tree

tree - state space
node - state
edge - action

Today "uniformed search" most basic search.
Tree Search - Idea:
1. Start at initial state
2. If current state is not goal:
   3. Expand current state
   4. Place successors in a list: frontier
   5. Pick a state from the frontier
   6. Check if it's the goal:
      if yes \( \Rightarrow \) Done
      if no \( \Rightarrow \) Repeat from 2

Implementation: based pseudocode:

TreeSearch (Problem)

\[
\text{Explored} = \emptyset
\]
\[
\text{Frontier} = \text{initial node}\}
\]

while Frontier not empty:

\[
S = \text{remove from Frontier}(\text{which node}\}
\] (add \( S \) to Explored)

if \( S \) is goal:

return solution found!

Let \( S \) path from initial to goal

else

expand \( S \)

for each resulting node \( n \):

\[
\text{if } n \text{ not in Frontier and not in Explored}
\]

add \( n \) to Frontier
```
// after while
return no solution!

Assume always remove from front of Frontier
(and add to end)

Frontier       S
F  A 3       A
F  B  C D E
F  C D E A F

Make more efficient?
Keep track of already visited states and do not add to frontier
⇒ Explored List

Any other nodes to ignore? Those already in frontier!
⇒ ADD TO CODE ⇒
```

```
ex

A
B C D
A E C A E F

Frontier      Explored       S
F  A E (2)    F  B 3 (1)    A 3
F  B c D E (6) F  A 3 (4)  B 6
F  C D E A F (5) F  A B 3 (7) C 9

not adding A (in Exp)
or C (in Frontier)
```
"Assumed always removing from front of Frontier." \( \Rightarrow \) **FIFO**

Frontier is a Queue

Always removing shallowest node

\( \Rightarrow \) **BFS**.

Could also remove from back of Frontier.

Frontier is a Stack

Always removing deepest node

\( \Rightarrow \) **DFS**.

**DFS**

<table>
<thead>
<tr>
<th>Frontier</th>
<th>Exp</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S )</td>
<td>( A )( B )( C )</td>
<td>( E ) ( D )</td>
</tr>
<tr>
<td>( 0 )</td>
<td>( 1 ) ( 2 )</td>
<td>( 3 ) ( 4 ) ( 5 )</td>
</tr>
</tbody>
</table>
Question: Why do we care about space?  
# of states may be infinite (or very large)

Activity: Analyze these 4 for BFS + DFS in terms of:
- b: branching factor (assume finite) (# actions)
- d: depth of (shallowest) goal
- m: maximum tree depth (may be ∞)

BFS:
(i) complete ✓ (as long as # actions, b, is finite).
   What if search space/depth is infinite?
   Still complete since goal is at finite depth.

(ii) time?
Suppose b = 2

Roughly:
# nodes generated = 1 + b + b^2 + b^3 + ... + b^d = O(b^d)