Tutorial for week 4

1. The 8-puzzle involves a 3 by 3 grid, with eight small squares that can be moved around. Actions can be designated by sliding a square into the vacant space, one of `Up`,`Down`,`Left`,`Right`.

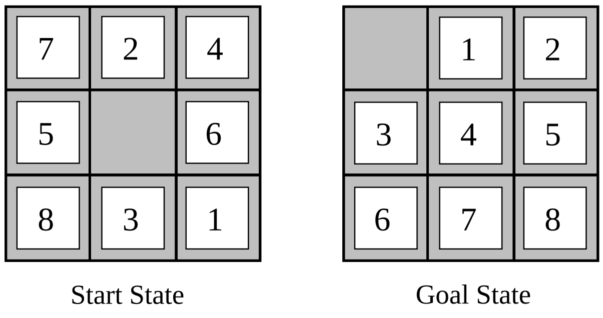

Figure 1: 8-puzzle

There are versions of the same game on $n \times n$ grids.

(a) Consider the three puzzle, played on a two by two board. How many states are there for this problem? What is the branching factor b?

(b) Sketch the state graph, indicating the actions between states as directed edges (lines with an arrow) between the states. (Just draw enough of the graph so that the general structure is made clear.)

(c) Now draw the search tree for depth limited search, looking as far as depth 2 (i.e. two actions away from the initial state).

(d) Indicate how an iterative deepening search searches the nodes of this tree down to this level, by numbering successive nodes visited.

(e) What is the problem with using depth-first search on the full search tree in this example?

(f) Depth first search can be used if we take account of repeated search, and do graph search. Depth-first tree search for a target state is given below.

The Expand function returns all the nodes that can be reached by a single action from the given node. This version does not keep track of the actions used, so only checks if there is a solution.

Modify this procedure to avoid the problem of repeated states.

```
to do DFS on (problem,fringe):
   set fringe = [ InitialState ]
   loop do
      if fringe = [] then return failure and halt
      set S1 = first of fringe
      set fringe = rest of fringe
      if S1 = target then return success and halt
      set NewStates = Expand(S1)
      set fringe = append( NewStates, fringe )
```

2. A pseudo-code version of breadth-first search is as follows:

```
to do BFS on (problem,fringe):
   set fringe = [ InitialState ]
   loop do
      if fringe = [] then return failure and halt
      set S1 = first of fringe
      set fringe = rest of fringe
      if S1 = target then return success and halt
      set NewStates = Expand(S1)
      set fringe = append( fringe, NewStates )
```

In the lecture, it was claimed that the space (number of nodes in memory) used by breadth-first search is

$$b + b^2 + \cdots + b^d + (b^{d+1} - b) = O(b^{d+1}).$$

Explain why this is the right worst-case analysis. (b is the branching factor, and d is the depth at which the first solution is encountered).