CS151 - Written Problem 1
Solutions

1. Sudoku
(from http://www-nlp.stanford.edu/ grenager/cs121//handouts/hw1.pdf)

   Consider the popular game Sudoku, in which one tries to fill a 9 x 9 grid of squares with numbers subject to some constraints:

   - every row must contain all of the digits 1,2, ..., 9
   - every column must contain all of the digits 1,2, ..., 9
   - each of the 9 different 3 x 3 boxes (look online if you don't know what I'm talking about :) must also contain all of the digits 1, . . . , 9

   A game is specified by filling in some of the boxes with numbers (in our case M). Each game is guaranteed to have a single solution, that is, there is only one assignment to the empty squares which satisfies all the constraints. For the purposes of this homework, use $n_{i,j}$ to refer to the number in row i, column j of the grid.

   (a) Formalize this problem as an incremental search problem. What are the start state, actions, goal test, and edge costs?

   There are many ways to formalize this problem. Here's one:

   i. **start state**: The starting board.
   ii. **actions**: Fill in the next empty square in reading order (i.e. left to right, top to bottom) with a "valid" number, i.e. one that meets the constraints above.
   iii. **goal test**: All the squares are filled in and all the constraints are met.

iv. **cost**: 1. In this case, each action fills in an empty square and we're only interested in finding a solution, so the cost doesn't matter.

(b) What is the branching factor, solution depth, and maximum depth of the search space? What is the size of the state space?

For the above formulation, the branching factor is 9, the solution depth is the number of blank squares in the starting configuration (lets call this number $s = 81 - M$), and the size of the state space is 9^s.

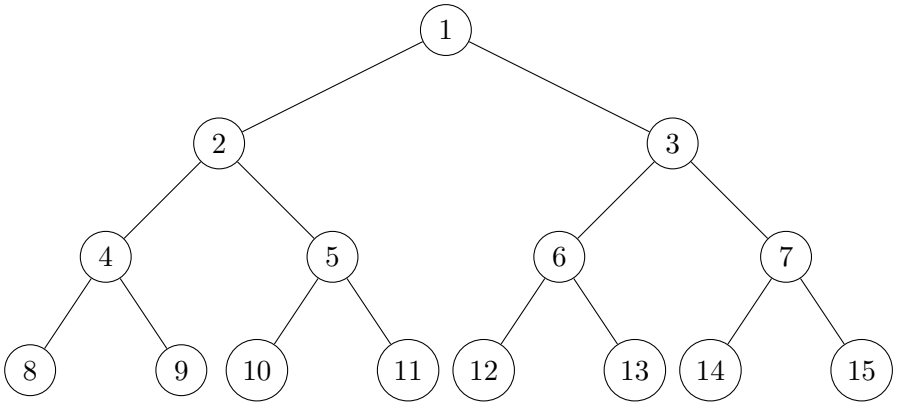
(c) Assuming we don't use a heuristic, which of the following would you recommend for solving the incremental search formulation of this problem: DFS, BFS, or Iterative Deepening (ID)? Why?

We should use DFS since we know that we have to go a fixed depth (i.e., s). The time and space complexity of DFS is just the depth of the search, $O(s)$. BFS has strictly worse space complexity (and the same time complexity, in the worst case). It doesnt make sense to use iterative deepening either because you know you have to get to depth s to solve the problem.

(d) Assuming we use the incremental search formulation, is heuristic search possible? If so, provide a heuristic. If not, why not?

There are no good heuristics for this problem that estimate how far we are from the solution because all states at depth d are equally far from the solution. (However, there may be heuristics we could choose that would help us decide which path is the most promising to explorei.e., help us order the nodes on the fringe).

2. Exercise 3.15 (parts a + b)



- BFS: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
- DL-DFS: 1, 2, 4, 8, 9, 5, 10, 11
- IDS: 1; 1, 2, 3; 1, 2, 4, 5, 3, 6, 7; 1, 2, 4, 8, 9, 5, 10, 11

3. Exercise 3.18

   "the twig", that is if we have a linear chain of nodes, each with just a single child. DFS will search to the bottom of this in $O(n)$ time, however, IDS will result in a run-time of: $1 + 2 + ... + n$ which is $O(n^2)$.

4. Exercise 3.21

   a. True. If we let the cost of each edge be 1, then $g(n)$ for uniform-cost search will be the depth of the node. Therefore, nodes will get visited based on their depth, with lower depths first, which is the same as BFS.

   b. True. Let $h(n) = depth(n)$. Best-first search will visit nodes with preference to deeper nodes.

   c. True. Let $h(n) = 0$, since $f(n) = g(n) + h(n)$, then $f(n) = g(n)$, which is uniform-cost search.

5. Exercise 3.23 (if you want more practice with A*)

```
 Path=Lugoj:
n=Mehadia:  g(n) = 70, h(n) = 241, f(n) = 311
n = Timisoara:  g(n) = 111, h(n) = 329, f(n) = 440
Fringe:  Mehadia (311), Timisoara (440)
```

```
Path=Lujog -> Mehadia:
n = Lujog:  g(n) = 140, h(n) = 244, h(n) = 384
n = Drobeta:  g(n) = 145, h(n) = 242, f(n) = 387
Fringe:  Timisoara (440), Lujog (384), Drobeta (387)

Path = Lugoj -> Mehadia -> Lugoj:
n = Mehadia:  g(n) = 210, h(n) = 241, f(n) = 451;
n = Timisoara:  g(n) = 251, h(n) = 329, f(n) = 580
Fringe:  Timisoara (440), Drobeta (387), Timisoara (580), Mehadia
(451)

Path = Lugoj -> Mehadia -> Drobeta
n = Mehadia:  g(n) =220, h(n) = 241, f(n) = 461
n = Craiova:  g(n) = 265, h(n) = 160, f(n) = 425
Fringe:  Timisoara (440), Timisoara (580), Mehadia (451), Mehadia
(461), Craiova (425)

Path = Lugoj ->Mehadia -> Drobeta -> Craiova
n = Pitesti:  g(n) = 403, h(n) = 100, f(n) = 503
n = RV: g(n) = 411, h(n) = 193, f(n) = 604
n = Drobeta:  g(n) = 385, h(n) = 242, f(n) = 627
Fringe:  Timisoara (440), Timisoara (580), Mehadia (451), Mehadia
(461), Pitesti (503), RV (604), Drobeta (527)

Path = Lugoj -> Timisoara
n = Lugoj:  g(n) = 222, h(n) = 244, f(n) = 466
n = Arad:  g(n) = 229, h(n) = 366, f(n) = 595
Fringe:  Timisoara (580), Mehadia (451) Mehadia (461), Pitesti
(503), Drobeta (627), Lugoj (466), Arad (595), RV (604)

Path = Lugoj -> Mehadia -> Lugoj -> Mehadia
n = Lugoj:  g(n) = 280, h(n) = 244, f(n) = 524
n = Drobeta:  g(n) = 285, h(n) = 242, f(n) = 527
Fringe:  Mehadia (461), Lugoj (466), Pitesti (503), Lugoj (524),
Drobeta (527), Timisoara (580), Arad (595), RV (604), Drobeta
(627)

Path = Lugoj -> Mehadia -> Drobeta -> Mehadia
n = Lugoj:  g(n) = 290, h(n) = 244, f(n)= 534
n = Drobeta:  g(n) = 295, h(n) = 242, f(n) = 537
```

```
Fringe:  Lugoj (466), Pitesti (503), Lugoj (524), Drobeta (527),
Lugoj (534), Drobeta (537), Timisoara (580), Arad (595), RV
(604), Drobeta (627)

Path = Lugoj -> Timisoara -> Lugoj
n = Timisoara:  g(n) = 333, h(n) = 329, f(n) = 662
n = Mehadia:  g(n) = 292, h(n) = 241, f(n) = 533
Fringe:  Pitesti (503), Lugoj (524), Drobeta (527), Mehadia
(533), Lugoj (534), Drobeta (537), Timisoara (580), Arad (595),
RV (604), Drobeta (627), Timisoara (662)

Path = Lugoj -> Mehadia -> Drobeta -> Craiova -> Pitesti
n = Bucharest:  g(n) = 504, h(n) = 0, f(n) = 504
n = Craiova:  g(n) = 541, h(n) = 160, f(n) = 701
n = RV: g(n) = 500, h(n) = 193, f(n) = 693
Fringe:  Bucharest (504), Lugoj (524), Drobeta (527), Mehadia
(533), Lugoj (534), Drobeta (537), Timisoara (580), Arad (595),
RV (604), Drobeta (627), Timisoara (662), RV (693), Craiova
(701)
```