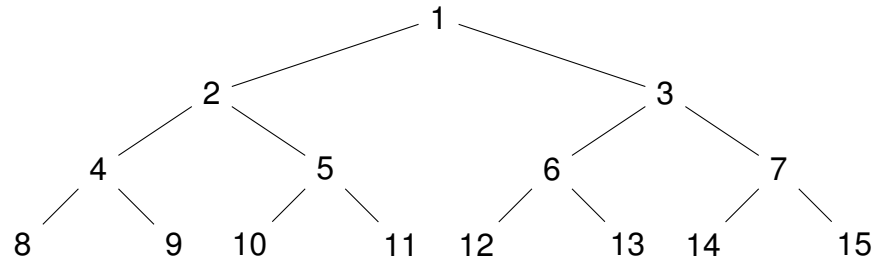
CSC242: Homework 1.2
AIMA Chapter 3.3.2–3.6

1. Consider a state space where the start state is number 1 and each state k has two successors numbered $2k$ and $2k + 1$ respectively.

   (a) Draw the portion of the state space for states 1 to 15.
   ANSWER:



   (b) Suppose the goal state is 11. List the order the nodes will be visited for breadth-first search, depth-first search to a depth limit of 3, and iterative deepening depth-first search.
   ANSWER:

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

2. For a search tree, let b be the branching factor, d the depth of the shallowest ("best") solution, and m the maximum depth of the tree. Complete the following table:

	Breadth-first search	Depth-first graph search	Depth-first tree search	Iterative-deepening depth-first tree search
Time Complexity				
Space Complexity				
Complete?				
Optimal?				

ANSWER: See AIMA p. 91.

3. Define what it means for a heuristic to be *admissible*. Give a simple example of an admissible heuristic for a problem and explain why it's admissible.

ANSWER: An admissible heuristic never *over*estimates the distance to a goal node.

An example of an admissible heuristic is the straight-line distance between cities used in the Romania map problem. Since the distance by road between two cities can never be *less* than the straight-line distance, this heuristic never *over*estimates.

4. Trace the operation of A* search applied to the problem of getting to Bucharest from Lugoj using the straight-line distance heuristic. Show the sequence of nodes and the g, h, and f scores for each node. Show the solution found by the search.

   ANSWER: A* search uses a frontier implemented as a priorty queue (heap) ordered by evaluation function $f(n) = g(n) + h(n)$ where $g(n)$ is the path cost to node n and $h(n)$ is an (admissible) estimate of the cost of the shortest path to a goal through n. Thus, "$f(n)$ is the estimated cost of the cheapest solution through n." (AIMA p. 93) For this question, we're told to use h_{SLD} from AIMA Fig. 3.22 for the heuristic estimate.

   To trace the execution, you should show the nodes being removed from and added to the frontier (the latter with their g, h, and f values in that order). At each step the lowest-cost node is removed from the heap. You don't necessarily have to show the heaps at each step.

   To recover the optimal path, you need to follow the parent pointers stored in the nodes. This is the SOLUTION function described on AIMA p. 79.

   Note that A* as described in AIMA uses tree search (no explored set). For example, you can see in Fig. 3.24 that it adds Arad again when expanding Sibiu even though Arad was the initial state.

   You should also note that it says (p. 93) "the algorithm is identical to UNIFORM-COST-SEARCH except that A* uses $g + h$ instead of g." On page 84, it describes how in uniform-cost search "a test is added in case a better path is found to a node currently on the frontier," in which case the worse node is replaced. It gives an example in the following text and in Fig. 3.15. This does not come up in the Arad-Bucharest example (Fig. 3.24), but it does come up in the Lugoj-Bucharest problem, as you can see in my trace.

   I would just add that several of these subtle understandings of the algorithms came from my having implemented them. It's also easier than doing the problem by hand for you.

   Output on next page...

```
node: L 244.0
  added child: M 70.0 241.0 311.0
  added child: T 111.0 329.0 440.0
node: M 311.0
  added child: L 140.0 244.0 384.0
  added child: D 145.0 242.0 387.0
node: L 384.0
  added child: M 210.0 241.0 451.0
node: D 387.0
  added child: C 265.0 160.0 425.0
node: C 425.0
  added child: R 411.0 193.0 604.0
  added child: P 403.0 100.0 503.0
  added child: D 385.0 242.0 627.0
node: T 440.0
  added child: A 229.0 366.0 595.0
  added child: L 222.0 244.0 466.0
node: M 451.0
  added child: D 285.0 242.0 527.0 replacing 627.0
node: L 466.0
  added child: M 292.0 241.0 533.0
  added child: T 333.0 329.0 662.0
node: P 503.0
  added child: B 504.0 0.0 504.0
  added child: C 541.0 160.0 701.0
node: B 504.0
  goal: B path cost=504.0
  number of nodes visited: 10
solution: L M D C P B
```

5. Consider the following heuristic functions for the 8-puzzle:

   h_1: Number of misplaced tiles

   h_2: Sum of Manhattan distances of tiles from final positions

   (a) Are these heuristics admissible? Why or why not?
   ANSWER: These heuristics are described on AIMA p. 103.

   (b) Consider the heuristic
   $$h = h_1 + h_2$$
   Is this heuristic admissible? Prove it or give a counterexample.
   ANSWER: No it is not admissible since it can overestimate the cost of a solution. For example, for the goal state in Figure 3.28 on AIMA p. 103, consider the following state of the 8-puzzle:

3	1	2
	4	5
6	7	8

   The only difference between this and the goal state is that the "3" tile is in the top-left rather than middle-left position. For this state, $h_1 = 1$ and $h_2 = 1$, so $h = 2$, but the true cost of a solution is just 1 (move the "3" down). So the heuristic has overestimated, and hence is not accessible.
   This is a good lesson about blindly combining heuristics (just because one is good doesn't mean that two are better).

6. (Harder) Prove that if h never overestimates by more than c, then A* using h returns a solution whose cost does not exceed that of the optimal solution by more than c. This is a very useful thing to know in practice.

   Hint: After you think about yourself, if you want some help, try the following approach:

   (a) Start with the definition of the evaluation function $f(n)$ for best-first search (of which A* is one flavor). See AIMA p. 93 if you don't remember what this is.

   (b) Suppose you had an optimal heuristic function $h^*(n)$ that always returned *exactly* the cost to a goal node. Using the condition given in the question, give an expression relating your heuristic evaluation function $h(n)$ to $h^*(n)$.

   (c) Now consider an optimal (cheapest) goal node, call it G^*. What is the path cost from the root to G^*? See AIMA p. 83 if you don't remember what this is (and it's also used in the definition of f).

   (d) Now suppose that there was some other goal node, call it G, that is suboptimal by more than c. Give an expression for the path cost from the root to G in terms of quantities you have previously defined.

   (e) Put the pieces together to show that G will never be expanded before an optimal goal is expanded, which is what the question asks you to prove.

   ANSWER: For the proof, and following the steps of the hint:

   (a) By definition, $f(n) = g(n) + h(n)$, where $g(n)$ is the path cost from the root to node n, and $h(n)$ is the estimated cost of the cheapest path from n to a goal.

   (b) As stated in the problem, suppose that $h(n) \leq h^*(n) + c$, where h^* is the minimal cost to a goal node (that is, it's the optimal heuristic function, which is obviously admissible).

   (c) The path cost of an optimal goal G^* is $g(G^*)$. Let $C^* = g(G^*)$.

   (d) Let G be a goal node that is suboptimal by more than c, that is, $g(G) > C^* + c$.

   (e) Now consider any node n on a path to an optimal goal. We have:

$$\begin{aligned} f(n) &= g(n) + h(n) && \text{(a)} \\ &\leq g(n) + h^*(n) + c && \text{(b)} \end{aligned}$$

   From (c), C^* is the path cost to an optimal goal, and n is a node on a path to an optimal goal, so $g(n) + h^*(n) = C^*$. Therefore:

$$\begin{aligned} f(n) &\leq C^* + c \\ &< g(G) && \text{(d)} \end{aligned}$$

   Since $f(n) < g(G)$ for any node n on a path to an optimal goal, the goal node G that is suboptimal by more than c will never be expanded before an optimal goal is expanded.

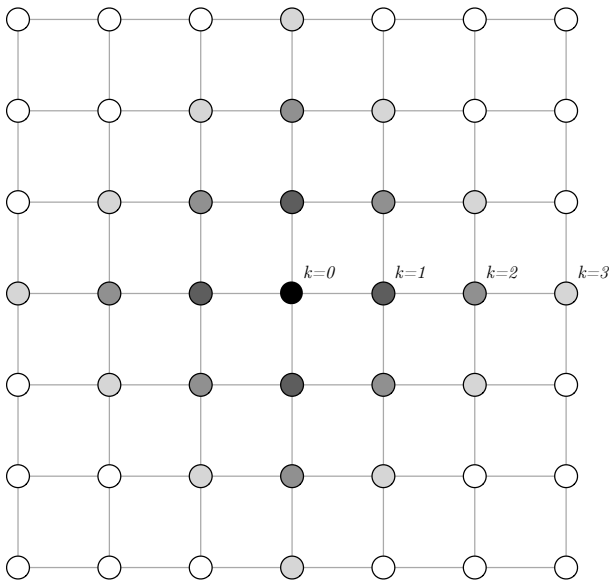
7. Suppose that you need to search a state space that is a regular (evenly-spaced), unbounded, discrete, 2D grid of points. States (points) are connected to their four neighbors (north, south, east, and west). The start state is the origin $(0, 0)$. The goal state is (x, y). Note that since the state space is discrete, these coordinates are *integers*.

   (a) What is the branching factor b in this state space?

   ANSWER: Branching factor is number of neighbors = 4.

   (b) How many distinct states are there at depth k (for $k > 0$)? Hint: Draw a picture and mark a couple of generations of states in different colors to see the pattern, then generalize.

   ANSWER: The states at depth k (i.e., the frontier at that depth) form a square diamond around the origin:



   Total states at depth k: $4k$.

   (c) What is the maximum depth of search for goal (x, y)?

   ANSWER: Worst case requires search to depth $x+y$ to expand node at (x, y). Try it and see, for a node with $x = y$.

   (d) Give a big-O expression for the maximum number of nodes expanded by breadth-first tree search?

   ANSWER: Without checking for repeated states, BFS expands exponentially many nodes to depth $x + y$: $O(4^{(x+y)})$.

   (e) Give a big-O expression for the maximum number of nodes expanded by breadth-first graph search?

   ANSWER: When states are not repeated, there are only quadratically many states within the square to depth $x + y$: $O((x + y)^2)$.

(f) For a state (u, v), consider the heuristic $h = |u - x| + |v - y|$. Is h admissible? Why or why not?

ANSWER: This is the Manhattan distance metric and is admissible (see AIMA p. 103).

(g) How many nodes are expanded by an A* search using h?

ANSWER: All nodes in the rectangle defined by $(0, 0)$ and (x, y) are candidates for the optimal path. There are $x \times y$ of these, all of which may be visited in the worst case.

(h) Is h still admissible if some links (edges) are removed?

ANSWER: Yes. Removing links can only lengthen a path by forcing a "detour", so h remains an underestimate. That is a cool thing to realize about admissible heuristics.

(i) Is h still admissible if some links (edges) are added between non-adjacent states?

ANSWER: No. "Non-local" or "shortcut" links can reduce the actual path length below the Manhattan distance h. That is also cool.