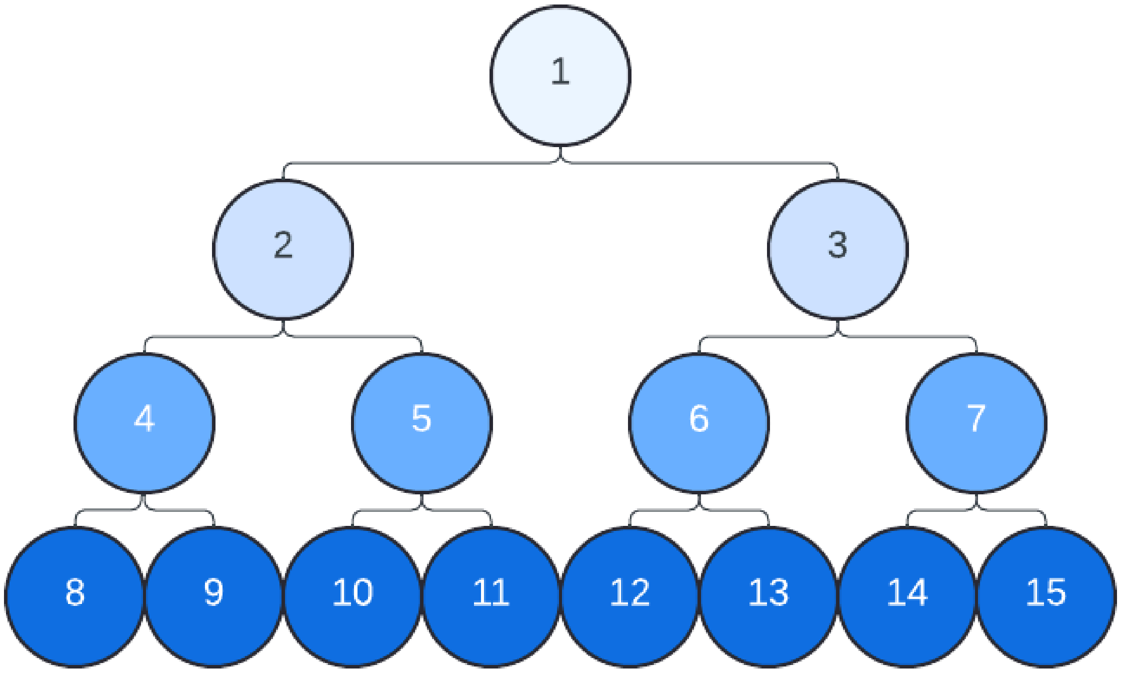
MAI5100 - Written part of homework
Submitted by: Maryam Bacchus

R&N Problem 3.15 (10 points)
Consider a state space where the start state is number 1 and each state k has two successors: numbers 2k and 2k+1.

1. Draw the portion of the state space for states 1 to 15.



2. Suppose the goal state is 11. List the order in which nodes will be visited for breadth-first search, depth-limited search with limit 3, and iterative deepening search.

Breadth-first search: 1, 2, 3, 4, 5, 6, 7, 8, 9 ,10, 11
Depth-limited search: 1, 2, 4, 8, 9, 5, 10, 11
Iterative deepening search:

- *Level 0 - 1*
- *Level 1 - 1, 2, 3*
- *Level 2 - 1, 2, 4, 5, 3, 6, 7*
- *Level 3 - 1, 2, 4, 8, 9, 5, 10, 11*

3. How well would bidirectional search work on this problem? What is the branching factor in each direction of the bidirectional search?

Bidirectional search would work well on this problem because it would lessen the nodes that need to be explored. In the forward direction, every node k has 2 successors, therefore, the forward branching factor is 2. While in the backward direction, every node has 1 predecessor, therefore, the backward branching factor is 1. Bidirectional search would reduce the total number of nodes explored compared to unidirectional search.

4. Does the answer to (3) suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?

Yes, because of the way the graph is structured, it has a branching factor of 1 in the backward direction, enabling you to backtrack from your known goal state by recursively dividing by 2 until the quotient is 1, eliminating the need for a search.

5. Call the action going from k to 2k Left, and the action going to 2k+1 Right. Can you find an algorithm that outputs the solution to this problem without any search at all?

```
// Define goal state
goal = 11

// Loop until goal is not 1
while goal > 1
    // Divide by 2 and update goal variable
    goal /=  2

    // If the quotient is odd, the value is some 2k+1, therefore go right.
    // If the quotient is even, the value is some 2k, there go right
    if isOdd(goal):
        path.append(Right)
    else:
        path.append(Left)

// Return reversed path
return path.reverse()
```

R&N Problem 3.27 (10 points)
n vehicles occupy squares (1,1) through (n,1) (i.e., the bottom row) of an n×n grid. The vehicles must be moved to the top row but in reverse order; so the vehicle i that starts in (i,1) must end up in (n−i+1,n). On each time step, every one of the n vehicles can move one square up, down, left, or right, or stay put; but if a vehicle stays put, one other adjacent vehicle (but not more than one) can hop over it. Two vehicles cannot occupy the same square.

1.Calculate the size of the state space as a function of n.

Total squares: nxn = n^2
Number of vehicles: n
State Space: Number of ways to place n vehicles on n^2 squares where no two vehicles overlap
Permutation formula: nPr = n!/(n-r)!
State space: f(n) = n^2! / (n^2 - n)! - using permutation formula

2. Calculate the branching factor as a function of n.

Total possible moves: up, down, left, right or hop = 5
Total possible moving vehicles: n
Branching factor: f(n) = 5^n

3. Suppose that vehicle i is at (x,y). Write a nontrivial admissible heuristic hi for the number of moves it will require to get to its goal location (n−i+1,n), assuming no other vehicles are on the grid.

Vehicle position: (x, y)
Goal location: (n−i+1, n)

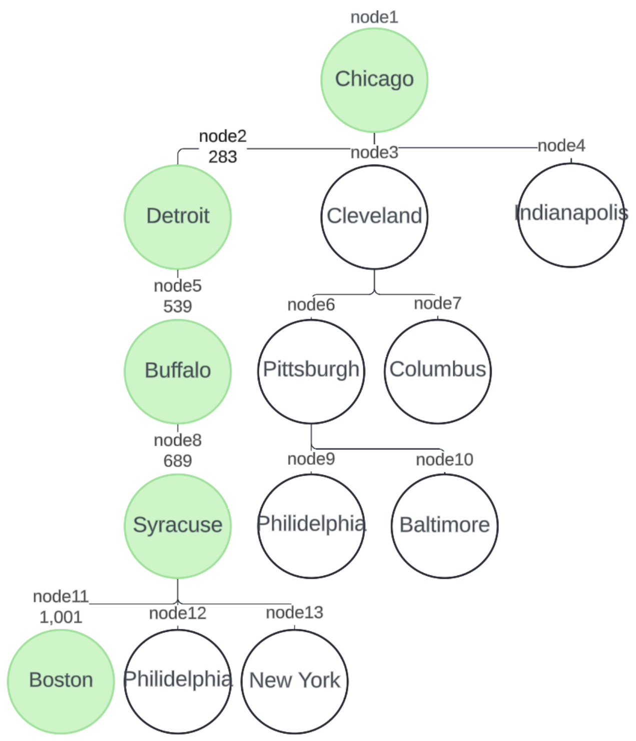
Using Euclidean distance formula
Horizontal distance = x - (n-i+1)
Vertical distance = (y - n)
hi = sqrt(x - (n-i+1)^2 + (y - n)^2)

4. Which of the following heuristics are admissible for the problem of moving all n vehicles to their destinations? Explain.

The heuristic max(h1,…,hn) is admissible because it takes the maximum value of all of the heuristics for n vehicles. Considering the euclidean distance heuristic from the previous question, the maximum cost will always be less than or equal to the actual cost because the heuristic represents a straight line to the goal and the path to the goal is not always a straight line.

“Manual” Search Tree Generation (20 points)

Breadth-First Search (BFS)



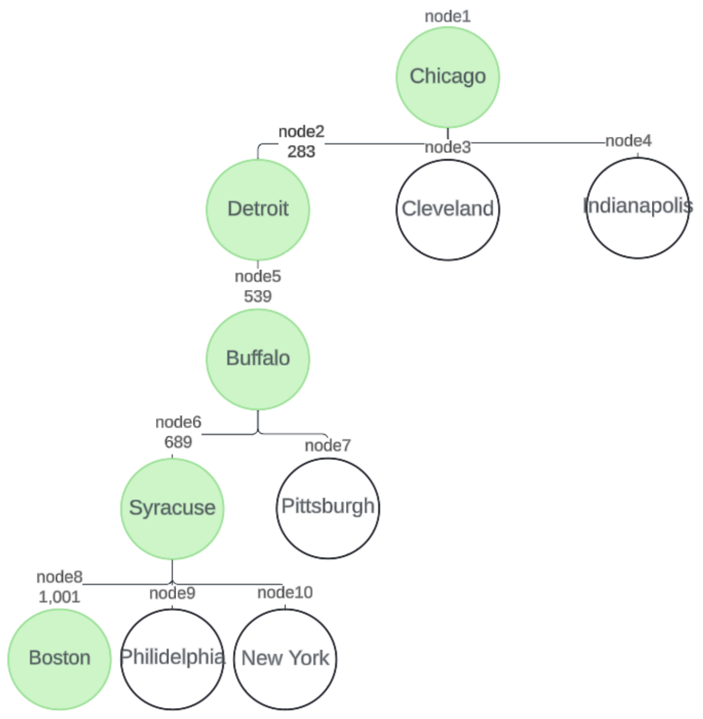
In the search graph above, the nodes in green represent the path to the goal. The outlined nodes represent explored nodes, and the labels (node1 - node13) represent the order in which nodes were explored for this search.

Using breadth first search, the solution path was found to be Detroit -> Buffalo -> Syracuse -> Boston.

The total cost is 1,001.

In this case, BFS is optimal.

Depth-first Search (DFS)



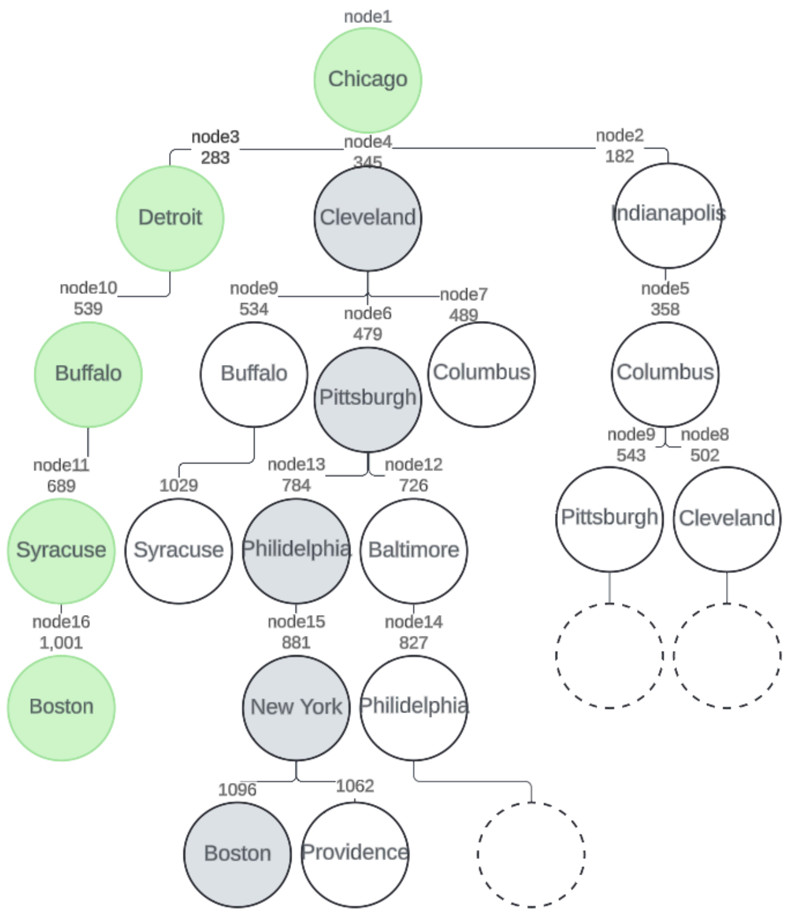
In the search graph above, the nodes in green represent the path to the goal. The outlined nodes represent explored nodes, and the labels (node1 - node10) represent the order in which nodes were explored for this search.

Using depth first search, the solution path was found to be Detroit -> Buffalo -> Syracuse -> Boston.

The total cost is 1,001.

In this case, DFS is optimal.

Uniform-Cost Search (UCS)



In the search graph above, the nodes in green represent the shortest path to the goal. The outlined nodes represent explored nodes, the labels (node1 - node13) represent the order in which nodes were explored for this search, and the dashed circles represent loops from already explored nodes.

Using Uniform Cost Search, the solution path was found to be Detroit -> Buffalo -> Syracuse -> Boston.

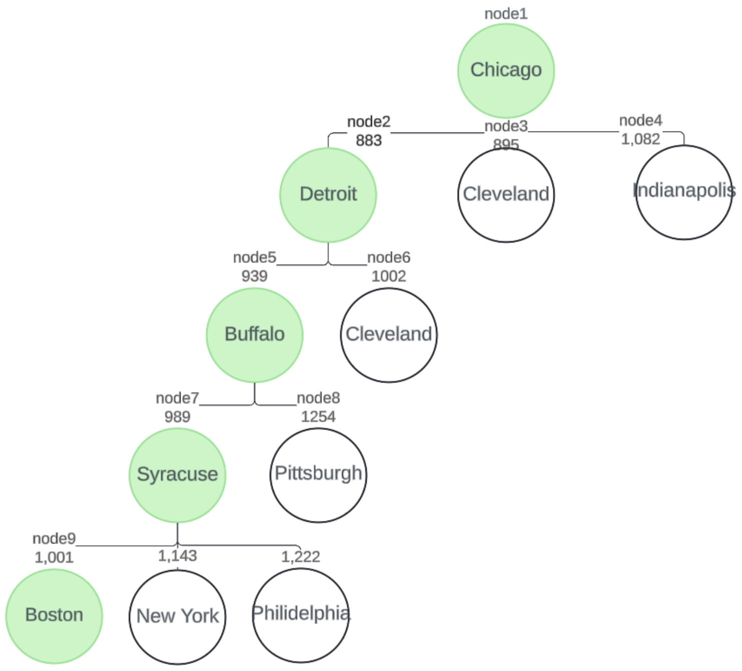
The total cost is 1,001.

In this case, UCS finds the shortest path, but is not not optimal due to the large numbers of nodes it expands out.

A* Search

Heuristics: Using the Haversine formula for straight-line distance calculation based on real coordinates for each city. (Each calculation was generated by AI)

City
Chicago: 850 miles
Indianapolis: 900 miles
Detroit: 600 miles
Cleveland: 550 miles
Columbus: 650 miles
Pittsburgh: 500 miles
Buffalo: 400 miles
Syracuse: 300 miles
Philadelphia: 280 miles
New York: 200 miles
Providence: 40 miles
Boston: 0 miles



In the search graph above, the nodes in green represent the shortest path to the goal. The outlined nodes represent explored nodes, the labels (node1 - node13) represent the order in which nodes were explored for this search,

Using A* search with an admissible and non-trivial heuristic, the solution path was found to be Detroit -> Buffalo -> Syracuse -> Boston.

The total cost is 1,001.

A* Search is optimal for this solution.