CSE 586 Assignment 1

Caleb Alexander

Problem 1. Algorithm Implementation

Answer 1. Will be shown w/ group.

Problem 2. Pacman and Ms. Pacman are lost in an $N \times N$ maze and would like to meet; they don't care where. In each time step, both simultaneously move in one of the following directions: {NORTH, SOUTH, EAST, WEST, STOP}. They do not alternate turns. We must devise a plan which positions them together, somewhere, in as few time steps as possible. Passing each other does not count as meeting; they must occupy the same square at the same time. We can formally state this problem as a single-agent state-space search problem as follows: States: The set of pairs of positions for Pacman and Ms. Pacman, that is, $\{(x_1, y_1), (x_2, y_2) : x_1, x_2, y_1, y_2 \in \{1, 2, \ldots, N\}\}$.

Size of state space: N^2 for both Pacmen, hence N^4 total.

Goal test: $Goal((x_1, y_1), (x_2, y_2)) := (x_1 = x_2) \wedge (y_1 = y_2)$

(**a**) Determine the branching factor. (2 points).

(**b**) For each of the following graph search methods determine and explain whether it is guaranteed to output optimal solutions to this problem? (6 points).

- DFS
- BFS
- UCS

Answer 2. (**a**) At any particular state, s, see that the maximum number of next steps for a single agent (Pacman or Ms. Pacman) is 5 (Cardinality of action space $|A| = |\{\text{NORTH, SOUTH, EAST, WEST, STOP}\}| = 5$). Let a_1 and a_2 be Pacman's and Ms. Pacman's actions respectively. Since we are looking at two agents, the space we are examining can be represented as:

$$S = \{(a_1, a_2) : a_1, a_2 \in A\} \quad \text{where } A = \{\text{NORTH, SOUTH, EAST, WEST, STOP}\}.$$

Since order matters (i.e. (NORTH, SOUTH) $\neq$ (SOUTH, NORTH)) we can say that $|S| = |A|^2 = 5^2 = 25$. Hence our branching factor is 25.

(**b**) We say that a solution is optimal if the cost of the path returned by the algorithm is the minimum possible cost.

- DFS will not necessarily be optimal. Consider the scenario in which Pacman is exactly one space to the left of Ms. Pacman. Assuming we have lettered our nodes such that (NORTH, NORTH) is lexicographically placed before (EAST, STOP); notice that DFS would always find a solution like $\{(\text{NORTH, NORTH}), (\text{EAST,STOP})\}$ before the more optimal solution $\{(\text{EAST, STOP})\}$.

In general, DFS is never *guanranteed* to be optimal (Though for some graphs it may happen to find the optimal path).

- BFS is said to be optimal whenever the step-cost is uniform or zero (i.e. edges are unweighted or every edge happens to be weighted the same). In this case, there is no cost to moving for Pacman or Ms. Pacman. Thus BFS ought to be optimal.
- When edges are unweighted UCS will always take the same path as BFS. Thus UCS will also be optimal in this scenario.

Problem 3. Consider a state space where the start state is number 1 and each state k has two successors: $2k$ and $2k+1$. Suppose you want to navigate your robot from state 1 to state 2020. Can you find an algorithm that outputs the solution to this problem without any search at all? Output the solution for 2020. (6 points) 10111011100 (1500) (left(2), right(5), right(11), right(23), left(46), right(93), right(187), right(375), left(750), left(1500))

Answer 3. Consider the following psuedocode.

```
num = 2020
bin_string = binary_string(num) # Will be 11111100100
bin_string.remove(0) # remove the first bit since we know the end is 2020
path = []

for bit in bin_string:
    if bit == 1:
        path.append(right)
    else:
        path.append(left)
```

Notice that for 11111100100 this should give us the path

[right(3), right(7), right(15), right(31), right(63), left(126), left(252), right(505), left(1010), left(2020)]

which is correct.

Note: the numbers to the right of each step are for your benefit. Since each left step is just taking $2k$ and each right step is taking $2k+1$ where k is the previous position; it should be rather easy to verify this solution.

Problem 4. Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall. We can formulate this problem as follows. We'll define the coordinate system so that the center of the maze is at $(0,0)$, and the maze itself is a square from $(-1,1)$ to $(1,1)$.
Initial state: Robot at coordinate $(0,0)$, facing North.

Successor function: Move forwards any distance d; change direction robot is facing.

Cost function: Total distance moved.

(**a**) Let (x, y) be the current location. What is the Goal test? (2 points)

(**b**) How large is the state space? (2 points)

Answer 4. (**a**) The following goal test works $Goal((x, y)) := |x| > 1$ or $|y| > 1$. With the following pseudo code.

```
function goal_check(x,y):
    if abs(x) > 1 or abs(y) > 1:
        return true
    else
        return false
```

(**b**) For any particular state, s, described by (x, y). Notice that x, y can be any real number $(x, y \in \mathbb{R})$ so that $-1 \leq x, y \leq 1$ (can also be stated as $x, y \in [-1, 1]$). Furthermore, the next step can be described as a tuple (dir, d) where $dir \in \{$NORTH, EAST, SOUTH, WEST$\}$ and $d \in \mathbb{R}$ such that $0 \leq d \leq 1$ (same as $d \in [0, 1]$). It should be clear that $|[-1, 1]| = \infty$, in fact this is uncountably infinite as is $|[0, 1]|$. Hence, the space of possible states is infinite and the space of possible next states is also infinite.

Problem 5. For each of the following assertions, say whether it is true or false. Justify your answers. (6 points: 2 points each)

(**a**) SKIP

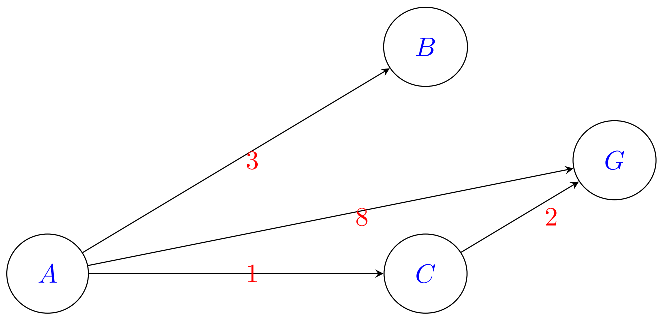
(**b**) Every optimal search strategy is necessarily complete.

(**c**) Breadth-first search is optimal if the step cost is positive.

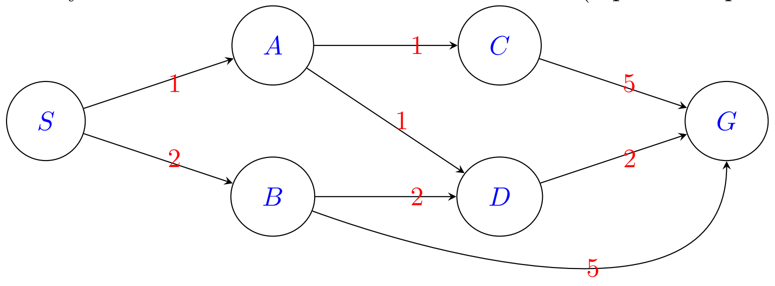
Answer 5. (**a**) SKIP

(**b**) TRUE. An optimal search strategy is one which will always find the path to the goal node with the lowest cost. A complete search strategy will always find a path to the goal node. If we suppose that a strategy is optimal (it will always find the path with the lowest cost) then we are implicitly supposing also that the strategy will always find a path to the goal node. Hence, any optimal strategy must necessarily be complete.

(**c**) FALSE. Consider the following graph. BFS would find the path $\{A, G\}$ with cost 8. But the optimal path is $\{A, C, G\}$ with cost 3.



Problem 6. For the search problem shown below, S is the start state, and G is the (only) goal state. Break any ties alphabetically. What paths would the following search algorithms return for this search problem? Give your answers in the form '$S - A - D - G$'. (6 points: 2 points each)



(**a**) BFS

(**b**) UCS

(**c**) DFS

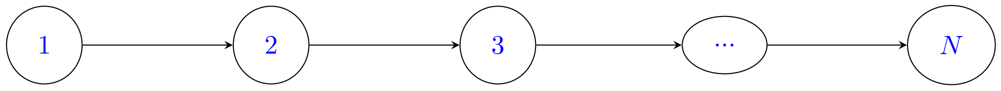
Answer 6. (**a**) $S - B - G$ with cost $= 7$.

(**b**) $S - A - D - G$ with cost $= 4$.

(**c**) $S - A - C - G$ with cost $= 7$.

Problem 7. Describe a state space in which iterative deepening search performs much worse than depth-first search (for example, $O(n^2)$ vs $O(n)$). (Only for CSE 586 students)

Answer 7. Consider any tree in which the branching factor $b \leq 1$. Use the following graph as an example.



[Note that any directed tree with $b \leq 1$ will take the appearance of a line]

Let 1 be our start node and N our goal node. In this case it should be clear that DFS will traverse the graph once, so N steps. But the iterative deepening search will traverse as shown below.

Node Traversal	Depth-Limit
{1}	1
{1, 2}	2
{1, 2, ..., N}	N

From this it is clear that the number of steps in IDS will be

$$\sum_{l=1}^{N} l = 1 + 2 + 3 + \ldots + N.$$

Notice that N can be paired with 1, $N-1$ with 2, $N-2$ with 3 and so on until we get the following equalities. If N is even,

$$\begin{aligned} 1 + 2 + 3 + \ldots + N &= (N+1) + (N-1+2) + (N-2+3) + \ldots + (N - \frac{N}{2} + \frac{N}{2} + 1) \\ &= \frac{N}{2}(N+1) \\ &\geq \frac{N^2}{2}. \end{aligned}$$

If N is odd,

$$\begin{aligned} 0 + 1 + 2 + 3 + \ldots + N &= 0 + 1 + 2 + 3 + \ldots + (N-1) + N \\ &= (N+0) + (N-1+1) + (N-2+2) + (N-3+3) + \ldots + (N - \frac{N}{2} + \frac{N}{2}) \\ &= N(\frac{N}{2}) \\ &= \frac{N^2}{2}. \end{aligned}$$

In either case the total number of steps, s, will have the following inequality $s \geq \frac{N^2}{2}$. Big-O discards proportions; so in this case IDS is $O(N^2)$ where as DFS is clearly $O(N)$.