CS 411: Artificial Intelligence I
Grad Homework 1
Due: Sunday of Week 2 11:59pm

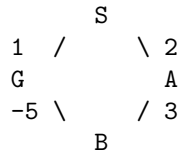
You may discuss the assignment with other students, but if you do you must note on your submission who you discussed it with. The actual submission must be entirely your own work. It must be submitted via gradescope. Please make sure to tag which page(s) each problem is answered on at the appropriate step in the submission process.

1. **Negative costs**

   (a) **Suboptimality**

   Give an example showing that uniform cost search is not optimal if costs can be negative

   Consider the following search graph

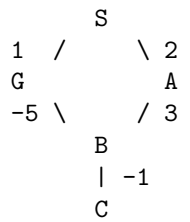


   UCS will first expand G as it is the lowest cost path from S. As it explores in increasing cost contours, it has found a path to the goal of cost 1 and the next node in the priority queue has cost 2. Thus it will terminate and not explore the rest of the graph, never discovering that the negative edge allows a zero cost path SABG.

   (b) **Infinite Loops**

   Give an example where there is no (finite) optimal path if costs can be negative

   Consider the following search graph



   Every time the loop $B \to C \to B$ is taken, the total cost is -2. Thus, the cost can be made arbitrarily small by going around the loop an arbitrary number of times. To make this a bit more precise, the cost of the path $SAB(CB)^kG$ is $-2k$, which diverges to $-\infty$ as k increases.

2. **Iterative Deepening** Give an example with of a graph n nodes where breadth first search and depth first search both expand $O(n)$ nodes while iterative deepening search expands $O(n^2)$ nodes. How can you reconcile this with Question 6 of Written Homework 1?

   Consider the graph

   ```
   S -> 1 -> 2 -> ... -> n-2 -> G
   ```

   This has n nodes and DFS and BFS both just follow the path expanding n nodes. However, iterative deepening expands every path starting at S for a total of $\sum_{i=2}^{n} i = n(n+1)/2 - 1 \in O(n^2)$.

   This may initially seem to contradict the answer to Q6, as there we claimed that the work is $O(b^d)$, while in this case $b^d = 1^d = 1$. We can reconcile these because there the analysis was asymptotic in both b and d.

3. **Completeness of bidirectional search** Prove that bidirectional uniform cost tree search is complete. You may assume there is a single goal state.

   The problem asks you to prove that it is complete, which our definition says that the algorithm must return a solution if one exists. In turn, a solution is defined as a sequence of actions (path) from the start state to a goal state. Thus, we will show that if a solution exists the algorithm will terminate having found a node in both searches. Then we argue that given such a node we can construct a solution. This yield the following proof by invariant:

   Suppose a solution exists. This means there exists a path from the start to the goal. Consider some such path. By the completeness of UCS, the forward and reverse searches will explore each node of this path in order (unless some other solution is found first). The following invariant this holds: either a node on this path exists in the fringe of both searches (with the node in the forward search earlier in the path) or the algorithm has terminated. At the start of the algorithm, the the start and goal states will be in the respective fringes. Each time a node is removed from the fringe, the next one along the path is added, maintaining our invariant. Thus, at some point a node along the path will be expanded while the next node along the path is in the fringe of the other search, causing the algorithm to terminate. We can then return the resulting path the forward search found to this intermediate node, combined with the (reversed) path the reverse search found from the goal to it.

   A common error is to realize the basic structure of this argument, but fail to realize that the algorithm is defined to terminate only when one search expands a node the other is aware of, not when a goal state is expanded. One tricky detail the above proof deals with this that the algorithm is only aware of nodes in the fringe. Thus, strictly speaking, it is not sufficient to argue that the two searches meet, but rather a proof needs to argue that the node is actually still in the fringe of the other search (as the algorithm does not track nodes not in the fringe).

Grading standards:

- Full Credit (1.0) - All questions attempted, most with minor or no errors
- Partial Credit (0.5) - A number of questions may have significant errors
- Some Credit (0.25) - Many questions attempted with answers demonstrating some understanding of relevant concepts
- Insufficient evidence (0) - Many questions either not attempted or with answers not demonstrating understanding of relevant concepts