CSCI-GA.2560-001, Artificial Intelligence January 31, 2022

Solutions to Problem 1 of Homework 1 (3 Points)

Name: Anav Prasad (ap7152) *Due: 5PM on Monday, January 31*

Collaborators:

Show the order of nodes visited using Breadth First Search (BFS), and the final path selected. Is this optimal? Why or why not?

Solution: The Breadth First Search (BFS) algorithm would run as follows:
(Note that we keep a track of per-path visited list to avoid revisiting nodes)

```
Start   -> K
        -> L
    (now K and L will be expanded in that order)
Start   -> K    -> L
                -> N
        -> L    -> K
                -> M
                -> O
    (now we expand the 5 open paths)
Start   -> K    -> L    -> M
                        -> O
                -> N    -> Goal # target reached, terminate
```

So, the path that ended up being selected was `Start -> K -> N -> Goal` with net cost of 28.

But, by inspection, it can be seen that `Start -> L -> K -> N -> Goal` has a cheaper net cost of 25. Thus, BFS was not able to find out the most optimal path. This is so because BFS does not consider the weights of the edges while searching for the target node and, as such, ends up just minimizing the number of *nodes* (or *edges*) traversed on the way to the target node instead of the cost incurred on the said path. As such, BFS is not the optimal search algorithm under non-unit edge weights. □

CSCI-GA.2560-001, Artificial Intelligence January 31, 2022

Solutions to Problem 2 of Homework 1 (3 Points)

Name: Anav Prasad (ap7152) *Due: 5PM on Monday, January 31*

Collaborators:

Show the order of nodes visited using Iterative Deepening (initial depth of 2, increasing depth by 1), and the final path selected. Is this optimal? Why or why not?

Solution: The Iterative Deepening Search (IDS) algorithm would run as follows:
(Note that we keep a track of per-path visited list to avoid revisiting nodes)

```
ID Depth = 2
Start   -> K    -> L # terminate depth
                -> N # terminate depth
        -> L    -> K # terminate depth
                -> M # terminate depth
                -> O # terminate depth

ID Depth = 3 # increase by 1
Start   -> K    -> L    -> M # terminate depth
                        -> O # terminate depth
                -> N    -> Goal # target reached, terminate
```

Again, as was the case in BFS, the path that ended up being selected was `Start -> K -> N -> Goal` with net cost of 28.

But we know that `Start -> L -> K -> N -> Goal` has a cheaper net cost of 25. Thus, IDS was also not able to find out the most optimal path. That is so because IDS, like BFS, also does not care for the weights of the edges its searching through. It just tries to find the target node (`Goal` in this case) at the most minimum depth it can. But, as we can see from the optimal path, the depth at which the optimal path reaches `Goal` is more than the minimum depth at which `Goal` can be reached. So IDS ends up not finding the optimal path. □

CSCI-GA.2560-001, Artificial Intelligence January 31, 2022

Solutions to Problem 3 of Homework 1 (4 Points)

Name: Anav Prasad (ap7152) *Due: 5PM on Monday, January 31*

Collaborators:

Using the above h-function, show the order of nodes visited using A*, and the final path selected. Is this optimal? Why or why not?

Solution:
`g(n)`: actual cost from `Start` to node `n`
`h(n)`: heuristic cost from `n` to node `Goal` (as defined in the question)
The A* Search algorithm would run as follows:
(Note that we keep a track of per-path visited list to avoid revisiting nodes)

```
Start   -> K (( g = 14 ) + ( h(K) = 9  ) => d = 23)
        -> L (( g = 8  ) + ( h(L) = 24 ) => d = 32)

Since d of Start -> K is smallest, expanding Start -> K now:
Start   -> K    -> L (( g = 17 ) + ( h(L) = 24 ) => d = 41)
                -> N (( g = 23 ) + ( h(N) = 2  ) => d = 25)
Start   -> L         (( g = 8  ) + ( h(L) = 24 ) => d = 32)

Since d of Start -> K -> N is smallest, expanding Start -> K -> N now:
Start   -> K    -> L            (( g = 17 ) + ( h(L) = 24   ) => d = 41)
Start   -> K    -> N    -> Goal (( g = 28 ) + ( h(Goal) = 0 ) => d = 28)
                        -> P    (( g = 27 ) + ( h(P) = 5    ) => d = 32)
                        -> Q    (( g = 34 ) + ( h(Q) = 12   ) => d = 46)
Start   -> L                    (( g = 8  ) + ( h(L) = 24   ) => d = 32)

Since d of Start -> K -> N -> Goal is smallest, we can now conclude that the target
has been reached and terminate the A* algorithm.
```

So, the path that ended up being selected was `Start -> K -> N -> Goal` which has a net cost of 28.

But, we know that `Start -> L -> K -> N -> Goal` has a cheaper net cost of 25. Thus, A* was also not able to find out the most optimal path. This is so because the heuristic function heavily penalized the path that goes through `L` to reach `Goal`. Therefore, that path (i.e. the path that starts with `Start -> L`) ended up never being expanded and, as such, the algorithm failed to catch the most optimal path. □