CS 4804 Homework 1
Solution Sketches

---

1. **(10 points)** Since depth-first search (DFS) is not complete, the only situation where it would be competitive with depth-first iterative deepening (DFID) is if it finds a solution. Consider a search space where the solution lies at depth d, on the left-most path from the starting state. DFS would find this in $O(d)$ steps! 'Preferred' here thus refers to fewer node expansions or time complexity. The space complexities are the same.

   To receive full credit for this question, you must have rigged up a search space so that DFS will find a solution and DFID does a lot of work. For instance, you could have assumed that the branching factor is 1, to make DFID look really silly (it would still require $O(d^2)$ steps). Alternatively, you could have assumed that you know the solution depth, so that this can be used as a depth-bound (to ensure completeness of DFS) and so that DFID does a lot of work looking for solutions of lesser depth. There are several other answers.

2. **(20 points)** A very simple heuristic can be created using ideas from constraint satisfaction search. Keep in mind that the operators place queens only in *legal* positions. Hence, upon placing a queen we can forward propagate the constraints and determine the available legal positions. A simple heuristic can then be formulated as:

   $$h(n) = 4 - \#\text{queens currently placed} - \frac{\#\text{legal positions remaining}}{16}$$

   Note that the heuristic is admissible and is also zero at the goal state, which is convenient. We are assuming that all moves have equal, unit cost.

3. **(10 points)** This is trivial. You are expected to diligently add $g(n)$ to your heuristic to obtain the node evaluation function $f(n)$, and incorporate $f(n)$ as your primary driver for selecting nodes for expansion.

4. **(10 points)** Consider the problem formulation where the starting state has the disks neatly stacked on the left peg, and the ending state with the disks on the right peg (the goal test merely checks for this state). Every operator makes one legal move according to the rules of the game. The heuristic is meant to be applied to every state that we encounter along the path.

   To relax the problem for finding a heuristic, all you have to do is to think out-of-the-box and in a slightly wacky manner. Assume we model the given problem as:

   ```
   MOVE(x,y,z) <- TOP(x,y), TOP(m,z), x < m.
   ```

   whose interpretation is: you can move disk `x` from peg `y` to the top of peg `z` if `x` is on top of `y` and is smaller than disk `m`, which is on top of `z`. One way to relax this problem is as:

   ```
   MOVE(x,y,z).
   ```

   meaning `x` can be moved from anywhere on peg `y` (even if it is not on the top) to the top of `z` (even if it is bigger than the current disk on `z`)! Given this relaxed problem the optimal solution is easy to determine.

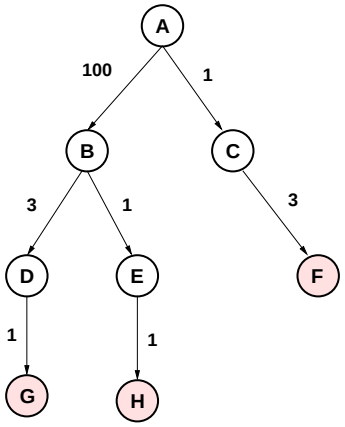
We should first remove any unwanted disks from peg `z` (placing them anywhere we like) and then move all disks, in the order we want, from wherever they are to peg `z`. This gives the heuristic as:

$$\begin{aligned} h(n) &= (2 * \#\text{misplaced disks currently on peg } \texttt{z}) + \\ &\quad \#\text{misplaced disks elsewhere} \end{aligned}$$

It is easy to show that this heuristic is admissible and ≥ 0 for all states.

5. **(10 points)** If $f(n) = h^*(n)$ and all edges have uniform cost, then best-first search is complete and will lead to the optimal solution. Even though we do not have the benefit of the 'restoring force' (i.e., $g(n)$), the heuristic will never lead us astray. This is because every local move is globally consistent — it is as if we first solved the problem and went back and solved it again with the hindsight of experience encoded as the heuristic. We will encounter this idea more when we study the topic called 'reinforcement learning.'

6. **(10 points)** If the edges do not have uniform cost, then the algorithm using $f(n) = h^*(n)$ is still complete but may not lead to the optimal solution. To see why, consider the search space shown below, where the goal nodes are F, G, and H. The optimal solution is F.



Using the above definition of $f(n)$, best-first search will expand node B after the root (because it has a shorter path to a goal node). After this point, nodes D and E look equally attractive. Either of them will lead to a suboptimal goal node! Notice that this algorithm will never backtrack.

The general problem can be characterized by saying that as long as every node has exactly one child node having a $h^*(n)$ strictly less than itself, then best-first search will lead to an optimal solution.

7. **(30 points)** Based on your experiments, it will be easy to show the following linear order of the heuristic functions (in order of worst to best):

$$h_5(n) \prec h_1(n) \prec h_4(n) \prec h_2(n) \prec h^*(n)$$

where $\prec$ denotes 'is dominated by.' All the given heuristics are admissible.