CIS 630 Homework #3

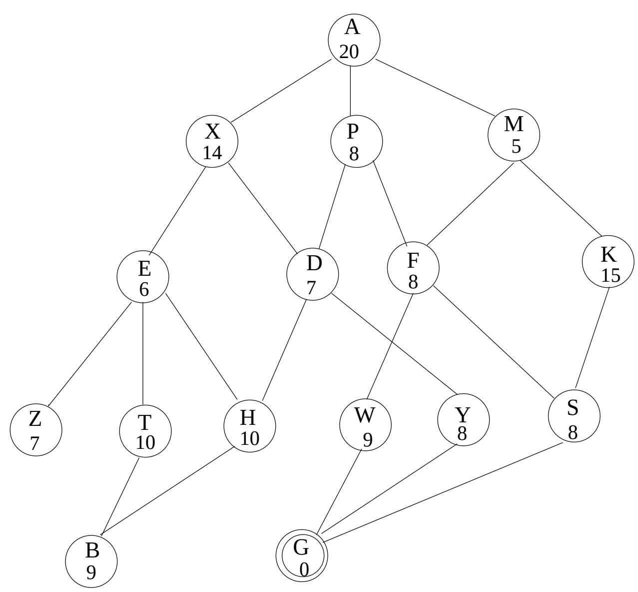
Due: October 10 (Wednesday) at class meeting.

Problem 1 – A Search Exercise (30%).

The figure below shows a graph structured representation for a search problem. Suppose that A is the initial node from which a search starts, and G is the goal node where a search ends. List the nodes in order they are visited by each of the following four search algorithms. Notice that the graph is not a tree, we need to mark the nodes which are visited. For the DFS and BFS, a child node will not be added to the OPEN list if it has been visited before (i.e. if it is in the current open or closed lists).

1. Depth-First Search.
2. Breadth-First Search.
3. Iterative Deepening Search (Starting with a depth bound $D_{max} = 0$, and increasing D_{max} by $\Delta = 2$ each time).
4. A heuristic search using $f(s) = g(s) + h(s)$. For each node s, use its depth for $g(s)$ (e.g. g(X) =1), and $h(s)$ is the number shown in the node. When there is a tie in number, the nodes are then ordered in their alphabetical order.

   Note that this is a search in a graph not a tree, in order to find the optimal path (lowest cost), the algorithm should update the $f(s)$ value for node s in the OPEN list. It proceeds in the following way. Suppose the algorithm is about to insert a new node s with an evaluation value $f(s)$ in the OPEN list. But it finds that s is already in the OPEN list with a value $f'(s)$ calculated by a previous route. If $f'(s) > f(s)$, then the algorithm should remove the old s node from the OPEN list and insert the new s with value $f(s)$. Otherwise, it disregards the current $f(s)$, and does not insert a second s.



Problem 2 – Uninformed Search (20%) .

Consider a complete tree of depth D and branching factor b (in our notation, the root always has depth 0). Suppose the goal node is at depth $g \leq D$.

1. In this tree, what is the number of terminal nodes (leaves)? and what is the number of non-terminal nodes?

2. What is the *minimum* (best case) and *maximum* (worst case) number of nodes that might be generated by a depth-first search algorithm with maximum search depth bound D?

3. What is the *minimum* and *maximum* number of nodes that might be generated by a breadth-first search algorithm?

4. What is the *minimum* and *maximum* number of nodes that might be generated by an iterative deepening search algorithm? Suppose it starts with depth 0 and increase depth by 1 each time.

Problem 3 – Heuristic Search (25%) .

For a heuristic algorithm to be optimal, a sufficient condition is that $h(s)$ is admissible,

$$h(s) \leq h^*(s) = c(s, s_g), \qquad s \in \Omega$$

Is this a necessary condition for optimality? If yes, prove it. If not, provide a counter-example. That is, to find an $h()$ which does not satisfy the admissible condition and still the heuristic search algorithm using this $h(s)$ can find the optimal solutions for general graphs. (I mean, you cannot provide an example which only work for a special graph of your choice.)

Problem 4 – Heuristic Search (25%) .

Let A_1^* and A_2^* be two A^* search algorithms with heuristic functions $h_1(s)$ and $h_2(s)$ respectively. We say that A_2^* is more informed than A_1^* if

$$h_1(s) \leq h_2(s) \leq h^*(s); \quad \forall s \in \Omega.$$

Prove that the nodes searched by A_2^* is always a subset of those searched by A_1^*. In other words, the CLOSED list of A_2^* is a subset of the CLOSED list of A_1^*.