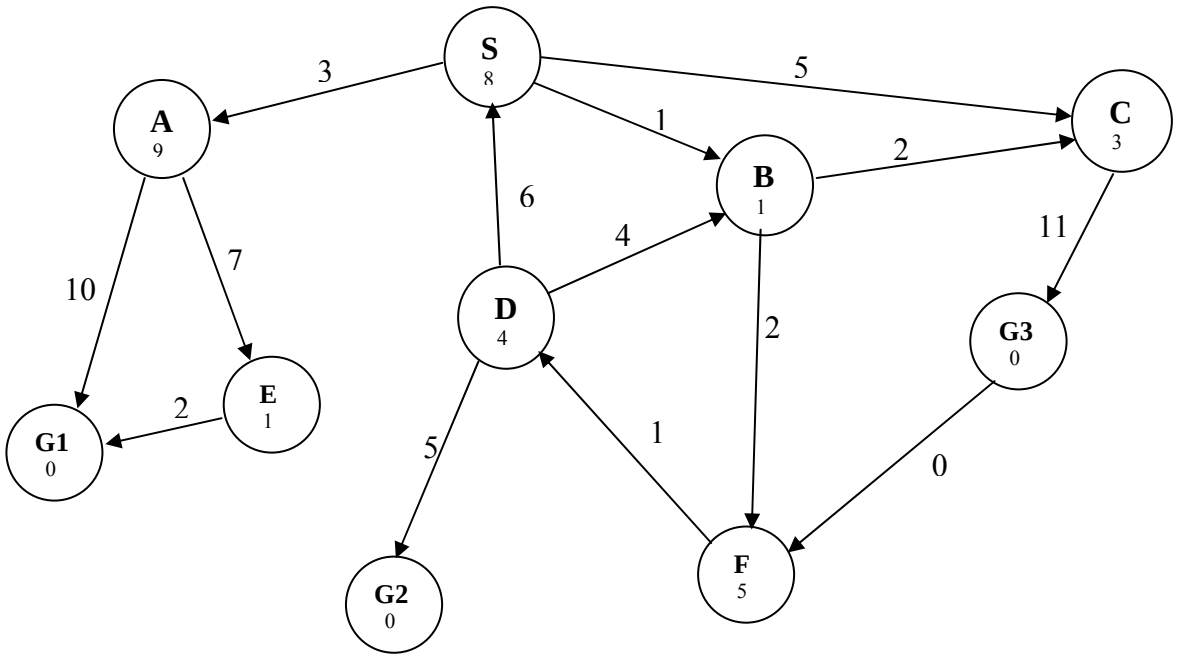
NAME: _____

CISC4/681 – Midterm Exam – Fall 2008

1. (30 possible) _____
2. (13 possible) _____
3. (12 possible) _____
4. (25 possible) _____
5. (20 possible) _____

Search (30 Points)

Consider the search space below, where S is the start node and G1, G2, and G3 all satisfy the goal test predicate. Arcs are labeled with the cost of traversing them and the estimated cost to a goal is reported inside nodes.



For each of the following search strategies, indicate which goal state is reached (if any) and list, *in order*, all the states *popped off of the OPEN list and made expl.*. For each search, indicate whether it guarantees an optimal solution if the heuristic function is admissible.

a) (5 points) Greedy Best First Search

b) (10 points) Algorithm A* Search. Suggestion: show the node properties and how they are updated during the search.

c) (10 points) Iterative Deepening A* Search. Suggestion: on open list nodes show the parent (when needed) and the f-value.

d) (5 points) Steepest Ascent Hill Climbing Search (remember, lower heuristics are better).

2. (13 points) Assume you are given three *admissible h* functions for a given search problem; for simplicity, call them *h1, h2,* and *h3*. For each of the following combinations, indicate whether or not the resulting *h* function is admissible and justify your response.

i) (3 points) $h(n) = h1(n) + h2(n) + h3(n)$

admissible? _____

why?

ii) (3 points) $h(n) = \frac{h1(n) + h2(n) + h3(n)}{3}$

admissible? _____

why?

iii) (7 points) Propose and justify a combination of *h1, h2,* and *h3* that, when used by A* as its *h* function, is guaranteed to lead to the expansion of no more nodes than the smallest number expanded by A* if it *individually* used each of *h1, h2,* and *h3*.

3. (12 points) The *heuristics path algorithm* is a best-first search in which the objective function is

$$f(n) = ((2 - w) * g(n)) + (w * h(n))$$

for integer $w \geq 0$. Assuming $h(n) \leq h'(n)$ for every node n in the graph. Where $h'(n)$ is the actual cost of a minimal cost path from n to a goal node.

a) What kind of search does the algorithm perform when w = 0?

b) What kind of search does the algorithm perform when w = 1?

c) What kind of search does the algorithm perform when w = 2?

d) For what values of w is the algorithm guaranteed to be optimal?

4. (25 points) Consider a search space with a path cost function g, and assume that h is an admissible heuristic for the space (i.e., h underestimates the actual cost of an optimal path from n to a goal). Thus we know that an Algorithm A* search which selects the nodes off of the open list according to:

$$f(n) = g(n) + h(n)$$

is guaranteed to find an optimal solution. Because of this, Algorithm A* is said to be admissible.

Show (using a brief proof) why algorithm A* remains admissible if it removes from OPEN any node n for which $f(n) > F$, where F is an upper bound on $f'(s)$. (Keep in mind that it simply removes n from OPEN, it does NOT put n on the CLOSED list.) So, it removes from the OPEN list any node whose f(n) is greater than F, where $F \geq f'(s)$. Recall that $f'(s)$ is the actual cost of a minimal cost path from s (the start node) to the goal which is cheapest to get to from s. HINT: think about what node(s) must not be deleted from the open list if the optimal path is to be found, and prove that it (they) won't be deleted.

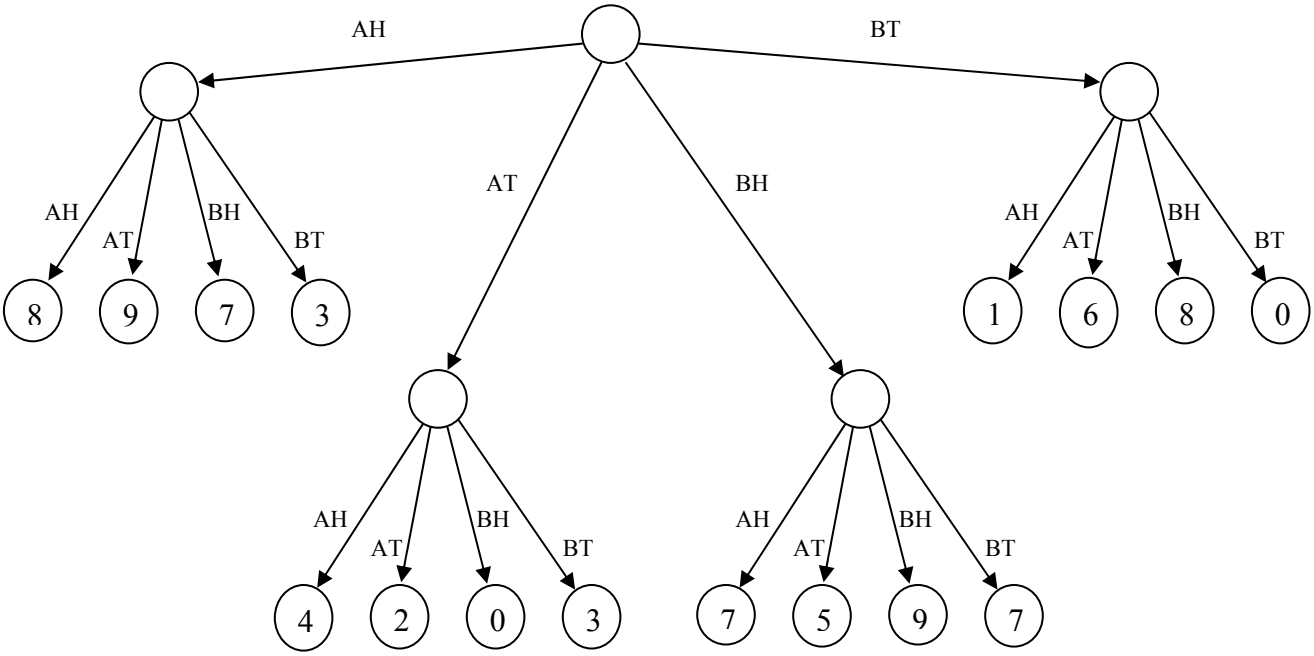
(This page if for your work.)

5. (20 points) Consider the following game:

> When it is their turn to move, players must first *choose which of two weighted coins, A and B, to flip.*
>
> Coin *A* comes up heads 75% of the time and tails the other 25%.
> If heads, the player *must* make move *AH* and if tails he or she (or it) *must* make move *AT*.
> (To do this problem, you needn't know exactly what each move means.)
>
> Coin *B* comes up heads 10% of the time and tails the other 90%.
> If heads, players *must* make move *BH* and if tails they *must* make move *BT*.

Assume it is the *computer's* turn to play, and the game tree looks like the one below, where the values at the leaf nodes are the results of calls to the *state evaluation function* (higher scores are better for the *computer*).

a) (9 points) Explain what move the *computer* should make.
(Hint: think about *expected-value* calculations. Also, you might want to do parts *b* and *c* first.)



b) (7 points) Now assume that *there is no randomness* and the players simply can choose *any* of the four moves (*AH, AT, BH*, or *BT*). Apply the *minimax* algorithm to the tree below and explain which move the computer should make. As in part (a), assume it is the computer's turn to play.

c) (4 points) Assuming leaf nodes are visited left-to-right, identify *the first* unnecessary call to the *state evaluation function* (for the *no randomness* case) because of an alpha or beta cutoff. In other words, show where the first cut-off would be. Explain your answer.