C1 Artificial Intelligence (25 Points)
Search

a) (12 Points) This question demonstrates different search methods using the flight routes of Colgan Airlines, a small East-coast carrier.

Table 1 contains the straight line distances between all of the cities. An asterisk in the table entry indicates that there is a direct route between the two cities (as indicated by the route map). Thus, this distance is the edge cost of that edge. The remaining distances do not correspond to edges in the search graph; the distance for routes between cities that are not directly connected is, as usual, the sum of all of the individual legs in the route.

In the following problems, you will execute various search algorithms on this domain. When doing your search, assume that a node's successors are added to the list in alphabetical order. Also assume that we eliminate repeated states only *along a path*, i.e., when expanding a node n, we prevent the search algorithm from inserting into the list any successor node of n which already appears on the path to n. We do not eliminate other repeated states.

For each of the following searches, draw the search tree generated by your search algorithm, in the following format: Your tree should contain all nodes inserted into the list, along with their priority (when applicable). Nodes that are expanded should be numbered consecutively, according to the order of expansion. (Note that, in the case of iterative deepening, nodes may be numbered more than once.)

(1) A breadth-first search from Hyannis, MA to Charlottesville, VA.

(2) A depth-first search for the same route.

(3) An iterative deepening search for the same route.

(4) A uniform cost search from Rockland, ME to Keene, NH.

Table 1: Distances between Colgan Air cities

	Aug	Bar	Bec	Blu	Bos	ChNC	ChVA	Hya	Kee	Lag	Man	Nan	New	Roc	Rut	DC
Augusta,ME	0	78	745	770	149*	858	626	185	157	328	552	210	333	36	166	529
Bar-Harbor,ME	78	0	814	838	200	920	690	216	227	389	616	235	394	49*	243	592
Beckley,WV	745	814	0	35*	621	177	149	637	588	434	212	638	427	765	589	238*
Bluefield,WV	770	838	35*	0	643	142*	159	656	613	454	228	655	448	789	616	254*
Boston,MA	149*	200	621	643	0	720	492	63*	74	190	418	89	196	155*	130	394
Charlotte,NC	858	920	177	142*	720	0	234	722	702	531	306	716	525	873	714	329
Charlottesville,VA	626	690	149	159	492	234	0	501	469	302*	74*	498	296	642	481	98
Hyannis,Ma	185	216	637	656	63*	722	501	0	135	204*	429	27*	212*	179	192	403
Keene,NH	157	227	588	613	74	702	469	135	0	177	396	159	180*	178	58*	373
La-Guardia,NY,NY	328	389	434	454	190	531	302*	204*	177	0	229	207*	9	342	207	204
Manassas,VA	552	616	212	228	418	306	74*	429	396	229	0	427	223	569	409	26*
Nantucket,MA	210	235	638	655	89	716	498	27*	159	207*	427	0	216*	201	217	402
Newark,NJ	333	394	427	448	196	525	296	212*	180*	9	223	216*	0	347	208	198
Rockland,ME	36	49*	765	789	155*	873	642	179	178	342	569	201	347	0	195	545
Rutland,VT	166	243	589	616	130	714	481	192	58*	207	409	217	208	195	0	388
Washington, DC	529	592	238*	254*	394	329	98	403	373	204	26*	402	198	545	388	0

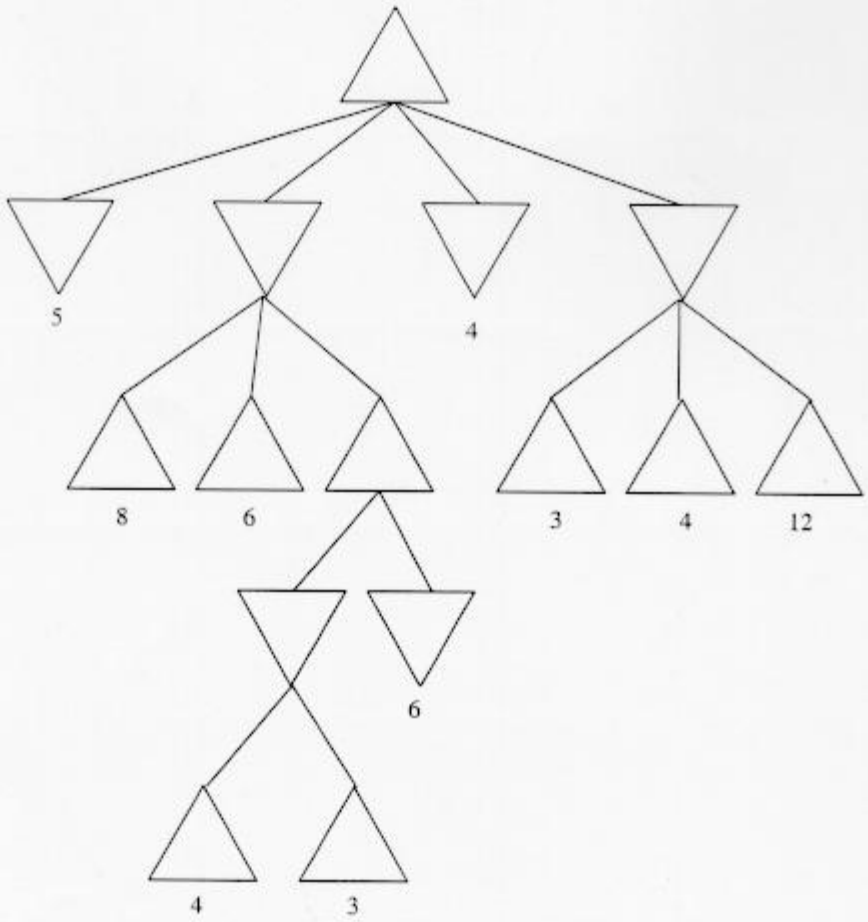
C1 Artificial Intelligence (Continued)

b) (13 points) Assume we are given a heuristic function which is only approximately admissible. A heuristic function is **approximately admissible** if there exists some $\varepsilon > 0$ such that for every node n in the tree, we have $h(n) <= h^*(n) + \varepsilon$ where $h^*(n)$ is the true cost of getting from n to the goal. Note that in this case it is possible for h(n) for a goal node n to be bigger than zero. Let c^* be the cost of the optimal path from start to the optimal goal.

Assume that we run standard A* with our ε-admissible heuristic h. Let n_g be the goal node that A* returns. Prove that the cost $g(n_g)$ is at most $c^* + \varepsilon$. You may assume that the function $f = g + h$ is monotonic. (Recall that if f is monotonic then A* expands nodes in increasing f-value.) HINT: you may use any results which have been proven about regular A* search. This should be about a four-line proof.

C2 Artificial Intelligence (25 Points)
Game Trees

a) (8 points) The figure below represents a two-player game tree. The upward-pointing triangles (such as the root) correspond to the maximizing nodes (and belong to the player who is attempting to maximize the score). The downward-pointing nodes are min nodes (and represent the opponent's moves). Using mini-max search with alpha-beta pruning, calculate the value for the root node, evaluating nodes from left to right. Write the backed-up values in each of the nodes in the figure, and be sure to indicate which nodes are pruned.



C2 Artificial Intelligence (Continued)

b) (13 points) Repeated States in α–β Pruning

Consider a game where the same game state (for example, the board position and player to move in chess) can be reached via many different paths in the tree. Thus, different nodes in the game tree can represent the same game state.

Suppose that we use α–β pruning to search the game tree, and decide that some successors of node n can be pruned. When searching another part of the same tree (while deciding on the same move, not later in the game), we encounter the sam game-state in node n'. (In other words, both n and n' represent two different ways of getting to the same state of the game and both of these nodes appear in the subtree on which we are executing the α–β algorithm.) Can we automatically prune the same successors o n' from our search tree? If so, prove it. If not, provide a counterexample (in an abstract game, not in chess).

For the purposes of this question, we assume that the tree is evaluated all the way to the end, and not to a fixed depth. Hence, your answer should not depend on the tree being evaluated to a limited depth. Also note that the game state includes both the game position and the player to move. Hence, two states in which the game position is the same but the turn belongs to a different player do not coun as the same state.

C3 Artificial Intelligence (25 points)

Planning

In STRIPS, every operator has fixed sets of preconditions, add effects, and delete effects. This formulation makes it difficult to represent the notion that different actions take differing amounts of time. In this problem, we explore this difficulty.

The bike club has just bought a new, snazzy bicycle that goes twice as fast as their old, rusty one. Carlos, as head bicycle racer, has access to both bikes. If Carlos rides the new bike, then after one timestep, he will be at his new location and ready to race to another one. Yet, if he rides the old bike, he is only ready to ride from his new location after two timesteps. We presume that all locations are equidistant from each other—riding between any two on the same bicycle takes the same amount of time. For every pair of locations *FromLocation* and *ToLocation*, we have actions *StartRidingNewBike(Person,FromLocation,ToLocation)* and *StartRidingOldBike(Person,FromLocation,ToLocation)*.

(a) [10 points] Write preconditions, and add/delete effects for these actions that yield the desired behavior. You will need to use the predicate *At(object,location)* with objects *NewBike*, *OldBike*, and *Carlos*. You may create new actions and predicates, if necessary (if you create a new action, define its preconditions and effects as well).

(b) [5 points] It would be much nicer to simply have one action, *StartRidingBike(Person,Bike,FromLocation,ToLocation)*. Is it possible to write a standard STRIPS specification for this action? Provide a one-sentence justification.

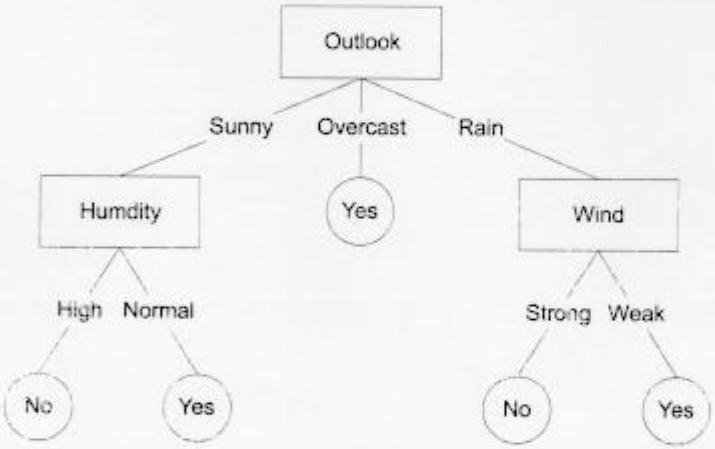
(c) [10 points] Let *Pre(a)*, *Add(a)*, and *Del(a)* represent the sets of Preconditions, Add Effects, and Delete Effects for an arbitrary STRIPS action *a*. Now imagine that we want to do planning in a multi-agent domain. Since agents can work at the same time, we can have many actions executing in parallel. Assume we have an unlimited number of agents, and that we will build our plans in phases. All actions in phase *i* are started after all actions in phase *(i-1)* are complete, and are themselves guaranteed to complete before any actions in phase *(i+1)* are started. Let S_i be the set of all actions in phase i. All that we know is that all these actions start and end sometime during phase *i*. What constraints must hold on the actions in S_i to guarantee that, no matter what order the actions in S_i are executed, the result of applying all the actions to the state resutling from phase *(i-1)* is the same (for the purposes of planning). Hint: a correct answer can be stated in two sentences or less.

C4 Artificial Intelligence (25 points)

Learning

In this question we assume that there is some decision tree generating the data. This question investigates whether we can accurately reconstruct the original decision tree by using the decision tree learning algorithm on data generated from the original tree.

Suppose Beatrice uses the following decision tree to decide whether she will practice her archery:



We wish to reconstruct this decision tree by observing when Beatrice does and does not go to practice her archery. Suppose we gather the following observations over the course of two weeks:

Day	Outlook	Temperature	Humidity	Wind	Practice Archery
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

C4 Artificial Intelligence ***(continued)***

Notice that ID3 (the decision tree learning algorithm presented in class that uses information gain to split on attributes) actually learns the original tree perfectly when presented with these 14 training instances. In the following questions we will require that all of the training data are consistent with Beatrice's original tree, meaning the PracticeArchery label on each of the training data should be what Beatrice herself would do if she were to observe those settings of the attributes. (Recall that we assumed that Beatrice is using the tree shown above.) Duplicate training examples are allowed; they count as separate individual examples in the information gain computations.

(a) [5 points] Is it possible for a training set to get ID3 to learn a tree that is identical to Beatrice's tree except that the learned tree further elaborates the tree below the rightmost leaf? Justify your answer.

(b) [10 points] Is it possible to add more training examples to the original 14 examples listed above that will cause ID3 to initially split on the attribute Temperature at the root node even though the original tree that Beatrice uses is independent of Temperature? You should explain your answer at the level of arguing about information gain, either by using formulae or explaining clearly in words.

(c) [10 points] We will say that a tree is *incorrect* if there is a setting of the attributes *of {Outlook, Temperature, Humidity, Wind}* that will cause this tree to classify *Practice Archery* differently from Beatrice's tree. Is it possible to get ID3 to learn an incorrect tree by adding new correct examples to the original fourteen? Justify your answer. Hint: you may like to see whether your answer to part (b) can help you answer this question.