Midterm Examination

CS540: Introduction to Artificial Intelligence

October 24, 2019

LAST NAME: ________________________________

FIRST NAME: ________________________________

Directions

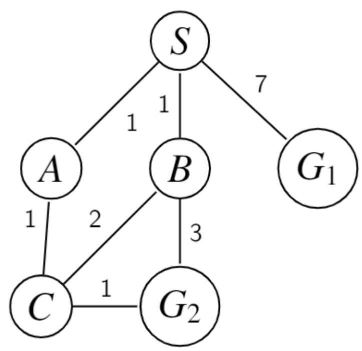
1. This exam contains **32** questions worth a total of 100 points
2. Fill in your **name** and **student ID number** carefully on the answer sheet
3. **Fill in each oval that you choose *completely***; do *not* use a check mark, an "X," or put a box or circle around the oval
4. Fill in the ovals with **pencil**, not pen
5. If you change an answer, be sure to completely erase the old filled-in oval
6. Fill in *only one oval* for each question
7. When you answer a question, be sure to check and make sure that the question number on the answer sheet matches the question number that you are answering on the exam
8. For True / False questions, fill in **A for True and B for False**
9. There is no penalty for guessing

Uninformed and Informed Search

1. [4] Which of the following statements is/are true about a **heuristic function** *h*?
   (i) If *h*(*n*) = *h**(*n*) for all *n*, then algorithm A* will *only* expand nodes on the optimal path (ignoring ties).
   (ii) If *h* is admissible, the smaller *h*(*n*) is, the fewer nodes that A* will expand.
   (iii) If *h*(*n*) is always less than or equal to the cost of the cheapest path from *n* to the goal, then A* is guaranteed to find an optimal solution.

   A. Only (i) is true
   B. Only (ii) is true
   C. Only (iii) is true
   D. Both (i) and (iii) are true
   E. All (i), (ii), and (iii) are true

2. [4] Which goal is reached *and* what is the total cost of the solution found for the following state-space graph when using **Breadth-First Search** and **Uniform-Cost Search** (*S* is the start state, *G*1 and *G*2 are the goal states, arcs are bidirectional, no repeated state checking, break any ties alphabetically)?



   A. BFS: *G*1 (Cost: 7), UCS: *G*2 (Cost: 4)
   B. BFS: *G*2 (Cost: 4), UCS: *G*1 (Cost: 7)
   C. BFS: *G*2 (Cost: 4), UCS: *G*2 (Cost: 4)
   D. BFS: *G*1 (Cost: 7), UCS: *G*2 (Cost: 3)
   E. BFS: *G*1 (Cost: 7), UCS: *G*1 (Cost: 7)

3. [2] True or False: If you are given a heuristic function *h*, such that for every state *n*, *h(n)* = *h**(*n*) (the optimal cost of moving from *n* to the goal), then using **Greedy Best-First Search** with this heuristic will always find an optimal solution.

4. [2] True or False: If h_1 and h_2 are two ***admissible*** heuristics for a given problem, then heuristic $h_3(n) = 2\,h_1(n) - h_2(n)$ for all states, *n*, must also be admissible.

5. [2] True or False: If we use a ***consistent*** heuristic with A* search, then when a node is expanded and put on *Explored*, we can guarantee that we have already reached that node's state via the minimum-cost path from the start state.

6. [2] True or False: If we know there is a non-optimal solution with cost C, then *any* node generated by the A* algorithm that has *f*(*n*) = *g*(*n*) + *h*(*n*) > C does *not* need to be put on *Frontier* (i.e., it can be thrown away) and A* will still find an *optimal* solution.

For the next ***three*** questions, say we define an evaluation function for a heuristic search problem as: $f(n) = (w * g(n)) + ((1 - w) * h(n))$
where $g(n)$ is the cost of the best path found from the start state to state n, $h(n)$ is an admissible heuristic function that estimates the cost of a path from n to a goal state, and $0.0 \leq w \leq 1.0$. What search algorithm do you get when:

7. [3] $w = 0.0$

   A. Breadth-First search
   B. Uniform-Cost search
   C. Greedy Best-First search
   D. Algorithm A* search
   E. None of the above

8. [3] $w = 0.5$

   A. Breadth-First search
   B. Uniform-Cost search
   C. Greedy Best-First search
   D. Algorithm A* search
   E. None of the above

9. [3] $w = 1.0$

   A. Breadth-First search
   B. Uniform-Cost search
   C. Greedy Best-First search
   D. Algorithm A* search
   E. None of the above

Local Search

10. [2] True or False: **Hill-climbing** *can* escape a local optimum when there are multiple optima.

11. [2] True or False: **Simulated Annealing** with a constant, positive temperature at all times is the *same* as **Hill-Climbing**.

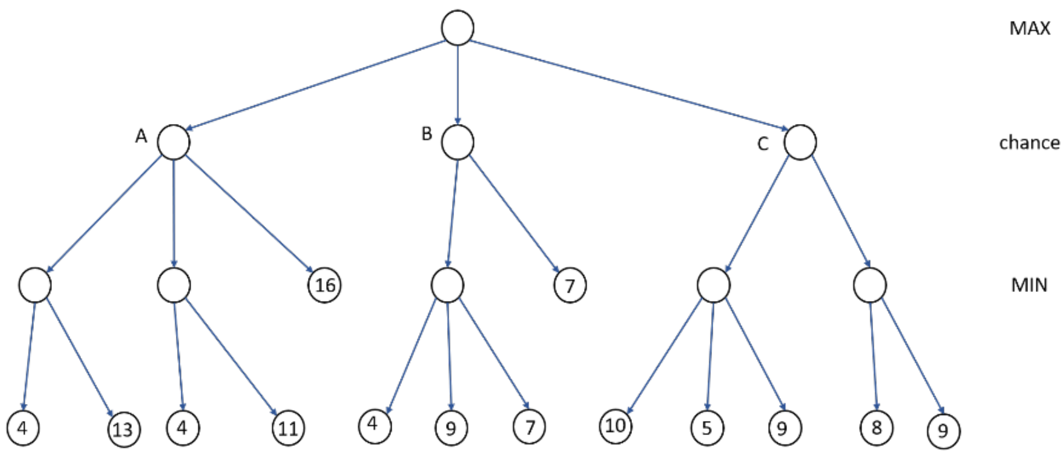
12. [4] What kind of search *best* describes what **Simulated Annealing** does (approximately) if the temperature is very large (i.e., close to ∞) at *every* iteration?

    A. It will halt immediately and do no search
    B. Breadth-First search
    C. Depth-First search
    D. Hill-Climbing
    E. It will move to a randomly selected successor state at each iteration

Game Playing

13. [2] True or False: No matter what the static board evaluation (SBE) function values are at the leaves of a search tree that is explored using **Alpha-Beta search** (assume child nodes are explored left to right), the *leftmost* child of every explored node can *never* be pruned.

14. [4] For the zero-sum game tree below find the *sum* of the **Expectiminimax** values computed at the three *chance nodes*, A, B and C. For each chance node, assume that the probability of taking leftmost move is *twice* as much as taking any other move. The probability of taking other moves of a chance node (for example, the middle and the rightmost moves from node A) are equally likely.



A. 48/7
B. 13
C. 18
D. 20
E. 31

15. [4] For the above game tree what is the **Expectiminimax** value at the root?

A. 4
B. 5
C. 7
D. 8
E. 16

16. [3] Which of the following methods is the *main* way to avoid the **horizon effect**?

A. Run Alpha-Beta search with an increasing depth-limit (iterative-deepening search)
B. When the SBE value is frequently changing, look deeper than the depth-limit
C. For each game state, consider only the *n* best moves (according to the SBE function) rather than considering all possible moves
D. Use Expectiminimax to calculate the value of non-terminal game states
E. None of the above

17. [2] True or False: The **Minimax** algorithm using static board evaluation (SBE) function f_1 is guaranteed to choose the *same* next move as the Minimax algorithm using SBE function f_2 when $f_2(n) = f_1(n) + c$, for all states, *n,* in a game tree, and *c* is a positive, real-valued constant.

Hierarchical Agglomerative Clustering

For the next ***three*** questions, consider a dataset containing six one-dimensional points: {2, 4, 7, 8, 12, 14}. After three iterations of **Hierarchical Agglomerative Clustering** using Euclidean distance between points, we get the 3 clusters: C_1 = {2, 4}, C_2 = {7, 8} and C_3 = {12, 14}.

18. [4] What is the distance between clusters C_1 and C_2 using **Single Linkage**?

A. 2
B. 3
C. 4
D. 5
E. 6

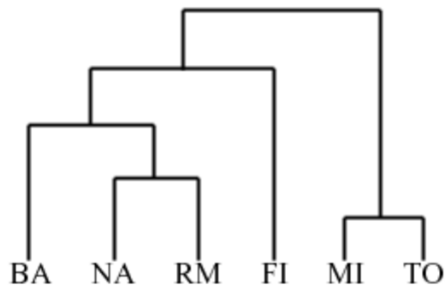
19. [4] What is the distance between clusters C_1 and C_2 using **Complete Linkage**?

A. 2
B. 3
C. 4
D. 5
E. 6

20. [4] What clusters are merged at the next iteration using **Single Linkage**?

A. C_1 and C_2
B. C_2 and C_3
C. C_1 and C_3
D. C_1, C_2 and C_3
E. No merging occurs because the algorithm terminates

21. [4] Consider the **dendrogram**:



Using this dendrogram to create 3 clusters, what would the clusters be?

A. {BA, NA}, {RM, FI}, {MI, TO}
B. {NA, RM}, {BA, FI}, {MI, TO}
C. {BA, NA, RM, FI}, {MI}, {TO}
D. {BA, NA, RM}, {FI}, {MI, TO}
E. None of these

k-Means Clustering

22. [4] You want to cluster 7 points into 3 clusters using the ***k*-Means Clustering** algorithm. Suppose after the first iteration, clusters C_1, C_2 and C_3 contain the following two-dimensional points:

C_1 contains the 2 points: {(0,6), (6,0)}
C_2 contains the 3 points: {(2,2), (4,4), (6,6)}
C_3 contains the 2 points: {(5,5), (7,7)}

What are the **cluster centers** computed for these 3 clusters?

A. C_1: (3,3), C_2: (4,4), C_3: (6,6)
B. C_1: (3,3), C_2: (6,6), C_3: (12,12)
C. C_1: (6,6), C_2: (12,12), C_3: (12,12)
D. C_1: (0,0), C_2: (48,48), C_3: (35,35)
E. None of these

23. [2] True or False: In general (not for the dataset above), it is possible that after new cluster centers are computed by the *k*-Means Clustering algorithm, a cluster center may be associated with an empty cluster (i.e., with zero points in it).

24. [2] True or False: To find the best number of clusters, *k*, to use with *k*-Means Clustering for a given dataset, you should pick the value of *k* that *minimizes* the ***distortion*** measure of cluster quality.

k-Nearest-Neighbors Classifier

25. [4] The table below shows a training set with 10 examples that is used for training a **3-nearest-neighbors classifier** that uses Manhattan distance, i.e., the distance between two points at coordinates *p* and *q* is $|p - q|$. The only attribute, *X*, is real-valued, and the label *Y* has two possible classes, 0 and 1. What is the **2-fold cross validation accuracy** (percentage correct classification)? The first fold contains the first 5 examples, and the second fold contains that last 5 examples. In case of ties in distance, use the example with smallest *X* value as the neighbor.

X	0	1	2	3	4	5	6	7	8	9
Y	1	0	1	0	1	0	1	0	1	0

A. 0 percent
B. 20 percent
C. 40 percent
D. 60 percent
E. 100 percent

26. [4] The table below shows the test set for a **1-nearest-neighbor classifier** that uses Manhattan distance, i.e., the distance between two points at coordinates *p* and *q* is $|p - q|$. The only attribute, *X*, is real-valued, and the label, *Y*, has two classes, 0 and 1. Suppose a *subset* containing $n \leq 8$ examples is selected from this set to train the classifier, and the accuracy of the classifier is 100 percent when tested on this set (with *all* 8 examples). What is the *smallest* possible value for *n*? In case of ties in distance, use the example with smallest *X* value as the neighbor.

X	-5	-4	-1	0	1	3	4	8
Y	0	1	0	0	0	0	0	1

A. 2
B. 3
C. 4
D. 5
E. 6

Decision Trees

27. [3] Which one of the following is the *main* reason for **pruning a Decision Tree**?

A. To save computing time during testing
B. To save space for storing the Decision Tree
C. To make the training set error smaller
D. To avoid overfitting the training set
E. To increase the information gain at the root of the Decision Tree

For the next ***three*** questions, use the table below that defines a training set containing 4 examples. The two attributes, X_1 and X_2, and the class label, Y, are all binary.

X_1	0	0	1	1
X_2	0	1	0	1
Y	1	1	1	0

28. [4] What is the **entropy** of Y, i.e., $H(Y)$? Use $\log_2 0.25 = -2$, $\log_2 0.5 = -1$, $\log_2 0.75 = -0.4$, $\log_2 1 = 0$, $\log_2 2 = 1$, $\log_2 4 = 2$, and the convention that $0 \log_2 0 = 0$.

    A. 0
    B. 0.3
    C. 0.5
    D. 0.8
    E. 1

29. [4] What is the **conditional entropy** of Y given X_1, i.e., $H(Y \mid X_1)$? Use $\log_2 0.25 = -2$, $\log_2 0.5 = -1$, $\log_2 0.75 = -0.4$, $\log_2 1 = 0$, $\log_2 2 = 1$, $\log_2 4 = 2$, and the convention that $0 \log_2 0 = 0$.

    A. 0
    B. 0.25
    C. 0.5
    D. 0.75
    E. 1

30. [4] Using the above training set, a **Decision Tree** is built that contains only 3 nodes: the root and its 2 children. Each leaf node is assigned the *majority* class of its associated set of examples; break ties in favor of $Y = 0$. What is the **classification accuracy** of this Decision Tree on the *training set* of 4 examples?

    A. 0 percent
    B. 25 percent
    C. 50 percent
    D. 75 percent
    E. 100 percent

31. [2] True or False: Given a binary attribute, A, that splits a set of examples, E, into 2 *nonempty* subsets E_1 and E_2 such that E_1 has $p_1 > 0$ positive examples and $n_1 > 0$ negative examples for a binary class label C, and E_2 has $p_2 > 0$ positive examples and $n_2 > 0$ negative examples, it *is possible* for **information gain** $I(C; A) = 0$.

32. [3] Under which of the following conditions is ***k*-fold cross-validation** the *same as* **leave-one-out cross-validation**?

    A. The training set and test set have the *same* number of examples
    B. The training set and tuning set have the *same* number of examples
    C. $k = 1$
    D. $k = n$, where n is the total number of examples
    E. None of the above