Assignment 2 - Search

Deadline: March 1, 11:55pm.
Perfect score: 100.

Assignment Instructions:

Teams: Assignments should be completed by teams of two students. No additional credit will be given for students that complete an assignment individually. Please inform the TAs as soon as possible about the members of your team so they can update the scoring spreadsheet (find the TAs' contact info under the course's website: `http://www.cs.rutgers.edu/~jy512/s16/CS520S16.html`).

Submission Rules: Submit your reports electronically as a PDF document through Sakai (`sakai.rutgers.edu`). For programming questions, you need to also submit a compressed file via Sakai, which contains your code. Do not submit Word documents, raw text, or hardcopies etc. Make sure to generate and submit a PDF instead. Each team of students should submit only a single copy of their solutions and indicate all team members on their submission. Failure to follow these rules will result in lower grade in the assignment.

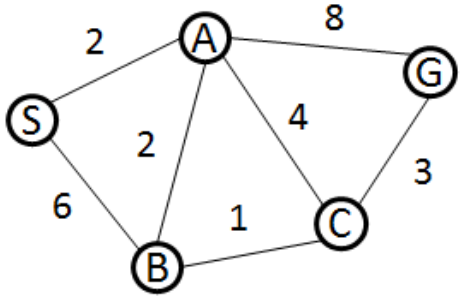
Program Demonstrations: You will need to demonstrate your program to the TAs on a date after the deadline,. The schedule will be coordinated by the TAs. During the demonstration you have to use the file submitted on Sakai and execute it either on a university machine at CORE/HILL center (one of the CS labs) or on your laptop computer. You will also be asked to describe the architecture of your implementation and key algorithmic aspects of the project. You need to make sure that you are able to complete the demonstration and answer the TAs' questions within the allotted 10 minutes of time for each team. If your program is not directly running on the computer you are using and you have to spend time to configure your computer, this counts against your allotted time.

Late Submissions: No late submission is allowed. 0 points for late assignments.

Precision: Try to be precise. Have in mind that you are trying to convince a very skeptical reader (and computer scientists are the worst kind...) that your answers are correct.

Collusion, Plagiarism, etc.: Each team must prepare its solutions independently from other teams, i.e., without using common notes, code or worksheets with other students or trying to solve problems in collaboration with other teams. You must indicate any external sources you have used in the preparation of your solution. Do not plagiarize online sources and in general make sure you do not violate any of the academic standards of the department or the university. Failure to follow these rules may result in failure in the course.

Problem 1 [15 points]: Fig. 1 provides a graph with associated edge costs and a table with values of the a heuristic function. The adjacency list for a node is ordered alphabetically, i.e., node A has the adjacency list $\langle B, C, G, S \rangle$. Using the tree-search and graph-search algorithms from our class (also provided below), manually execute BFS, DFS, uniform-cost, greedy best-first, and A^* to find a path from S to G. For each of the 10 combinations, provide as your answer a search tree, the nodes on the queue (frontier), and the cost of the associated solution. For tree-search, if an infinite loop is encountered, you only need to provide a search tree with enough depth to demonstrate the infinite loop. You may draw your search tree manually and scan them for submission.



State	$h(x)$
S	7
A	6
B	2
C	1
G	0

Figure 1: A graph with associated edge costs and heuristics.

```
1  Tree-Search(G, x_I)
2  AddToQueue(x_I, Queue);
3  while(!IsEmpty(Queue))
4     x ← Front(Queue);
5     if(x is goal) return solution;
6     for each successor n_i of x
7        AddToQueue(n_i, Queue);
8  return failure;
```

Figure 2: Pseudocode for tree-search.

```
1  Graph-Search(G, x_I)
2  AddToQueue(x_I, Queue);
3  while(!IsEmpty(Queue))
4     x ← Front(Queue);
5     if(x.expanded) continue;
6     x.expanded ← true;
7     if(x is goal) return solution;
8     for each successor n_i of x
9        if(!n_i.expanded) AddToQueue(n_i, Queue);
10 return failure;
```

Figure 3: Pseudocode for graph-search.

Problem 2 [20 points]: Answer the following questions on informed search and heuristics.

a) [3 + 3 + 3 = 9 points] Which of the following are admissible, given admissible heuristics h_1, h_2? Which of the following are consistent, given consistent heuristics h_1, h_2? (hint: start with definitions of admissible heuristics and consistent heuristics)

(i) $h_{min}(n) = min\{h_1(n), h_2(n)\}$,

(ii) $h_{max}(n) = max\{h_1(n), h_2(n)\}$,

(iii) $h_{lin}(n) = wh_1(n) + (1-w)h_2(n), 0 \leq w \leq 1$.

b) [6 points] Consider an informed, best-first search algorithm in which the objective function is $f(n) = (2-w)g(n) + wh(n)$. For what values of w is this algorithm guaranteed to be optimal when h is admissible (consider the case of

TREE−SEARCH)? What kind of search does this perform when $w = 0$? When $w = 1$? When $w = 2$? (hint: it may help to think about the later questions first)

c) [5 points] A^* search uses $f(x) = g(x) + h(x)$. Consider a modified version of A^*, A^*_ϵ, that allows more flexibility in picking the node for expansion. In choosing the next node for expansion, unlike A^*, A^*_ϵ expands a node with $f(n) \leq (1 + \epsilon) \min_{n' \in Queue}\{f(n')\}$, in which ϵ is an arbitrary positive number. Assuming that $h(x)$ is admissible, formally argue about the cost of the solution returned by A^*_ϵ.

Problem 3 [15 points]: Consider the two-player game described in Figure 4.

a. [3 points] Draw the complete game tree, using the following conventions:

- Write each state as (s_A, s_B) where s_A and s_B denote the token locations.
- Put each terminal state in a square box and write its game value in a circle.
- Put *loop states* (states that already appear on the path to the root) in double square boxes. Since it is not clear how to assign values to loop states, annotate each with a "?" in a circle.

b. [3 points] Now mark each node with its backed-up MinMax value (also in circle). Explain how you handled the "?" values and why.

c. [3 points] Explain why the standard MinMax algorithm would fail on this game tree and briefly sketch how you might fix it, drawing on your answer to (b). Does your modified algorithm give optimal decisions for all games with loops?

d. [6 points] This 4-square game can be generalized to n squares for any $n > 2$. Prove that A wins if n is even and loses if n is odd.

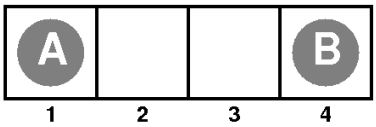


Figure 4: The start position of a simple game. Player A moves first. The two players take turns moving, and each player must move his token to an open adjacent space in *either direction*. If the opponent occupies an adjacent space, than a player may jump over the opponent to the next open space if any. (For example, if A is on 3 and B is on 2, then A may move back to 1.) The game ends when one player reaches the opposite end of the board. If player A reaches space 4 first, then the value of the game to A is +1; if player B reaches space 1 first, then the value of the game to A is -1.

Problem 4 [10 points]: Consider the problem of constructing (not solving) crossword puzzles: fitting words into a rectangular grid. The grid, which is given as part of the problem, specifies which squares are blank and which are shaded. Assume that a list of words (i.e., a dictionary) is provided and that the task is to fill in the blank squares using any subset of the list. Formulate the problem in two ways (see items a and b below) (hint: there might be multiple correct answers here)

a) [4 points] As a general search problem. Choose an appropriate algorithm. Is it better to fill in blanks one letter or one word at a time?

b) [4 points] As a constraint satisfaction problem. Should the variables be words or letters?

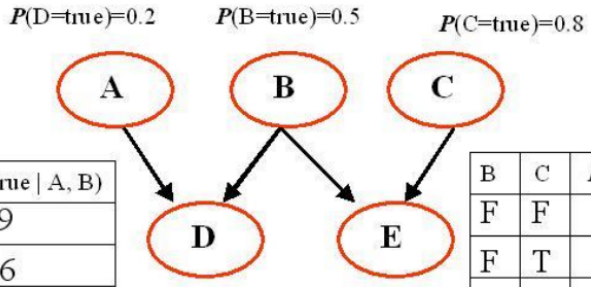
c) [2 points] Which formulation do you think will be better? Why?

Problem 5 [10 points]: Consider the following sequence of statements, which describe the operation of a security system.

If the system is armed and motion is detected, then the alarm will sound. If the alarm sounds, then the system has been armed or there has been a fire. Regardless of whether the system is armed, the alarm should go off when there is a fire. Motion is constantly detected.

a. [4 points] Convert the above statements into a knowledge base using the symbols of propositional logic.

b. [6 points] Using the resolution rule and your knowledge base, derive the following theorem: *The alarm will sound if and only if the system is armed or there is a fire*.



A	B	P(D=true | A, B)
F	F	0.9
F	T	0.6
T	F	0.5
T	T	0.1

B	C	P(E=true | B, C)
F	F	0.2
F	T	0.4
T	F	0.8
T	T	0.3

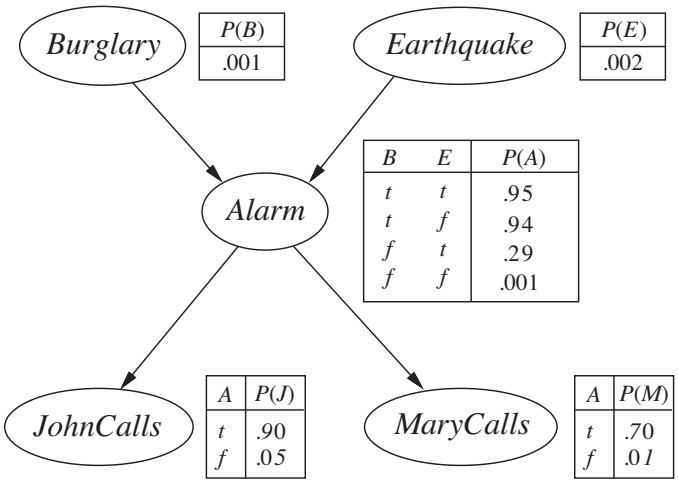
Problem 6 [15 points]: Consider the following Bayesian network, where variables A through E are all Boolean valued:

a) [5 points] What is the probability that all five of these Boolean variables are simultaneously true? (hint: you have to compute the joint probability distribution. The structure of the Bayesian network suggests how the joint probability distribution is decomposed to the conditional probabilities available).

b) [5 points] What is the probability that all five of these Boolean variables are simultaneously false?

c) [5 points] What is the probability that A is false given that the four other variables are all known to be true?

Problem 7 [15 points]: For this problem, check the Variable Elimination algorithm in your book. Also consider the Bayesian network from the "burglary" example.



a) [5 points] Apply variable elimination to the query:

$$P(Burglary|JohnsCalls = true, MaryCalls = true)$$

and show in detail the calculations that take place. Use your book to confirm that your answer is correct.

b) [5 points] Count the number of arithmetic operations performed (additions, multiplications, divisions), and compare it against the number of operations performed by the tree enumeration algorithm.

c) [5 points] Suppose a Bayesian network has the from of a *chain*: a sequence of Boolean variables $X_1, \ldots X_n$ where $Parents(X_i) = \{X_{i-1}\}$ for $i = 2, \ldots, n$. What is the complexity of computing $P(X_1|X_n = true)$ using enumeration? What is the complexity with variable elimination?