CS 188
Fall 2013

Introduction to Artificial Intelligence — Midterm 1

- You have approximately 2 hours and 50 minutes.
- The exam is closed book, closed notes except your one-page crib sheet.
- Please use non-programmable calculators only.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation. All short answer sections can be successfully answered in a few sentences AT MOST.

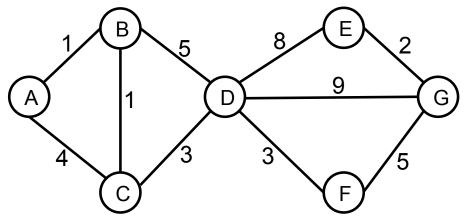
First name	
Last name	
SID	
edX username	

First and last name of student to your left	
First and last name of student to your right	

For staff use only:

Q1.	Search	/9
Q2.	InvisiPac	/9
Q3.	CSPs	/27
Q4.	Utilities	/10
Q5.	Games: Three-Player Cookie Pruning	/9
Q6.	The nature of discounting	/10
Q7.	The Value of Games	/10
Q8.	Infinite Time to Study	/16
	Total	/100

Q1. [9 pts] Search



Node	h_1	h_2
A	9.5	10
B	9	12
C	8	10
D	7	8
E	1.5	1
F	4	4.5
G	0	0

Consider the state space graph shown above. A is the start state and G is the goal state. The costs for each edge are shown on the graph. Each edge can be traversed in both directions. Note that the heuristic h_1 is consistent but the heuristic h_2 is not consistent.

(a) [4 pts] **Possible paths returned**

For each of the following graph search strategies (*do not answer for tree search*), mark which, if any, of the listed paths it could return. Note that for some search strategies the specific path returned might depend on tie-breaking behavior. In any such cases, make sure to mark *all* paths that could be returned under some tie-breaking scheme.

Search Algorithm	**A-B-D-G**	**A-C-D-G**	**A-B-C-D-F-G**
Depth first search	x	x	x
Breadth first search	x	x	
Uniform cost search			x
A* search with heuristic h_1			x
A* search with heuristic h_2			x

The return paths depend on tie-breaking behaviors so any possible path has to be marked. DFS can return any path. BFS will return all the shallowest paths, i.e. A-B-D-G and A-C-D-G. A-B-C-D-F-G is the optimal path for this problem, so that UCS and A* using consistent heuristic h_1 will return that path. Although, h_2 is not consistent, it will also return this path.

(b) Heuristic function properties

Suppose you are completing the new heuristic function h_3 shown below. All the values are fixed except $h_3(B)$.

Node	A	B	C	D	E	F	G
h_3	10	?	9	7	1.5	4.5	0

For each of the following conditions, write the set of values that are possible for $h_3(B)$. For example, to denote all non-negative numbers, write $[0, \infty]$, to denote the empty set, write $\varnothing$, and so on.

(i) [1 pt] **What values of $h_3(B)$ make h_3 admissible?**

To make h_3 admissible, $h_3(B)$ has to be less than or equal to the actual optimal cost from B to goal G, which is the cost of path B-C-D-F-G, i.e. 12. The answer is $0 \leq h_3(B) \leq 12$

(ii) [2 pts] **What values of $h_3(B)$ make h_3 consistent?**

All the other nodes except node B satisfy the consistency conditions. The consistency conditions that do involve the state B are:

$$\begin{aligned} h(A) &\leq c(A,B) + h(B) & h(B) &\leq c(B,A) + h(A) \\ h(C) &\leq c(C,B) + h(B) & h(B) &\leq c(B,C) + h(C) \\ h(D) &\leq c(D,B) + h(B) & h(B) &\leq c(B,D) + h(D) \end{aligned}$$

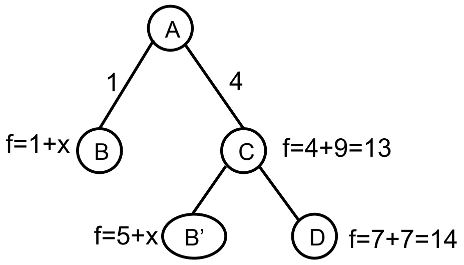
Filling in the numbers shows this results in the condition: $9 \le h_3(B) \le 10$

(iii) [2 pts] **What values of $h_3(B)$ will cause A* graph search to expand node A, then node C, then node B, then node D in order?**

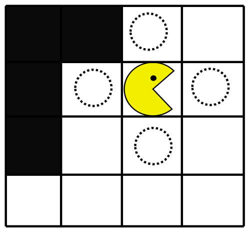
The A* search tree using heuristic h_3 is on the right. In order to make A* graph search expand node A, then node C, then node B, suppose $h_3(B) = x$, we need

$$1 + x > 13$$
$$5 + x < 14 \quad (\textit{expand } B') \quad \textit{or} \quad 1 + x < 14 \quad (\textit{expand } B)$$

so we can get $12 < h_3(B) < 13$



Q2. [9 pts] InvisiPac



Pacman finds himself to have an invisible "friend", InvisiPac. Whenever InvisiPac visits a square with a food pellet, InvisiPac will eat that food pellet—giving away its location at that time. Suppose the maze's size is MxN and there are F food pellets at the beginning.

Pacman and InvisiPac alternate moves. Pacman can move to any adjacent square (including the one where InvisiPac is) that are not walls, just as in the regular game. After Pacman moves, InvisiPac can teleport into any of the four squares that are adjacent to Pacman, as marked with the dashed circle in the graph. InvisiPac can occupy wall squares.

(a) For this subquestion, whenever InvisiPac moves, it chooses randomly from the squares adjacent to Pacman. The dots eaten by InvisiPac don't count as Pacman's score. Pacman's task is to eat as many food pellets as possible.

(i) [1 pt] Which of the following is best suited to model this problem from Pacman's perspective?

state space search **CSP** **minimax game** [✓ **MDP**] **RL**

InvisiPac moves to each adjacent square randomly with probably $\frac{1}{4}$. From pacman's point of view, it is a MDP problem with the transition function reflecting this uncertainty.

(ii) [2 pts] What is the size of a minimal state space for this problem? Give your answer as a product of factors that reference problem quantities such as M, N, F, etc. as appropriate. Below each factor, state the information it encodes. For example, you might write $4 \times MN$ and write *number of directions* underneath the first term and *Pacman's position* under the second.

$2^F \times MN$ (boolean vector for whether each food has been eaten, pacman's position)

(b) For this subquestion, whenever InvisiPac moves, it always moves into the same square relative to Pacman. For example, if InvisiPac starts one square North of Pacman, InvisiPac will always move into the square North of Pacman. Pacman knows that InvisiPac is stuck this way, but doesn't know which of the four relative locations he is stuck in. As before, if InvisiPac ends up being in a square with a food pellet, it will eat it and Pacman will thereby find out InvisiPac's location. Pacman's task is to find a strategy that minimizes the worst-case number of moves it could take before Pacman knows InvisiPac's location.

(i) [1 pt] Which of the following is best suited to model this problem from Pacman's perspective?

[✓ **state space search**] **CSP** **minimax game** **MDP** **RL**

The invisiPac will be stuck in one of the four squares relative to Pacman. It is a search problem and state space include the boolean vector for which each of the four locations invisiPac might be. The goal is to reach a state only one possible location the invisiPac can be.

(ii) [2 pts] What is the size of a minimal state space for this problem? Give your answer as a product of factors that reference problem quantities such as M, N, F, etc. as appropriate. Below each factor, state the information it encodes. For example, you might write $4 \times MN$ and write *number of directions* underneath the first term and *Pacman's position* under the second.

$2^F \times MN \times 2^4$ (boolean vector for whether each food has been eaten, pacman's position, boolean vector for which each of the four locations invisiPac might be)

(c) For this subquestion, whenever InvisiPac moves, it can choose freely between any of the four squares adjacent to Pacman. InvisiPac tries to eat as many food pellets as possible. Pacman's task is to eat as many food pellets as possible.

(i) [1 pt] Which of the following is best suited to model this problem from Pacman's perspective?

state space search **CSP** [✓ **minimax game**] **MDP** **RL**

InvisiPac tries to eat as many food pellets as possible, thus plays adversially. It is a minimax game problem.

(ii) [2 pts] What is the size of a minimal state space for this problem? Give your answer as a product of factors that reference problem quantities such as M, N, F, etc. as appropriate. Below each factor, state the information it encodes. For example, you might write $4 \times MN$ and write *number of directions* underneath the first term and *Pacman's position* under the second.
$2^F \times MN$ (boolean vector for whether each food has been eaten, pacman's position)

Q3. [27 pts] CSPs

(a) Pacman's new house

After years of struggling through mazes, Pacman has finally made peace with the ghosts, Blinky, Pinky, Inky, and Clyde, and invited them to live with him and Ms. Pacman. The move has forced Pacman to change the rooming assignments in his house, which has 6 rooms. He has decided to figure out the new assignments with a CSP in which the variables are Pacman **(P)**, Ms. Pacman **(M)**, Blinky **(B)**, Pinky **(K)**, Inky **(I)**, and Clyde **(C)**, the values are which room they will stay in, from 1-6, and the constraints are:

i) No two agents can stay in the same room	
ii) $\mathbf{P} > 3$	vi) **B** is even
iii) **K** is less than **P**	vii) **I** is not 1 or 6
iv) **M** is either 5 or 6	viii) $|\mathbf{I-C}| = 1$
v) $\mathbf{P} > \mathbf{M}$	ix) $|\mathbf{P-B}| = 2$

(i) [1 pt] **Unary constraints** On the grid below cross out the values from each domain that are eliminated by enforcing unary constraints.

P	~~1~~	~~2~~	~~3~~	4	5	6
B	~~1~~	2	~~3~~	4	~~5~~	6
C	1	2	3	4	5	6
K	1	2	3	4	5	6
I	~~1~~	2	3	4	5	~~6~~
M	~~1~~	~~2~~	~~3~~	~~4~~	5	6

The unary constraints are ii, iv, vi, and vii. ii crosses out 1,2, and 3 for P. iv crosses out 1,2,3,4 for M. vi crosses out 1,3, and 5 for B. vii crosses out 1 and 6 for I. K and C have no unary constraints, so their domains remain the same.

(ii) [1 pt] **MRV** According to the Minimum Remaining Value (MRV) heuristic, which variable should be assigned to first?

( ) P ( ) B ( ) C ( ) K ( ) I [✓ M]

M has the fewest value remaining in its domain (2), so it should be selected first for assignment.

(iii) [2 pts] **Forward Checking** For the purposes of decoupling this problem from your solution to the previous problem, assume we choose to assign P first, and assign it the value 6. What are the resulting domains after enforcing unary constraints (from part i) and running forward checking for this assignment?

P						6
B	~~1~~	~~2~~	~~3~~	4	~~5~~	~~6~~
C	1	2	3	4	5	~~6~~
K	1	2	3	4	5	~~6~~
I	~~1~~	2	3	4	5	~~6~~
M	~~1~~	~~2~~	~~3~~	~~4~~	5	~~6~~

In addition to enforcing the unary constraints from part i, the domains are further constrained by all constraints involving P. This includes constraints i, iii, v, and ix. i removes 6 from the domains of all variables. iii removes 6 from the domain of K (already removed by constraint i). v removes 6 from the domain of M (also already removed by i). ix removes 2 and 6 from the domain of B.

(iv) [3 pts] **Iterative Improvement** Instead of running backtracking search, you decide to start over and run iterative improvement with the min-conflicts heuristic for value selection. Starting with the following assignment:

P:6, B:4, C:3, K:2, I:1, M:5

First, for each variable write down how many constraints it violates in the table below.
Then, in the table on the right, for all variables that could be selected for assignment, put an x in any box

that corresponds to a possible value that could be assigned to that variable according to min-conflicts. When marking next values a variable could take on, only mark values different from the current one.

Variable	# violated
P	0
B	0
C	1
K	0
I	2
M	0

	1	2	3	4	5	6
P						
B						
C		x				
K						
I		x		x		
M						

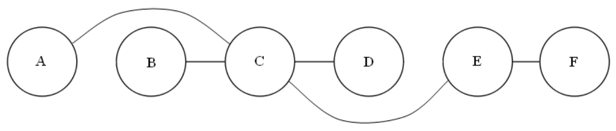
Both I and C violate constraint viii, because |I-C|=2. I also violates constraint vii. No other variables violate any constraints. According to iterative improvement, any conflicted variable could be selected for assignment, in this case I and C. According to min-conflicts, the values that those variables can take on are the values that minimize the number of constraints violated by the variable. Assigning 2 or 4 to I causes it to violate constraint i, because other variables already have the values 2 and 4. Assigning 2 to C also only causes C to violate 1 constraint.

(b) Variable ordering

We say that a variable X is backtracked if, after a value has been assigned to X, the recursion returns at X without a solution, and a different value must be assigned to X.
For this problem, consider the following three algorithms:

1. Run backtracking search with no filtering
2. Initially enforce arc consistency, then run backtracking search with no filtering
3. Initially enforce arc consistency, then run backtracking search while enforcing arc consistency after each assignment

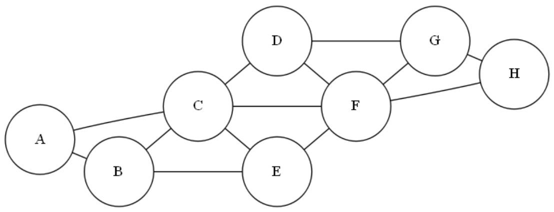
(i) [5 pts]
For each algorithm, circle all orderings of variable assignments that guarantee that no backtracking will be necessary when finding a solution to the CSP represented by the following constraint graph.



Algorithm 1	Algorithm 2	Algorithm 3
A-B-C-D-E-F	A-B-C-D-E-F	[✓ A-B-C-D-E-F]
F-E-D-C-B-A	F-E-D-C-B-A	[✓ F-E-D-C-B-A]
C-A-B-D-E-F	[✓ C-A-B-D-E-F]	[✓ C-A-B-D-E-F]
B-D-A-F-E-C	B-D-A-F-E-C	[✓ B-D-A-F-E-C]
D-E-F-C-B-A	D-E-F-C-B-A	[✓ D-E-F-C-B-A]
B-C-D-A-E-F	[✓ B-C-D-A-E-F]	[✓ B-C-D-A-E-F]

Algorithm 1:
No filtering means that there are no guarantees that an assignment to one variable has consistent assignments in any other variable, so backtracking may be necessary.
Algorithm 2:
This algorithm is very similar to the tree-structured CSP algorithm presented in class, in which arcs are enforced from one right to left, and then variables are assigned from left to right. The arcs enforced in that algorithm are a subset of all arcs enforced when enforcing arc consistency. Thus, any linear ordering of variables in which each variable is assigned before all of its children in the tree will guarantee no backtracking.
Algorithm 3:
Any first assignment can be the root of a tree, which, from class, we know is consistent and will not require backtracking. This assignment can then be viewed as conditioning the graph on that variable, and after re-running arc consistency, it can be removed from the graph. This results in either one or two tree-structured graphs that are also arc consistent, and the process can be repeated.

(ii) [5 pts]
For each algorithm, circle all orderings of variable assignments that guarantee that no more than two variables will be backtracked when finding a solution to the CSP represented by the following constraint graph.



Algorithm 1	Algorithm 2	Algorithm 3
C-F-A-B-E-D-G-H	C-F-A-B-E-D-G-H	$\boxed{\text{C-F-A-B-E-D-G-H}}$
F-C-A-H-E-B-D-G	F-C-A-H-E-B-D-G	$\boxed{\text{F-C-A-H-E-B-D-G}}$
A-B-C-E-D-F-G-H	A-B-C-E-D-F-G-H	A-B-C-E-D-F-G-H
G-C-H-F-B-D-E-A	G-C-H-F-B-D-E-A	$\boxed{\text{G-C-H-F-B-D-E-A}}$
A-B-E-D-G-H-C-F	A-B-E-D-G-H-C-F	A-B-E-D-G-H-C-F
A-D-B-G-E-H-C-F	A-D-B-G-E-H-C-F	A-D-B-G-E-H-C-F

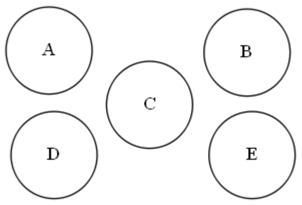
Algorithm 1:
This might backtrack for the same reason as algorithm 1 for the previous problem.
Algorithm 2:
If the first two assignments are not a cutset (C-F, C-G, or F-B), the graph will still contain cycles, for which there is no guarantee that backtracking will not be necessary. If the first two assignments are a cutset, the remaining graph will be a tree. However, because arc consistency was not enforced after the assignment, there is no guarantee against further backtracking. To see this, consider the sub-graph A,B,C,E, with domains $\{1,2,3\}$, and constraints $A=C$, $B \geq A$, $E=C+2$, $B \geq C$, $E=B$. If this is assigned in the order C-A-B-E, then by assigning 1 to C and A, assigning either 1 or 2 to B would result in an empty domain for E and cause B to backtrack.
Algorithm 3:
After assigning the cutset, the remaining graph is a tree, which guarantees no further backtracking with algorithm 3 as seen in the previous problem.

(c) All Satisfying Assignments Now consider a modified CSP in which we wish to find every possible satisfying assignment, rather than just one such assignment as in normal CSPs. In order to solve this new problem, consider a new algorithm which is the same as the normal backtracking search algorithm, except that when it sees a solution, instead of returning it, the solution gets added to a list, and the algorithm backtracks. Once there are no variables remaining to backtrack on, the algorithm returns the list of solutions it has found.

For each graph below, select whether or not using the MRV and/or LCV heuristics could affect the number of nodes expanded in the search tree in this new situation.

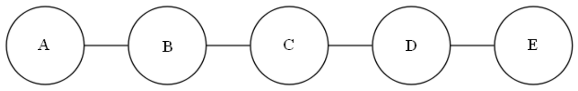
The remaining parts all have a similar reasoning. Since every value has to be checked regardless of the outcome of previous assignments, the order in which the values are checked does not matter, so LCV has no effect.
In the general case, in which there are constraints between variables, the size of each domain can vary based on the order in which variables are assigned, so MRV can still have an effect on the number of nodes expanded for the new "find all solutions" task.
The one time that MRV is guaranteed to not have any effect is when the constraint graph is completely disconnected, as is the case for part i. In this case, the domains of each variable do not depend on any other variable's assignment. Thus, the ordering of variables does not matter, and MRV cannot have any effect on the number of nodes expanded.

(i) [2 pts]



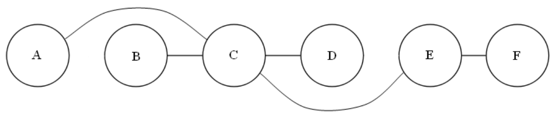
[✓ Neither MRV nor LCV can have an effect.]
Only MRV can have an effect.
Only LCV can have an effect .
Both MRV and LCV can have an effect.

(ii) [2 pts]



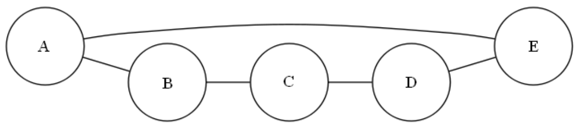
Neither MRV nor LCV can have an effect.
[✓ Only MRV can have an effect.]
Only LCV can have an effect .
Both MRV and LCV can have an effect.

(iii) [2 pts]



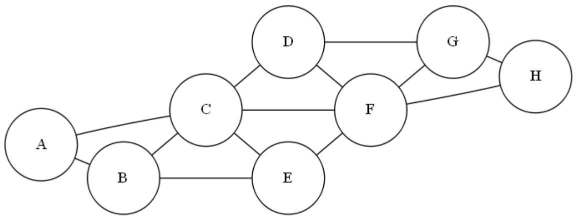
Neither MRV nor LCV can have an effect.
[✓ Only MRV can have an effect.]
Only LCV can have an effect .
Both MRV and LCV can have an effect.

(iv) [2 pts]



Neither MRV nor LCV can have an effect.
[✓ Only MRV can have an effect.]
Only LCV can have an effect .
Both MRV and LCV can have an effect.

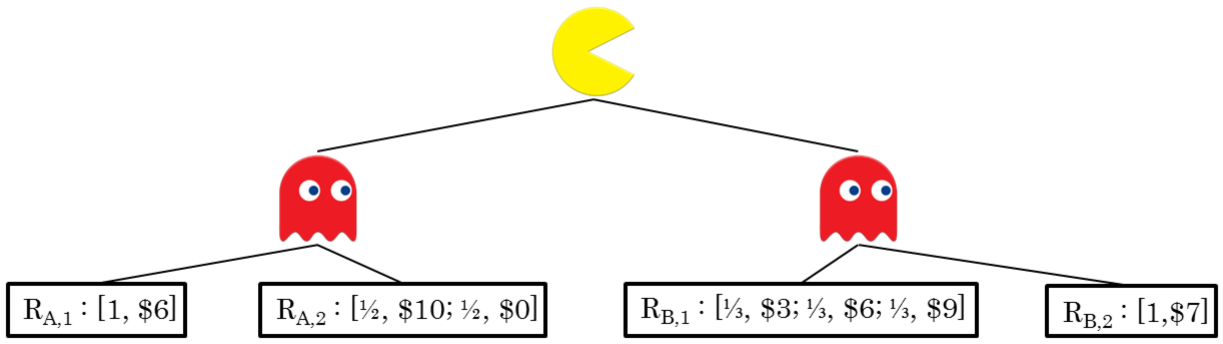
(v) [2 pts]



- ○ Neither MRV nor LCV can have an effect.
- [✓ Only MRV can have an effect.]
- ○ Only LCV can have an effect .
- ○ Both MRV and LCV can have an effect.

Q4. [10 pts] Utilities

Pacman is buying a raffle ticket from the ghost raffle ticket vendor. There are two ticket types: A and B, but there are multiple specific tickets of each type. Pacman picks a ticket type, but the ghost will then choose which specific ticket Pacman will receive. Pacman's utility for a given raffle ticket is equal to the utility of the lottery of outcomes for that raffle ticket. For example, ticket $R_{A,2}$ corresponds to a lottery with equal chances of yielding 10 and 0, and so $U(R_{A,2}) = U([\frac{1}{2}, 10; \frac{1}{2}, 0])$. Pacman, being a rational agent, wants to maximize his expected utility, but the ghost may have other goals! The outcomes are illustrated below.



$R_{A,1}$: [1, \$6]
$R_{A,2}$: [½, \$10; ½, \$0]
$R_{B,1}$: [⅓, \$3; ⅓, \$6; ⅓, \$9]
$R_{B,2}$: [1,\$7]

(a) Imagine that Pacman's utility for money is $U(m) = m$.

(i) [2 pts] What are the utilities to Pacman of each raffle ticket?

$U(R_{A,1}) = 6$ $\qquad$ $U(R_{A,2}) = 5$

$U(R_{B,1}) = 6$ $\qquad$ $U(R_{B,2}) = 7$

$U(R_{B,1}) = \frac{1}{3} \times U(3) + \frac{1}{3} \times U(6) + \frac{1}{3} \times U(9) = \frac{1}{3} \times 3 + \frac{1}{3} \times 6 + \frac{1}{3} \times 9 = 6$

(ii) [1 pt] Which raffle ticket will Pacman receive under optimal play if the ghost is trying to minimize Pacman's utility (and Pacman knows the ghost is doing so)? (circle one)

$R_{A,1}$ $\qquad$ $R_{A,2}$ $\qquad$ [✓ $R_{B,1}$] $\qquad$ $R_{B,2}$

(iii) [1 pt] What is the equivalent monetary value of raffle ticket $R_{B,1}$?

$U(m) = m = U(R_{B,1}) = 6$ $\qquad$ $m = 6$

(b) Now imagine that Pacman's utility for money is given by $U(m) = m^2$.

(i) [2 pts] What are the utilities to Pacman of each raffle ticket?

$U(R_{A,1}) = 36$ $\qquad$ $U(R_{A,2}) = 50$

$U(R_{B,1}) = 42$ $\qquad$ $U(R_{B,2}) = 49$

$U(R_{B,1}) = \frac{1}{3} \times U(3) + \frac{1}{3} \times U(6) + \frac{1}{3} \times U(9) = \frac{1}{3} \times 9 + \frac{1}{3} \times 36 + \frac{1}{3} \times 81 = 42$

(ii) [1 pt] The ghost is still trying to minimize Pacman's utility, but the ghost mistakenly thinks that Pacman's utility is given my $U(m) = m$, and Pacman is aware of this flaw in the ghost's model. Which raffle ticket will Pacman receive? (circle one)

$R_{A,1}$ $\qquad$ [✓ $R_{A,2}$] $\qquad$ $R_{B,1}$ $\qquad$ $R_{B,2}$

(iii) [1 pt] What is the equivalent monetary value of raffle ticket $R_{B,1}$? (you may leave your answer as an expression) $U(m) = m^2 = U(R_{B,1}) = 42$ $\qquad$ $m = \sqrt{42}$

(c) [2 pts] Pacman has the raffle with distribution [0.5, \$100; 0.5, \$0]. A ghost insurance dealer offers Pacman an insurance policy where Pacman will get \$100 regardless of what the outcome of the ticket is. If Pacman's utility for money is $U(m) = \sqrt{m}$, what is the maximum amount of money Pacman would pay for this insurance?

Call c the cost of the insurance.

Pacman's utility with insurance: $U(\$100 - c) = \sqrt{100 - c}$

Pacman's utility without insurance: $U([0.5, \$100; 0.5, \$0]) = 0.5 \times \sqrt{100} + 0.5 \times 0 = 5$

The maximum cost c Pacman would be willing to pay for the insurance is the cost c such that the two utilities are equal.

$\sqrt{100 - c} = 5$ $\qquad$ $c = 75$

Q5. [9 pts] Games: Three-Player Cookie Pruning

Three of your TAs, Alvin, Sergey, and James rent a cookie shuffler, which takes in a set number of cookies and groups them into 3 batches, one for each player. The cookie shuffler has three levers (with positions either UP or DOWN), which act to control how the cookies are distributed among the three players. Assume that 30 cookies are initially put into the shuffler.

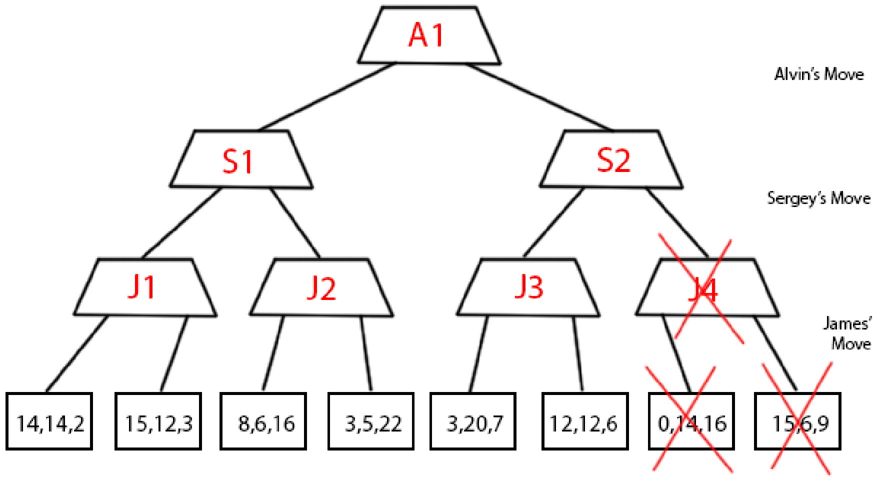
Each player controls one lever, and they act in turn. Alvin goes first, followed by Sergey, and finally James. Assume that all players are able to calculate the payoffs for every player at the terminal nodes. Assume the payoffs at the leaves correspond to the number of cookies for each player in their corresponding turn order. Hence, an utility of (7,10,13) corresponds to Alvin getting 7 cookies, Sergey getting 10 cookies, and James getting 13 cookies. No cookies are lost in the process, so the sum of cookies of all three players must equal the number of cookies put into the shuffler. Players want to maximize their own number of cookies.

(a) [3 pts] **What is the utility triple propagated up to the root?** 15,12,3

(b) [6 pts] **Is pruning possible in this game? Fill in "Yes" or "No". If yes, cross out all nodes (both leaves and intermediate nodes) that get pruned. If no, explain in one sentence why pruning is not possible.** Assume the tree traversal goes from left to right.

[✓ Yes.]

No, Reasoning:



Left lowest subtree: James chooses the node (15,12,3) at node J1. This gets propagated up to S1. Sergey doesn't know what value he can get on his right child, so we explore that. Upon propagating (8,6,16) to J2, we must explore J2's right child since there could be a triple better for James' best option (greater than 16) and better than Sergey's best option. (greater than 12). (15,12,3) gets propagated up to A1.

Alvin might have a good option (greater than 15) in the right subtree, so we explore down. On the (3,20,7) node, we propagate this up to J3. We need to continue going down this path because Sergey doesn't know if he can get more than 20, and Alvin doesn't know if he can get more than 15 (A1's value). Hence, (12,12,6) is explored. (3,20,7) is propagated to S2. Now, we can guarantee that Sergey will prefer any cookie count over 20. But, because the sum of cookies must be 30, this means that Alvin can get no more than 10 cookies in the right subtree. Hence, we can immediately prune any children of S2.

Q6. [10 pts] The nature of discounting

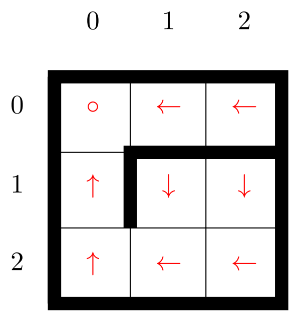
Pacman in stuck in a friendlier maze where he gets a reward every time he visits state (0,0). This setup is a bit different from the one you've seen before: Pacman can get the reward multiple times; these rewards do not get ”used up” like food pellets and there are no “living rewards”. As usual, Pacman can not move through walls and may take any of the following actions: go North (↑), South (↓), East (→), West (←), or stay in place (∘). State (0,0) gives a total reward of 1 every time Pacman takes an action in that state regardless of the outcome, and all other states give no reward.

The first sentence in the paragraph above was confusing at exam time. The precise reward fucntion is: $R_{(0,0),a} = 1$ for any action a and $R_{s',a} = 0$ for all $s' \neq (0,0)$

You should not need to use any other complicated algorithm/calculations to answer the questions below. We remind you that geometric series converge as follows: $1 + \gamma + \gamma^2 + \cdots = 1/(1-\gamma)$.

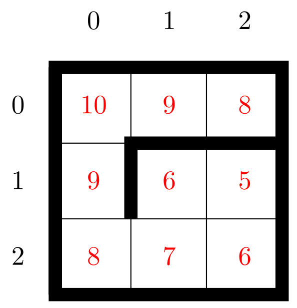
(a) [2 pts] Assume finite horizon of $h = 10$ (so Pacman takes exactly 10 steps) and no discounting ($\gamma = 1$).

Fill in an optimal policy:



(available actions: ↑, ↓, →, ←, ∘)

Fill in the value function:



(b) The following Q-values correspond to the value function you specified above.

$Q_{s,a}^{10 \text{ steps to go}} = R_s + V_{s'}^{9 \text{ steps to go}}$ where s' is the successor of state s after taking actions a

(i) [1 pt] The Q value of state-action $(0, 0)$, (East) is: 9

(ii) [1 pt] The Q value of state-action $(1, 1)$, (East) is: 4

(c) Assume finite horizon of $h = 10$, no discounting, but the action to stay in place is temporarily (for this sub-point only) unavailable. Actions that would make Pacman hit a wall are not available. Specifically, Pacman can not use actions North or West to remain in state $(0, 0)$ once he is there.

(i) [1 pt] [*true* or [✓ *false*]] There is just one optimal action at state $(0, 0)$

East and South are both optimal actions

(ii) [1 pt] The value of state $(0, 0)$ is: 5

Since the “stay action” is no longer available, Pacman needs to exit state $(0, 0)$ at even time steps

(d) [2 pts] Assume infinite horizon, discount factor $\gamma = 0.9$.

The value of state $(0, 0)$ is: $1/(1-\gamma) = 10$

(e) [2 pts] Assume infinite horizon and no discount ($\gamma = 1$). At every time step, after Pacman takes an action and collects his reward, a power outage could suddenly end the game with probability $\alpha = 0.1$.

The value of state $(0, 0)$ is: $\underline{\quad 1/\alpha = 10 \quad}$

Q7. [10 pts] The Value of Games

Pacman is the model of rationality and seeks to maximize his expected utility,
but that doesn't mean he never plays games.

(a) [3 pts] **Q-Learning to Play under a Conspiracy.** Pacman does tabular Q-learning (where every state-action pair has its own Q-value) to figure out how to play a game against the adversarial ghosts. As he likes to explore, Pacman always plays a random action. After enough time has passed, every state-action pair is visited infinitely often. The learning rate decreases as needed. For any game state s, the value $\max_a Q(s,a)$ for the learned $Q(s,a)$ is equal to (for complete search trees)

- [✓ The minimax value where Pacman maximizes and ghosts minimize.]
- ◯ The expectimax value where Pacman maximizes and ghosts act uniformly at random.
- ◯ The expectimax value where Pacman plays uniformly at random and ghosts minimize.
- ◯ The expectimax value where both Pacman and ghosts play uniformly at random.
- ◯ None of the above.

Only minimax search correctly models the adversarial game of Pacman's learned policy: although the acting policy is random, the learned policy is the optimal policy for max.

Tabular Q-learning and full-depth minimax search both compute the exact value of all states, since Q-learning has a value for every state-action (and thus every state) and the conditions are right for convergence.

(b) [3 pts] **Feature-based Q-Learning the Game under a Conspiracy.** Pacman now runs feature-based Q-learning. The Q-values are equal to the evaluation function $\sum_{i=1}^{n} w_i f_i(s,a)$ for weights w and features f. The number of features is much less than the number of states. As he likes to explore, Pacman always plays a random action. After enough time has passed, every state-action pair is visited infinitely often. The learning rate decreases as needed. The value $\max_a Q(s,a)$ for the learned $Q(s,a)$ is equal to (for complete search trees)

- ◯ The minimax value where Pacman maximizes and ghosts minimize and the same evaluation function is used at the leaves.
- ◯ The expectimax value where Pacman maximizes and ghosts act uniformly at random and the same evaluation function is used at the leaves.
- ◯ The expectimax value where Pacman plays uniformly at random and ghosts minimize and the same evaluation function is used at the leaves.
- ◯ The expectimax value where both Pacman and ghosts play uniformly at random and the same evaluation function is used at the leaves.
- [✓ None of the above.]

Full-depth minimax search computes the approximate value of all the leaves by the evaluation function and then exactly propagates these values up the search tree. Feature-based Q-learning approximates the value of all states with the evaluation function and not only the leaves. Since there are fewer features than states, the approximation is not expressive enough to capture the true values of all the states.

(c) [2 pts] **A Costly Game.** Pacman is now stuck playing a new game with only costs and no payoff. Instead of maximizing expected utility $V(s)$, he has to minimize expected costs $J(s)$. In place of a reward function, there is a cost function $C(s,a,s')$ for transitions from s to s' by action a. We denote the discount factor by $\gamma \in (0,1)$. $J^*(s)$ is the expected cost incurred by the optimal policy. Which one of the following equations is satisfied by J^*?

- ◯ $J^*(s) = \min_a \sum_{s'} [C(s,a,s') + \gamma \max_{a'} T(s,a',s') * J^*(s')]$
- ◯ $J^*(s) = \min_{s'} \sum_a T(s,a,s')[C(s,a,s') + \gamma * J^*(s')]$
- ◯ $J^*(s) = \min_a \sum_{s'} T(s,a,s')[C(s,a,s') + \gamma * \max_{s'} J^*(s')]$
- ◯ $J^*(s) = \min_{s'} \sum_a T(s,a,s')[C(s,a,s') + \gamma * \max_{s'} J^*(s')]$
- [✓ $J^*(s) = \min_a \sum_{s'} T(s,a,s')[C(s,a,s') + \gamma * J^*(s')]$]
- ◯ $J^*(s) = \min_{s'} \sum_a [C(s,a,s') + \gamma * J^*(s')]$

Minimum expected cost has the same form as maximum expected utility except that the optimization is in the opposite direction and costs replace rewards.

(d) [2 pts] **It's a conspiracy again!** The ghosts have rigged the costly game so that once Pacman takes an action they can pick the outcome from all states $s' \in S'(s,a)$, the set of all s' with non-zero probability according to $T(s,a,s')$. Choose the correct Bellman-style equation for adversarial ghosts.

○ $J^*(s) = \min_a \max_{s'} T(s,a,s')[C(s,a,s') + \gamma * J^*(s')]$

○ $J^*(s) = \min_{s'} \sum_a T(s,a,s')[\max_{s'} C(s,a,s') + \gamma * J^*(s')]$

○ $J^*(s) = \min_a \min_{s'} [C(s,a,s') + \gamma * \max_{s'} J^*(s')]$

[✓ $J^*(s) = \min_a \max_{s'} [C(s,a,s') + \gamma * J^*(s')]$]

○ $J^*(s) = \min_{s'} \sum_a T(s,a,s')[\max_{s'} C(s,a,s') + \gamma * \max_{s'} J^*(s')]$

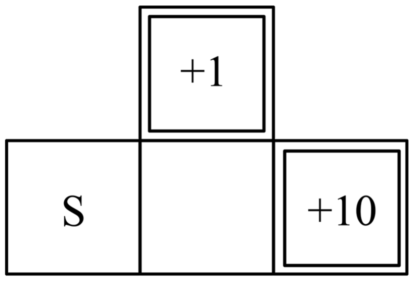
○ $J^*(s) = \min_a \min_{s'} T(s,a,s')[C(s,a,s') + \gamma * J^*(s')]$

Pacman is still minimizing cost, but instead of expected cost it is worst-case (maximum) cost among all possible successors s'. The transition probability $T(s,a,s')$ is dropped since the worst-case outcome is selected with certainty.

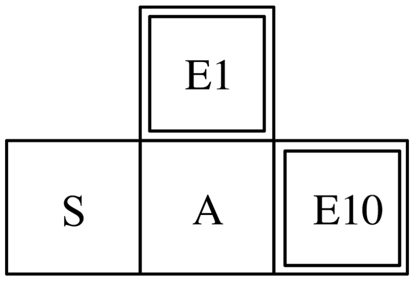
Q8. [16 pts] Infinite Time to Study

Pacman lives in a calm gridworld. S is the start state and double-squares are exit states. In exits, the only action available is exit, which earns the associated reward and transitions to a terminal state X (not shown). In normal states, the actions are to move to neighboring squares (for example, S has the single action $\rightarrow$) and they always succeed. There is no living reward, so all non-exit actions have reward 0.

Throughout the problem the discount $\gamma = 1$.



The calmworld



State names

The Q-learning update equation is $Q'(s,a) = (1-\alpha)Q(s,a) + \alpha[R(s,a,s') + \max_{a'} Q(s',a')]$. However, this problem can be solved without manually computing any Q-value updates.

(a) [2 pts] What are the optimal values of S and A?

$V^*(S) =$ 10

$V^*(A) =$ 10

In a deterministic undiscounted ($\gamma = 1$) MDP the optimal value is the maximum return from the state.

Pacman doesn't know the details of this gridworld so he does Q-learning with a learning rate of 0.5 and all Q-values initialized to 0 to figure it out.

Consider the following sequence of transitions in the calmworld:

s	a	s'	r
S	$\rightarrow$	A	0
A	$\uparrow$	E1	0
E1	exit	X	1
S	$\rightarrow$	A	0
A	$\rightarrow$	E10	0
E10	exit	X	10

(b) [2 pts] Circle the Q-values that are non-zero after these episodes.

$Q(S,\rightarrow)$ $\quad Q(A,\uparrow)$ $\quad Q(A,\rightarrow)$ $\quad \boxed{Q(E1, exit)}$ $\quad \boxed{Q(E10, exit)}$

Q-values are only updated when a transition is experienced. $Q(E1, exit), Q(E10, exit)$ are updated to the reward earned, but the other states were updated when all the Qs were still zero.

(c) [2 pts] What do the Q-values converge to if these episodes are repeated infinitely with a constant learning rate of 0.5? Write *none* if they do not converge.

The MDP is undiscounted and deterministic, so Q-learning converges even though the learning rate is constant. With infinite visits the Q-values will converge to the true values.

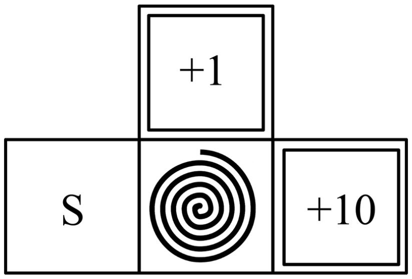
$Q(S, \rightarrow) =$ 10. $Q(S, \rightarrow)$ is the only state-action for S, so it converges to the optimal value $V^*(S)$.

$Q(A, \leftarrow) =$ 0. The episode $A, \leftarrow$ is never experienced so it is unchanged after initialization.

$Q(A, \uparrow) =$ 1. The only return possible after $A, \uparrow$ is 1.

(Q-learning details reminder: assume $\alpha = 0.5$ and the Q-values are initialized to 0.)

It's vortex season in the gridworld. In the vortex state the only action is escape, which delivers Pacman to a neighboring state uniformly at random.



The vortexworld

(d) [2 pts] What are the optimal values of S and A in the vortex gridworld?

The optimal value is the mean of the end returns 1 and 10 because the exit states have equal probability. The value of S is the same as A since the discount $\gamma = 1$ and the transition $S, \rightarrow, A$ is deterministic. The transition $A, escape, S$ has no impact on the value because the MDP is undiscounted / infinite horizon.

$V^*(S) =$ 5.5

$V^*(A) =$ 5.5

Consider the following sequences of transitions in the vortexworld:

S1

s	a	s'	r
S	$\rightarrow$	A	0
A	escape	E1	0
E1	exit	X	1
S	$\rightarrow$	A	0
A	escape	E10	0
E10	exit	X	10

S2

s	a	s'	r
S	$\rightarrow$	A	0
A	escape	E1	0
E1	exit	X	1
S	$\rightarrow$	A	0
A	escape	E10	0
E10	exit	X	10
S	$\rightarrow$	A	0
A	escape	E10	0
E10	exit	X	10

(e) [2 pts] What do the Q-values converge to if the sequence S1 is repeated infinitely with appropriately decreasing learning rate? Write *never* if they do not converge.

$Q^{S1}(S, \rightarrow) =$ 5.5 $\qquad Q^{S1}(A, escape) =$ 5.5

The conditions for convergence are satisfied and the Q-values converge to the expected return. The expectation of returns is $\frac{1}{2} \times 1 + \frac{1}{2} \times 10$.

(f) [2 pts] What if the sequence S2 is repeated instead?

$Q^{S2}(S, \rightarrow) =$ 7 $\qquad Q^{S2}(A, escape) =$ 7

The expectation of returns is $\frac{1}{3} \times 1 + \frac{2}{3} \times 10$ because two out of three exits in the sequence have reward 10.

(g) [2 pts] Which is the true optimum $Q^*(S, \rightarrow)$ in the vortex gridworld? Circle the answer.

[✓ $Q^{S1}(S, \rightarrow)$] $\qquad Q^{S2}(S, \rightarrow) \qquad$ *other*

The sequence $S1$ has the same distribution of returns as the true distribution, even though all of the possible transitions are not experienced.

(h) [2 pts] Q-learning with constant $\alpha = 1$ and visiting state-actions infinitely often converges

[✓ in calmworld] $\qquad$ in vortexworld $\qquad$ in neither world

For learning rate $\alpha = 1$ the Q-learning update sets $Q(s, a)$ to the sample $[R(s, a, s') + \max_{a'} Q(s', a')]$ with no regard for the previous value of $Q(s, a)$.

In deterministic MDPs (like calmworld), even with constant learning rate $\alpha = 1$, Q-learning converges. In fact, this learning rate is optimal for deterministic MDPs in the sense that it converges fastest.

In stochastic MDPs (like vortexworld), with constant learning rate $\alpha = 1$, the $Q(s, a)s$ are always equal to the most recent sample for the state-action (s, a). The $Q(s, a)$s will cycle among the possible samples and never converge.

THIS PAGE IS INTENTIONALLY LEFT BLANK