CS 188 Fall 2012 — Introduction to Artificial Intelligence — Midterm I

- You have approximately 3 hours.
- The exam is closed book, closed notes except a 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	

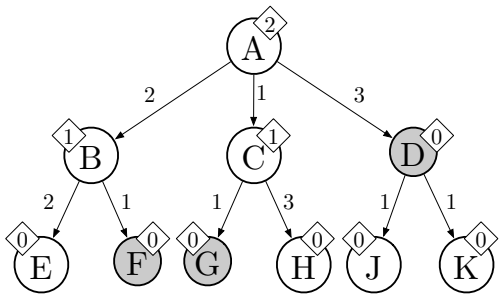
For staff use only:

Q1.	Search: Algorithms	/10
Q2.	Search: Heuristic Function Properties	/6
Q3.	Search: Slugs	/8
Q4.	Value Functions	/10
Q5.	CSPs: Apple's New Campus	/9
Q6.	CSPs: Properties	/7
Q7.	Games: Alpha-Beta Pruning	/8
Q8.	Utilities: Low/High	/18
Q9.	MDPs and RL: Mini-Grids	/24
	Total	/100

THIS PAGE IS INTENTIONALLY LEFT BLANK

Q1. [10 pts] Search: Algorithms

Consider the state space search problem shown to the right. A is the start state and the shaded states are goals. Arrows encode possible state transitions, and numbers by the arrows represent action costs. Note that state transitions are directed; for example, $A \rightarrow B$ is a valid transition, but $B \rightarrow A$ is not. Numbers shown in diamonds are heuristic values that estimate the optimal (minimal) cost from that node to a goal.



For each of the following search algorithms, write down the nodes that are removed from fringe in the course of the search, as well as the final path returned. Because the original problem graph is a tree, the tree and graph versions of these algorithms will do the same thing, and you can use either version of the algorithms to compute your answer.

Assume that the data structure implementations and successor state orderings are all such that *ties are broken alphabetically*. For example, a partial plan $S \rightarrow X \rightarrow A$ would be expanded before $S \rightarrow X \rightarrow B$; similarly, $S \rightarrow A \rightarrow Z$ would be expanded before $S \rightarrow B \rightarrow A$.

(a) [2 pts] **Depth-First Search (ignores costs)**

Nodes removed from fringe: A, B, E, F

Path returned: A, B, F

(b) [2 pts] **Breadth-First Search (ignores costs)**

Nodes removed from fringe: A, B, C, D

Path returned: A, D

(c) [2 pts] **Uniform-Cost Search**

Nodes removed from fringe: A, C, B, G

Path returned: A, C, G

(d) [2 pts] **Greedy Search**

Nodes removed from fringe: A, D

Path returned: A, D

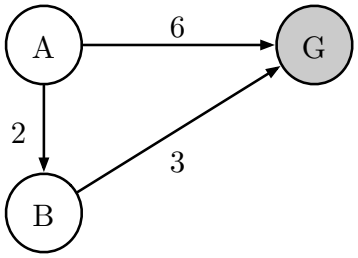
(e) [2 pts] **A* Search**

Nodes removed from fringe: A, C, G

Path returned: A, C, G

Q2. [6 pts] Search: Heuristic Function Properties

For the following questions, consider the search problem shown on the left. It has only three states, and three directed edges. A is the start node and G is the goal node. To the right, four different heuristic functions are defined, numbered I through IV.



	h(A)	h(B)	h(G)
I	4	1	0
II	5	4	0
III	4	3	0
IV	5	2	0

(a) [4 pts] **Admissibility and Consistency**

For each heuristic function, circle whether it is admissible and whether it is consistent with respect to the search problem given above.

	Admissible?		Consistent?	
I	[Yes]	No	Yes	[No]
II	Yes	[No]	Yes	[No]
III	[Yes]	No	[Yes]	No
IV	[Yes]	No	Yes	[No]

II is the only inadmissible heuristic, as it overestimates the cost from B: $h(B) = 4$, when the actual cost to G is 3.

To check whether a heuristic is consistent, ensure that for all paths, $h(N) - h(L) \leq \text{path}(N \to L)$, where N and L stand in for the actual nodes. In this problem, $h(G)$ is always 0, so making sure that the direct paths to the goal ($A \to G$ and $B \to G$) are consistent is the same as making sure that the heuristic is admissible. The path from A to B is a different story.

Heuristic I is not consistent: $h(A) - h(B) = 4 - 1 = 3 \not\leq \text{path}(A \to B) = 2$.
Heuristic III is consistent: $h(A) - h(B) = 4 - 3 = 1 \leq 2$
Heuristic IV is not consistent: $h(A) - h(B) = 5 - 2 = 3 \not\leq 2$

(b) [2 pts] **Function Domination**

Recall that *domination* has a specific meaning when talking about heuristic functions.

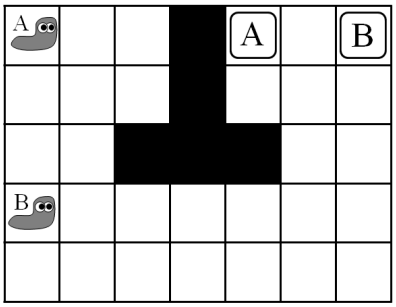
Circle all true statements among the following.

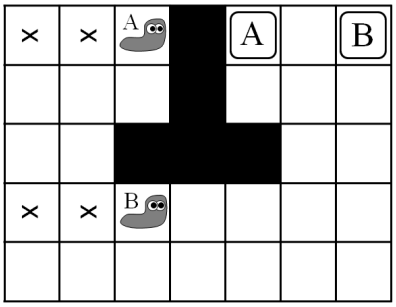
1. Heuristic function III dominates IV.
2. Heuristic function IV dominates III.
3. [Heuristic functions III and IV have no dominance relationship.]
4. Heuristic function I dominates IV.
5. [Heuristic function IV dominates I.]
6. Heuristic functions I and IV have no dominance relationship.

For one heuristic to dominate another, *all* of its values must be greater than or equal to the corresponding values of the other heuristic. Simply make sure that this is the case. If it is not, the two heuristics have no dominance relationship.

Q3. [8 pts] Search: Slugs

You are once again tasked with planning ways to get various insects out of a maze. This time, it's slugs! As shown in the diagram below to the left, two slugs A and B want to exit a maze via their own personal exits. In each time step, both slugs move, though each can choose to either stay in place or move into an adjacent free square. The slugs cannot move into a square that the other slug is moving into. In addition, the slugs leave behind a sticky, poisonous substance and so they cannot move into any square that *either* slug has ever been in. For example, if both slugs move right twice, the maze is as shown in the diagram below to right, with the x squares unpassable to either slug.





You must pose a search problem that will get them to their exits in as few time steps as possible. You may assume that the board is of size N by M; all answers should hold for a general instance, not simply the instance shown above. (You do not need to generalize beyond two slugs.)

(a) [3 pts] How many states are there in a minimal representation of the space? Justify with a brief description of the components of your state space.

$2^{MN}(MN)^2$

The state includes a bit for each of the MN squares, indicating whether the square has been visited (2^{MN} possibilities). It also includes the locations of each slug (MN possibilities for each of the two slugs).

(b) [2 pts] What is the branching factor? Justify with a brief description of the successor function.

$5 \times 5 = 25$ for the first time step, $4 \times 4 = 16$ afterwards.

At the start state each slug has at most five possible next locations (North, South, East, West, Stay). At all future time steps one of those options will certainly be blocked off by the snail's own trail left at the previous time step. Only 4 possible next locations remain.

We accepted both 25 and 16 as correct answers.

(c) [3 pts] Give a non-trivial admissible heuristic for this problem.

max(maze distance of bug A to its exit, maze distance of bug B to its exit)

Many other correct answers are possible.

Q4. [10 pts] Value Functions

Consider a general search problem defined by:

- A set of states, S.
- A start state s_0.
- A set of goal states G, with $G \subset S$.
- A successor function $Succ(s)$ that gives the set of states s' that you can go to from the current state s.
- For each successor s' of s, the cost (weight) $W(s, s')$ of that action.

As usual, the search problem is to find a lowest-cost path from the state state s_0 to a goal $g \in G$. You may assume that each non-goal state has at least one successor, that the weights are all positive, and that all states can reach a goal.

Define $C(s)$ to be the *optimal cost* of the state s; that is, the lowest-cost path from s to any goal. For $g \in G$, clearly $C(g) = 0$.

(a) [4 pts] Write a Bellman-style (one-step lookahead) equation that expresses $C(s)$ for a non-goal s in terms of the optimal costs of other states.

$$C(s) = \min_{s' \in Succ(s)} [W(s, s') + C(s')]$$

(b) [2 pts] Consider a heuristic function $h(s)$ with $h(s) \geq 0$. What relation must hold between $h(s)$ and $C(s)$ for $h(s)$ to be an admissible heuristic? (Your answer should be a mathematical expression.)

$h(s) \leq C(s), \forall s \in S$

(c) [4 pts] By analogy to value iteration, define $C_k(s)$ to be the minimum cost of any plan starting from s that is either length k or reaches a goal in at most k actions. Imagine we use C_k as a heuristic function.

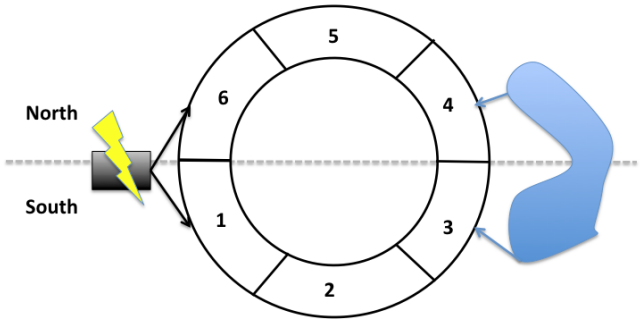
Circle all true statement(s) among the following:

1. $C_k(s)$ might be inadmissible for any given value of k.
2. $C_k(s)$ is admissible for all k. If there is a goal reachable within k actions, then $C_k(s)$ gives the exact cost to the nearest such goal. If all goals require plans of longer than k to reach, then the cheapest plan of length k underestimates the true cost.
3. $C_k(s)$ is only guaranteed to be admissible if k exceeds the length of the shortest (in steps) optimal path from a state to a goal.
4. $C_k(s)$ is only guaranteed to be admissible if k exceeds the length of the longest (in steps) optimal path from a state to a goal.
5. $C(s)$ (the optimal costs) are admissible.
6. $C_k(s)$ might be inconsistent for any given value of k.
7. $C_k(s)$ is consistent for all k. Moving from s to a successor s' decreases C_k by at most $W(s, s')$. Since the heuristic value decreases by at most the cost of the transition, the heuristic is consistent.
8. $C_k(s)$ is only guaranteed to be consistent if k exceeds the length of the shortest (in steps) optimal path from a state to a goal.
9. $C_k(s)$ is only guaranteed to be consistent if k exceeds the length of the longest (in steps) optimal path from a state to a goal.
10. $C(s)$ (the optimal costs) are consistent.

Q5. [9 pts] CSPs: Apple's New Campus

Apple's new circular campus is nearing completion. Unfortunately, the chief architect on the project was using Google Maps to store the location of each individual department, and after upgrading to iOS 6, all the plans for the new campus were lost!

The following is an approximate map of the campus:



The campus has six offices, labeled 1 through 6, and six departments:

- Legal (L)
- Maps Team (M)
- Prototyping (P)
- Engineering (E)
- Tim Cook's office (T)
- Secret Storage (S)

Offices can be *next to* one another, if they share a wall (for an instance, Offices 1-6). Offices can also be *across* from one another (specifically, Offices 1-4, 2-5, 3-6).

The Electrical Grid is connected to offices 1 and 6. The Lake is visible from offices 3 and 4. There are two "halves" of the campus – South (Offices 1-3) and North (Offices 4-6).

The constraints are as follows:

i. (L)egal wants a view of the lake to look for prior art examples.

ii. (T)im Cook's office must not be across from (M)aps.

iii. (P)rototyping must have an electrical connection.

iv. (S)ecret Storage must be next to (E)ngineering.

v. (E)ngineering must be across from (T)im Cook's office.

vi. (P)rototyping and (L)egal cannot be next to one another.

vii. (P)rototyping and (E)ngineering must be on opposite sides of the campus (if one is on the North side, the other must be on the South side).

viii. No two departments may occupy the same office.

This page is repeated as the second-to-last page of this midterm for you to rip out and use for reference as you work through the problem.

(a) [3 pts] **Constraints**. Note: There are multiple ways to model constraint *viii*. In your answers below, assume constraint *viii* is modeled as multiple pairwise constraints, not a large n-ary constraint.

(i) [1 pt] Circle your answers below. Which constraints are unary?

i *ii* iii *iv* *v* *vi* *vii* *viii*

(ii) [1 pt] In the constraint graph for this CSP, how many edges are there?

Constraint *vii* connects each pair of variables; there are $\binom{6}{2} = 15$ such pairs.

(iii) [1 pt] Write out the explicit form of constraint *iii*.

$P \in \{1, 6\}$

(b) [6 pts] **Domain Filtering.** *We strongly recommend that you use a pencil for the following problems.*

(i) [2 pts] The table below shows the variable domains after unary constraints have been enforced and the value 1 has been assigned to the variable P.

Cross out all values that are eliminated by running Forward Checking after this assignment.

L			3	4		
M	1	2	3	4	5	6
P	1					
E	1	2	3	4	5	6
T	1	2	3	4	5	6
S	1	2	3	4	5	6

(ii) [4 pts] The table below shows the variable domains after unary constraints have been enforced, the value 1 has been assigned to the variable P, and now the value 3 has been assigned to variable T.

Cross out all values that are eliminated if arc consistency is enforced after this assignment. (Note that enforcing arc consistency will subsume all previous pruning.)

L			3	4		
M	1	2	3	4	5	6
P	1					
E	1	2	3	4	5	6
T			3			
S	1	2	3	4	5	6

Q6. [7 pts] CSPs: Properties

(a) [1 pt] When enforcing arc consistency in a CSP, the set of values which remain when the algorithm terminates does not depend on the order in which arcs are processed from the queue.

True **False**

(b) [1 pt] In a general CSP with n variables, each taking d possible values, what is the maximum number of times a backtracking search algorithm might have to backtrack (i.e. the number of the times it generates an assignment, partial or complete, that violates the constraints) before finding a solution or concluding that none exists? (circle one)

0 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ $O(d^n)$ ∞

In general, the search might have to examine all possible assignments.

(c) [1 pt] What is the maximum number of times a backtracking search algorithm might have to backtrack in a general CSP, if it is running arc consistency and applying the MRV and LCV heuristics? (circle one)

0 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ $O(d^n)$ ∞

The MRV and LCV heuristics are often helpful to guide the search, but are not guaranteed to reduce backtracking in the worst case.

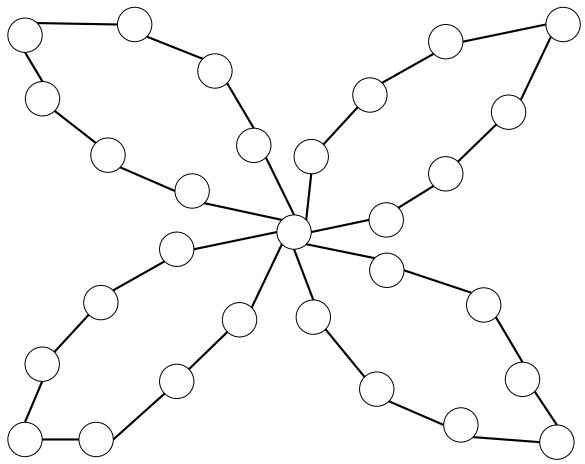
In fact, CSP solving is NP-complete, so any polynomial-time method for solving general CSPs would consititute a proof of $P = NP$ (worth a million dollars from the Clay Mathematics Institute!).

(d) [1 pt] What is the maximum number of times a backtracking search algorithm might have to backtrack in a *tree-structured* CSP, if it is running arc consistency and using an optimal variable ordering? (circle one)

0 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ $O(d^n)$ ∞

Applying arc consistency to a tree-structured CSP guarantees that no backtracking is required, if variables are assigned starting at the root and moving down towards the leaves.

(e) [3 pts] **Constraint Graph** Consider the following constraint graph:

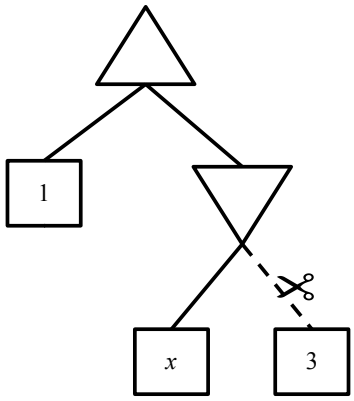


In two sentences or less, describe a strategy for efficiently solving a CSP with this constraint structure.

Loop over assignments to the variable in the middle of the constraint graph. Treating this node as a cutset, the graph becomes four independent tree-structured CSPs, each of which can be solved efficiently.

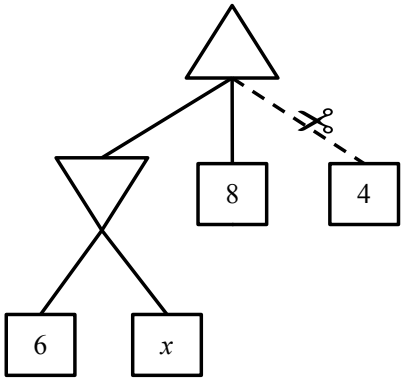
Q7. [8 pts] Games: Alpha-Beta Pruning

For each of the game-trees shown below, state for which values of x the dashed branch with the scissors will be pruned. If the pruning will not happen for any value of x write "none". If pruning will happen for all values of x write "all".

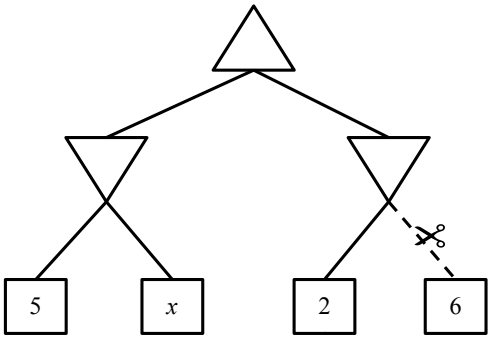


(a) Example Tree. Answer: $\mathbf{x \leq 1}$.

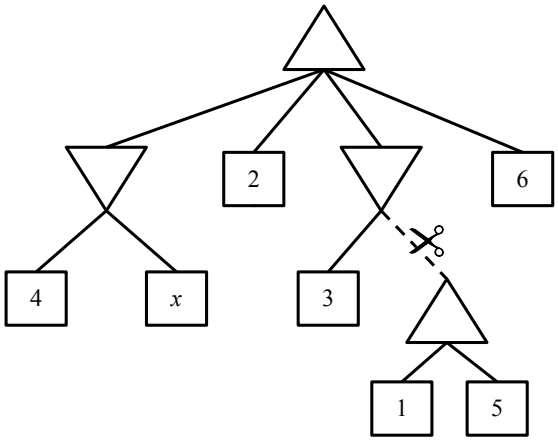
We are assuming that nodes are evaluated left to right and ties are broken in favor of the latter nodes. A different evaluation order would lead to different interval bounds, while a different tie breaking strategies could lead to strict inequalities ($>$ instead of $\geq$).
Successor enumeration order and tie breaking rules typically impact the efficiency of alpha-beta pruning.



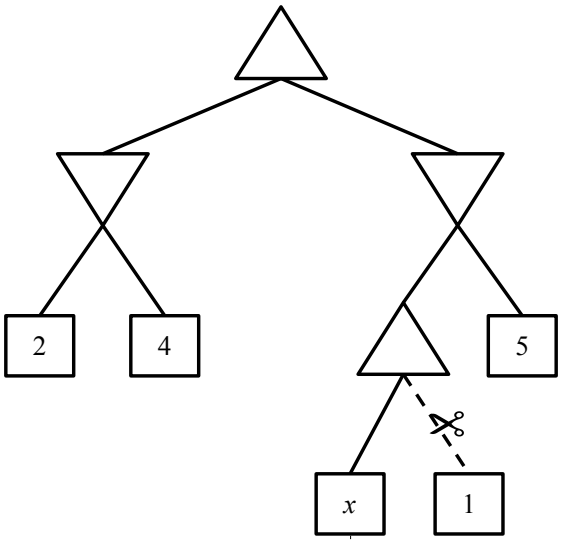
(b) Tree 1. Answer: None



(c) Tree 2. Answer: $x \geq 2$



(d) Tree 3. Answer: $x \geq 3$,

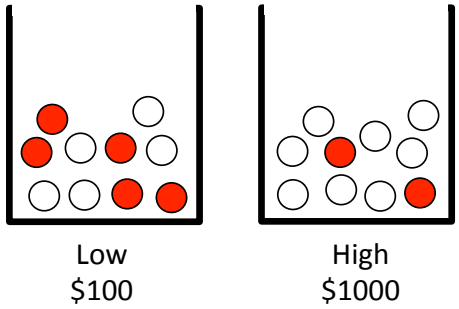


(e) Tree 4. Answer: None

Q8. [18 pts] Utilities: Low/High

After a tiring day of eating food and escaping from ghosts, Pacman heads to the casino for some well-deserved rest and relaxation! This particular casino has two games, Low and High, which are both free to play.

The two games are set up very similarly. In each game, there is a bin of marbles. The Low bin contains 5 white and 5 dark marbles, and the High bin contains 8 white and 2 dark marbles:



Play for each game proceeds as follows: the dealer draws a single marble at random from the bin. If a dark marble is drawn, the game pays out. The Low payout is \$100, and the High payout is \$1000. The payout is divided evenly among everyone playing that game. For example, if two people are playing Low and a dark marble is drawn, they each receive \$50. If a white marble is drawn, they receive nothing. The drawings for both games are done simultaneously, and only once per night (there is no repeated play).

(a) [2 pts] **Expectations.** Suppose Pacman is at the casino by himself (there are no other players). Give his expected winnings, in dollars:

(i) [1 pt] From playing a single round of Low: $\frac{5}{10} \cdot \$100 + \frac{5}{10} \cdot \$0 = \$50$

(ii) [1 pt] From playing a single round of High: $\frac{2}{10} \cdot \$1000 + \frac{8}{10} \cdot \$0 = \$200$

(b) [6 pts] **Preferences.** Pacman is still at the casino by himself. Let p denote the amount of money Pacman wins, and let his utility be given by some function $U(p)$. Assume that Pacman is a rational agent who acts to maximize expected utility.

(i) [3 pts] If you observe that Pacman chooses to play Low, which of the following must be true about $U(p)$? Assume $U(0) = 0$. (circle any that apply)

$U(50) \geq U(1000)$ $\quad$ $U(100) \geq U(1000)$

$\frac{1}{2}U(100) \geq \frac{2}{10}U(1000)$ ✓ $\quad$ $U(50) \geq U(100)$

Review Axioms of Rationality.

(ii) [3 pts] Given that Pacman plays Low, which of the following are possibilities for $U(p)$? You may use $\sqrt[3]{100} \approx 4.6$, although this question should not require extensive calculation. (circle any that apply)

p $\quad$ $-p$✓ $\quad$ $2^p - 1$ $\quad$ p^2 $\quad$ $\sqrt[3]{p}$✓

Check whether the response you gave for the previous question applies to these functions.

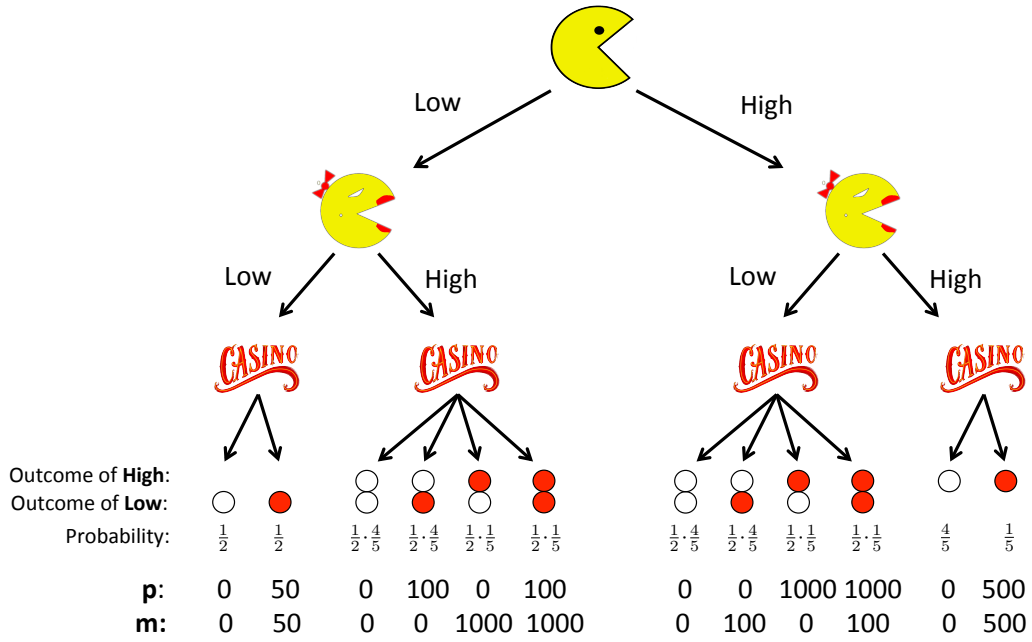


Figure 1: Game tree for Low/High as played by Pacman and Ms. Pacman.

(c) [10 pts] **Multiple Players**. Ms. Pacman is joining Pacman at the casino! Assume that Pacman arrives first and chooses which game he will play, and then Ms. Pacman arrives and chooses which game she will play. Let p denote Pacman's winnings and m denote Ms. Pacman's winnings. Since both Pacman and Ms. Pacman are rational agents, we can describe Pacman's utility with a function $U_1(p, m)$ and Ms. Pacman's utility with a function $U_2(p, m)$. You might find it helpful to refer to the game tree given in Figure 1.

(i) [6 pts] Suppose $U_1(p, m) = p$ and $U_2(p, m) = m$; that is, both players are attempting to maximize their own expected winnings. Compute the expected utilities of both players, for each combination of games they could play:

Pacman	**Ms. Pacman**	$\mathbb{E}[U_1(p, m)]$	$\mathbb{E}[U_2(p, m)]$
Low	Low	25	25
Low	High	50	200
High	Low	200	50
High	High	100	100

Recall that both games pay out only once, so if Mr. and Ms. Pacman play the same game, then the have to split the payout.

Given that Pacman chooses first, which of the following are possibilities for the games Pacman and Ms. Pacman respectively choose to play? (circle all that apply)

(Low, Low) (Low, High) (High, Low) (High, High)

You would model the problem as a minimax game tree since Mr. Pacman knows Ms. Pacman's utility.

(ii) [4 pts] **Scenarios**. Now rather than simply maximizing their own winnings, Pacman and Ms. Pacman have different objectives. Here are five utility functions $U_1(p, m)$ for Pacman:

$$p \qquad p+m \qquad m \qquad (p+m)^2 \qquad -m$$

and five utility functions $U_2(p, m)$ for Ms. Pacman:

$$m \qquad p+m \qquad -p \qquad 2m-p \qquad \log_{10} m$$

For each of the following scenarios, give the utility functions listed above which best encode the motivations of each player. A particular function may appear more than once. The first scenario is done for you.

Pacman	**Mrs. Pacman**	**Scenario**
p	m	Pacman and Ms. Pacman each want to maximize their own expected winnings.
-m	-p	Pacman and Ms. Pacman have had a terrible fight and are very angry at each other. Each wants the other to lose as much money as possible.
p+m	m	Pacman has gotten over the fight, and now wants to maximize their expected combined winnings (since Pacman and Ms. Pacman share a bank account). However, Ms. Pacman does not trust Pacman to deposit his share, so she just wants to maximize her own expected winnings.
m	m	Pacman is being extorted by the Ghost Mafia, who will immediately confiscate any money that he wins (that is, if Pacman wins \$100, he will still have $p = 100$ but does not actually get to keep the money). The Mafia is not monitoring Ms. Pacman and does not know about her winnings, so they will not be confiscated. Both Pacman and Ms. Pacman want to maximize the expected total amount the couple gets to keep.

Q9. [24 pts] MDPs and RL: Mini-Grids

The following problems take place in various scenarios of the gridworld MDP (as in Project 3). In all cases, A is the start state and double-rectangle states are exit states. From an exit state, the only action available is *Exit*, which results in the listed reward and ends the game (by moving into a terminal state X, not shown).

From non-exit states, the agent can choose either *Left* or *Right* actions, which move the agent in the corresponding direction. There are no living rewards; the only non-zero rewards come from exiting the grid.

Throughout this problem, assume that value iteration begins with initial values $V_0(s) = 0$ for all states s.

First, consider the following mini-grid. For now, the discount is $\gamma = 1$ and legal movement actions will always succeed (and so the state transition function is deterministic).

(a) [1 pt] What is the optimal value $V^*(A)$?

10

Since the discount $\gamma = 1$ and there are no rewards for any action other than exiting, a policy that simply heads to the right exit state and exits will accrue reward 10. This is the optimal policy, since the only alternative reward if 1, and so the optimal value function has value 10.

(b) [1 pt] When running value iteration, remember that we start with $V_0(s) = 0$ for all s. What is the first iteration k for which $V_k(A)$ will be non-zero?

2
The first reward is accrued when the agent does the following actions (state transitons) in sequence: Left, Exit. Since two state transitions are necessary before any possible reward, two iterations are necessary for the value function to become non-zero.

(c) [1 pt] What will $V_k(A)$ be when it is first non-zero?

1
As explained above, the first non-zero value function value will come from exiting out of the left exit cell, which accrues reward 1.

(d) [1 pt] After how many iterations k will we have $V_k(A) = V^*(A)$? If they will never become equal, write *never*.

4
The value function will equal the optimal value function when it discovers this sequence of state transitions: Right, Right, Right, Exit. This will obviously happen in 4 iterations.

Now the situation is as before, but the discount γ is less than 1.

(e) [2 pts] If $\gamma = 0.5$, what is the optimal value $V^*(A)$?

The optimal policy from A is Right, Right, Right, Exit. The rewards accrued by these state transitions are: 0, 0, 0, 10. The discount values are $\gamma^0, \gamma^1, \gamma^2, \gamma^3$, which is 1, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$. Therefore, $V^*(A) = 0 + 0 + 0 + \frac{10}{8}$.

(f) [2 pts] For what range of values γ of the discount will it be optimal to go *Right* from A? Remember that $0 \leq \gamma \leq 1$. Write *all* or *none* if all or no legal values of γ have this property.

The best reward accrued with the policy of going left is $\gamma^1 * 1$. The best reward accrued with the policy of going right is $\gamma^3 * 10$. We therefore have the inequality $10\gamma^3 \geq \gamma$, which simplifies to $\gamma \geq \sqrt{1/10}$. The final answer is $1/\sqrt{10} \leq \gamma \leq 1$

Let's kick it up a notch! The *Left* and *Right* movement actions are now stochastic and fail with probability f. When an action fails, the agent moves *up* or *down* with probability $f/2$ each. When there is no square to move *up* or *down* into (as in the one-dimensional case), the agent stays in place. The *Exit* action does not fail.

For the following mini-grid, the failure probability is $f = 0.5$. The discount is back to $\gamma = 1$.

(g) [1 pt] What is the optimal value $V^*(A)$?

10. Same reasoning as for the previous problem.

(h) [1 pt] When running value iteration, what is the smallest value of k for which $V_k(A)$ will be non-zero?

4. Same reasoning as for the previous problem, but now the only reward-accruing sequence of actions is Left, Left, Left, Exit.

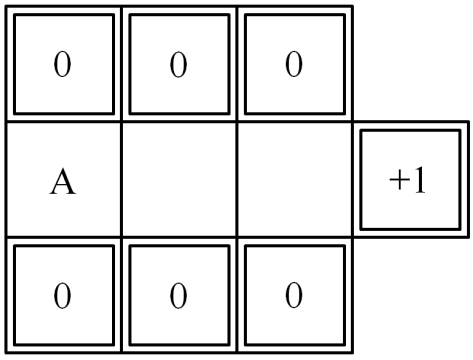
(i) [1 pt] What will $V_k(A)$ be when it is first non-zero?

10/8. Although $\gamma = 1$, the probability that the agent succesfully completes the sequence of actions that leads to a reward at $k = 4$ (Left, Left, Left, Exit) is only $\frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{8}$, as at each non-Exit step it has only a $\frac{1}{2}$ probability of success.

(j) [1 pt] After how many iterations k will we have $V_k(A) = V^*(A)$? If they will never become equal, write *never*.

Never. There is always only a $\frac{1}{2}$ probability of success on any movement action, so while V_k will asymptotically approach V^*, it won't ever equal it. Consider the square right next to the exit, which we'll call C: $V_{k+1}(C) = \frac{1}{2}10 + \frac{1}{2}V_k(C)$.

Now consider the following mini-grid. Again, the failure probability is $f = 0.5$ and $\gamma = 1$. Remember that failure results in a shift *up* or *down*, and that the only action available from the double-walled exit states is *Exit*.



(k) [1 pt] What is the optimal value $V^*(A)$?

1/8. Same reasoning as for the previous problem. Note that the exit node value is now only 1, not 10.

(l) [1 pt] When running value iteration, what is the smallest value of k for which $V_k(A)$ will be non-zero?

4

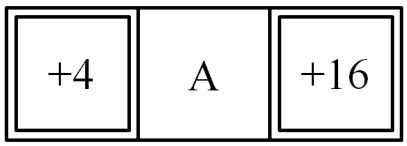
(m) [1 pt] What will $V_k(A)$ be when it is first non-zero?

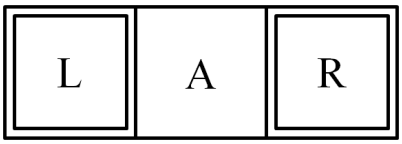
1/8

(n) [1 pt] After how many iterations k will we have $V_k(A) = V^*(A)$? If they will never become equal, write *never*.

4. This problem is different from the previous one, in that a state transition never fails by looping to the same state. Here, a movement action may fail, but that always moves the agent into an absorbing state.

Finally, consider the following mini-grid (rewards shown on left, state names shown on right).





In this scenario, the discount is $\gamma = 1$. The failure probability is actually $f = 0$, but, now we do not actually know the details of the MDP, so we use reinforcement learning to compute various values. We observe the following transition sequence (recall that state X is the end-of-game absorbing state):

s	a	s'	r
A	*Right*	R	0
R	*Exit*	X	16
A	*Left*	L	0
L	*Exit*	X	4
A	*Right*	R	0
R	*Exit*	X	16
A	*Left*	L	0
L	*Exit*	X	4

(o) [2 pts] After this sequence of transitions, if we use a learning rate of $\alpha = 0.5$, what would temporal difference learning learn for the value of A? Remember that $V(s)$ is intialized with 0 for all s.

3. Remember how temporal difference learning works: upon seeing a s, a, r, s' tuple, we update the value function as $V_{i+1}(s) = (1-\alpha)V_i(s) + \alpha(r + V_i(s'))$. To get the answer, simply write out a table of states, all initially with value 0, and then update it with information in each row of the table above. When all rows have been processed, see what value you ended up with for A.

(p) [2 pts] If these transitions repeated many times and learning rates were appropriately small for convergence, what would temporal difference learning converge to for the value of A?

10. We are simply updating the value function with the results of following this policy, and that's what we will converge to. For state A, the given tuples show the agent going right as often as it goes left Clearly, if the agent goes left as often as it goes right from A, the value of being in A is only $16/2 + 4/2 = 10$.

(q) [2 pts] After this sequence of transitions, if we use a learning rate of $\alpha = 0.5$, what would Q-learning learn for the Q-value of (A, *Right*)? Remember that $Q(s, a)$ is initialized with 0 for all (s, a).

4. The technique is the same as in problem (o), but use the Q-learning update (which includes a max). How do you get the max? Here's an example:

The sample sequence: $(A, Right, R, 0)$.

$Q(s, a) \leftarrow (1-\alpha)Q(s, a) + \alpha(r + \gamma \max_{a'}(s', a'))$.

$Q(A, right) \leftarrow (1-\alpha)Q(A, right) + \alpha(r + \gamma \max_{a'}(R, a'))$.

But since there is only one exit action from R, then:

$Q(A, right) \leftarrow (1-\alpha)Q(A, right) + \alpha(r + \gamma Q(R, Exit))$.

Note that this MDP is very small – you will finish the game in two moves (assuming you have to move from A).

(r) [2 pts] If these transitions repeated many times and learning rates were appropriately small for convergence, what would Q-learning converge to for the Q-value of (A, *Right*)?

16. Q-learning converges to the optimal Q-value function, if the states are fully explored and the convergence rate is set correctly.