

6.4110
Representation, Inference and Reasoning in AI

Quiz 2

March 18, 2026

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

You are permitted to use a single sheet of paper with notes on (both sides). You may not use a calculator. **Box all answers** for free response questions.

Name: _____

MIT email: _____

Question	Points	Score
1	20	
2	20	
3	30	
4	18	
5	12	
Total:	100	

1 Continuing resolution

1. We are going to consider several temporal stochastic processes. For each one, we supply the transition model $P(s_t | s_{t-1})$ and observation model $P(o_t | s_t)$ in the form of Python-ish functions that provide samples from the relevant distributions. The `normal_variate(m, s)` procedure draws a random sample from a normal distribution with mean `m` and variance `s**2`.

Process A

```
STATES = range(10)
def transition_A(s):
    if rand() < 0.5:
        return max(0, s - 1)
    else:
        return min(9, s + 1)

def observe_A(s_t):
    return random_choice([s, s-1, s+1])
```

Process B

```
def transition_B(x):
    if rand() < 0.5:
        return x - 1 + normal_variate(0, 1)
    else:
        return x + 1 + normal_variate(0, 1)

def observe_B(x):
    return normal_variate(x+1, 2)
```

Process C

```
def transition_C(x):
    return x**2 + normal_variate(0, 1)

def observe_C(x):
    return normal_variate(x, 2)
```

Process D

```
def transition_D(x):
    return x + normal_variate(0, 1)

def observe_D(x):
    return normal_variate(x, 0.01)
```

Process E

```
def transition_E(x, x_prev):
    return = 2*x + x_prev + normal_variate(0, 1)

def observe_E(x):
    return normal_variate(x, 1)
```

- (a) (15 points) We would like to build filters for each of these models. We are considering three modeling and filtering methods: HMM filter, Kalman filter, and particle filter. For each process, indicate which of these methods could be used effectively to model the problem. For each method that *you do not check*, explain in one sentence why it is not suitable.
- i. Process A HMM filter Kalman filter Particle filter

- ii. Process B HMM filter Kalman filter Particle filter

iii. Process C HMM filter Kalman filter Particle filter

iv. Process D HMM filter Kalman filter Particle filter

v. Process E HMM filter Kalman filter Particle filter

(b) (5 points) We are interested in using a particle-filter approximation to process D. Assume our initial distribution is uniform on the interval $[1, 1000]$. Describe **concretely** what difficulty we are likely to encounter.

2 Lattice entertain you!

2. You are modeling the state of a 3D solid as an equally spaced 3D grid of random variables, v_{ijk} where each i, j, k is in $\{0, \dots, M-1\}$. Each location can be in **one of two** states.

(a) (3 points) How many possible states does this system have?

You use a Markov Random Field (MRF) to model a distribution over possible states of your solid. You include a pairwise potential (factor) between each neighboring pair of v 's (which share two indices in common).

(b) (4 points) What is the approximate time required to compute an exact marginal distribution on one of the variables by computing the joint and marginalizing? Choose one answer and provide a short justification.

$O((2^{M^3})^2)$
 $O(M^6)$
 $O(M^5)$
 $O(2^{M^3})$
 $O(3^{M^2})$

(c) (4 points) What is the approximate computation time for one iteration of loopy belief propagation? Choose one answer and provide a short justification.

$O(1)$
 $O(2^{M^3})$
 $O(M^3)$
 $O(3^{M^2})$
 $O(M^2)$

6.4110

Quiz 2, March 18, 2026

(d) (5 points) Suppose the pairwise potential between neighboring variables is given by

a	b	$\phi(a, b)$
0	0	3
0	1	2
1	0	2
1	1	3

Assume the current values of the six neighbors of V_{345} are

$$V_{245} = 1, \quad V_{445} = 1, \quad V_{335} = 0, \quad V_{355} = 0, \quad V_{344} = 0, \quad V_{346} = 1.$$

What probability distribution would you use to sample a new value of V_{345} in Gibbs sampling?

(e) (4 points) You would like to compute the probability that more than half of the locations of the solid are in state 1. Here is incomplete pseudocode for using Gibbs sampling to solve this problem. Fill in the empty boxes with Python-like code.

You don't need to worry about burn-in or skipping samples.

1. Initialize values $\bar{v} = (v_1, \dots, v_{M^3})$ at random from $\{0, 1\}$
- 2.

3. For N iterations:
 - Choose i, j, k randomly from $1, \dots, M$
 - Set v_{ijk} to a sample from

$$P(V_{ijk} \mid (V \setminus V_{ijk}) = (\bar{v} \setminus v_{ijk}))$$

•

4. Return

3 Staying dry

3. You are at MIT, where it might be raining, and you need to get to class! You can move from building to building outside and risk getting wet or go between some buildings via tunnels and stay dry.

You have:

- A set of buildings \mathcal{B}
- The locations of the buildings, which we will treat as points in x, y space for simplicity, defined by $L(b)$ for $b \in \mathcal{B}$
- A function $F(b_1, b_2)$ specifying the length of the shortest tunnel between buildings b_1 and b_2 , which has value ∞ if there is no tunnel connecting them
- A *perfectly accurate* weather forecast w : the value of $w(t_1, t_2)$ is the number of seconds, during the time interval $[t_1, t_2]$, it will be raining (the output will be between 0 and $t_2 - t_1$)
- A fixed walking velocity v

Your goal will be to find a path from a starting building b_0 to a target building b_g that minimizes $d + 3r$ where d is the length of time you travel in which you're *not* getting rained on and r is the length of time you travel in which you *are* getting rained on. If you travel between buildings outdoors, you'll take the shortest Euclidean path.

- (a) (3 points) Assuming that you are not allowed to just wait around (this is MIT!) and have to leave each building as soon as you arrive, until you reach your destination, what is the action space in this problem?

- (b) (5 points) Assume the state space is defined as $\mathcal{B} \times \mathcal{T}$ where \mathcal{T} is the set of times, modeled as real numbers. So, a state has the form (b, t) .

Given a state (b, t) and the action of walking *outside* to building b' , define the next state in terms of b and b' .

6.4110

Quiz 2, March 18, 2026

- (c) Given a state (b, t) and the action of walking *outside* to building b' , define the step cost in terms of b , and b' .

- (d) (5 points) Bob is worried that this problem can't be well formalized as a min-cost path search problem because we made the state space continuous by including time. Alice disagrees. Alice is right! Explain why.

- (e) (5 points) Bob is convinced by Alice's argument, but says he thinks it would be better if he were allowed to wait inside buildings for the rain to stop. But he's worried that if we do that, then the problem can't be well formalized as a min-cost path search problem. This time, Bob is right! Explain why.

6.4110

Quiz 2, March 18, 2026

- (f) (12 points) Which of the following search methods is guaranteed to find a cost-optimal solution in our original problem, without the ability to wait?

If you answer "Not optimal," explain why.

- i. uniform-cost search

optimal not optimal

- ii. A* search with $h((b, t))$ equal to 3 times the amount of time required to walk a shortest path from b to the goal

optimal not optimal

- iii. A* search, with $h((b, t))$ equal to the cost of the optimal solution to a version of the problem where there are no tunnels

optimal not optimal

- iv. A* search, with $h((b, t))$ equal to the cost of the optimal solution to a version of the problem where the shortest path between each building pair goes through a tunnel

optimal not optimal

4 Shop the job

4. Let's return to our simple job-shop scheduling problem, with some modifications.

- There are two machines: painter, dryer
- Each part must be painted and then dried, in that order.
- The painter can only accommodate one part at a time.
- The dryer can accommodate two parts at a time.

We'll use the following predicates:

```
(painted ?part)
(dried ?part)
(location ?part ?location)
```

As a reminder of the syntax, here is a PDDL operator description for moving a part from one location to another. You can assume we have locations: `storage-shelf`, `painter`, `dryer`.

```
(:action move
:parameters (?part ?from ?to)
:precondition (and
  (location ?part ?from))
:effect (and
  (not (location ?part ?from))
  (location ?part ?to)))
```

For simplicity, in the above operator we are omitting modeling the fact that there must be space in the machine in order to move a part into it.

- (a) (5 points) Write the PDDL operator description for drying two parts simultaneously.

6.4110

Quiz 2, March 18, 2026

- (b) (3 points) Provide a plan with the minimal number of steps for producing 2 dried parts, assuming the parts all start neither painted, nor dried, on the storage shelf. Available actions are `move`, `paint`, `dry-two`.

For ease of grading, write your answer as a sequence of grounded PDDL actions, for example:

```
(paint part1)
(move part2 storage-shelf dryer)
```

For the remaining questions, assume the following PDDL problem description.

```
(define (problem finish-two-parts)
  (:domain job-shop)
  (:objects p1 p2 storage-shelf painter dryer)
  (:init
    (location p1 storage-shelf)
    (location p2 storage-shelf))
  (:goal
    (dried p1) (dried p2)))
```

- (c) (4 points) The relaxed plan graph lets us compute the sets of new fluents achievable at level l , F_l , for all the levels. For each level, list the fluents newly achievable at that level. Stop when you have included the goal fluents.

6.4110

Quiz 2, March 18, 2026

(d) (2 points) What is h_{\max} for (and (dried p1) (dried p2)) ?

(e) (2 points) What is h_{add} for (and (dried p1) (dried p2)) ?

(f) (2 points) Which, if any, of these two heuristics is admissible in general? Explain your answer.

h_{\max} h_{add}

5 MCTStar

5. We're going to apply UCT to a minimum-cost-path problem! We will:

- Add a new state to our problem, called **done**
- If we reach any state $s \in \mathcal{G}$, we go to state **done**
- Assume every action taken from the **done** state returns to **done**
- Assume the cost of each action to 1, so $R(s, a) = -1$ for all actions in all states, except $R(\text{done}, a) = 0$
- Define $\text{UCT_path}(s_0, (A, T, R, H), M)$, to call $\text{UCT}(s_0, (A, T, R, H), M)$ and return the path in the tree constructed so far that has the largest summed reward

The pseudocode for UCT from the notes is on the next page for reference.

(a) Generally, in a minimum-cost path problem, we are not sure how many steps the optimal path will have. We have come up with several strategies for handling this. For each, explain whether it will *eventually* (assuming we never terminate the outer loop) find an optimal path to a goal state and if not why not.

i. (3 points) Make a call to UCT_path with $H = \infty$, and $M = \infty$, letting all of the random roll-outs run until they hit a state in \mathcal{G} .

- Guaranteed optimal Not guaranteed optimal

ii. (3 points) Pick a number M of samples, and then do a nested loop increasing H and starting the search fresh each time, and remembering the best solution found for each H . The question is: will `sols` eventually contain an optimal solution?

```
H = 1 ; sols = {}
```

```
while True:
```

```
    sols[H] = UCT_path(s0, (A, T, R, H), M)
```

```
    H = H + 1
```

- Guaranteed optimal Not guaranteed optimal

6.4110

Quiz 2, March 18, 2026

- iii. (3 points) Pick a number M of samples, and then do a nested loop, increasing H each time, but retaining the tree including all of the N and U values from the previous iteration

```
H = 1
```

```
sol = UCT_path(s0, (A, T, R, H), M)
```

```
while True:
```

```
    if sol reaches G:
```

```
        return sol
```

```
    H = H + 1
```

```
    Retaining the existing UCT tree, run M more  
    iterations with the increased H
```

- Guaranteed optimal Not guaranteed optimal

- (b) (3 points) We happen to have an admissible heuristic h . Your friend suggests that, instead of random roll-outs, we should behave greedily with respect to h to roll-out a leaf node—that is, in line 3 of SIMULATE, choose the action that produces an s' that minimizes $h(s')$. Is that a good or bad idea? Explain.

A UCT pseudocode

MCTS($s_0, (\mathcal{A}, T, R, H), iters$)

```

1  root = NODE( $s_0, horizon = H, parent = \mathbf{None}, children = \{ \}, U = 0, N = 0$ )
2  for iter  $\in \{1, \dots, iters\}$ :
3      leaf = SELECT( $root$ )
4      child = EXPAND( $leaf, \mathcal{A}, T$ )
5      value = SIMULATE( $child, \mathcal{A}, T, R$ )
6      BACKUP( $child, value$ )
7  max_child = max( $root.children, key = \lambda n. n.U/n.N$ )
8  return root.children[max_child] // Returns the associated action

```

SELECT(n)

```

// Follow optimistically best path through tree
1  if n.children
2      return SELECT(max( $n.children, key = \lambda c. UCB(n.N, c.N, c.U)$ ))
3  else
4      return n

```

EXPAND(n, \mathcal{A}, T)

```

// Unless remaining horizon is 0, add child nodes and return one
1  if n.horizon = 0:
2      return n
3  else
4      for a  $\in \mathcal{A}$ :
5          s' = T( $n.s, a$ )
6          n' = NODE( $s', n.horizon - 1, parent = n, children = \{ \}, U = 0, N = 0$ )
7          n.children[n'] = a
8      return RANDOM_CHOICE( $n.children$ )

```

SIMULATE(n, \mathcal{A}, T, R)

```

// Randomly finish path and return cumulative reward
1  s = n.s; total_reward = 0
2  for h  $\in (n.horizon, \dots, 1)$ :
3      a = RANDOM_CHOICE( $\mathcal{A}$ )
4      s' = T( $s, a$ )
5      total_reward += R( $s, a, s'$ )
6      s = s'
7  return total_reward

```

BACKUP(n, v_below)

```

// Add value v to n's statistics and pass it up
1  n.N += 1
2  if n.parent:
3      a = n.parent.children[n] // Action that led to n
4      v = v_below + R( $n.parent.s, a, n.s$ ) // Value of executing a in parent
5      n.U += v
6      BACKUP( $n.parent, v$ )

```