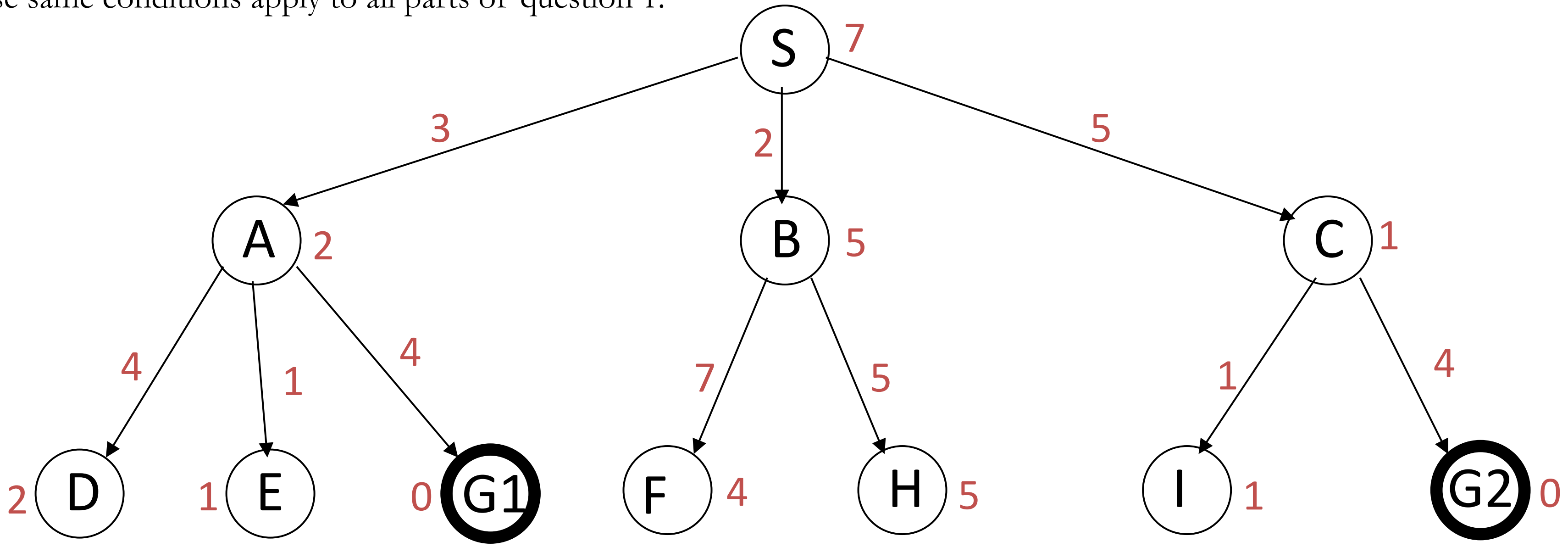
Exam 1 CS x260 Name: ____

I will not use a source other than my brain on this exam (not neighbors, not notes, not books, etc):

____________________________________ (please sign)

1. Consider the search graph below. The h value of a node is given adjacent to that node. The actual cost of traversing an arc is given adjacent to that arc. Node S is the start/initial state. Nodes G1 and G2 are goals. Leaf states/nodes have no successors. These same conditions apply to all parts of question 1.

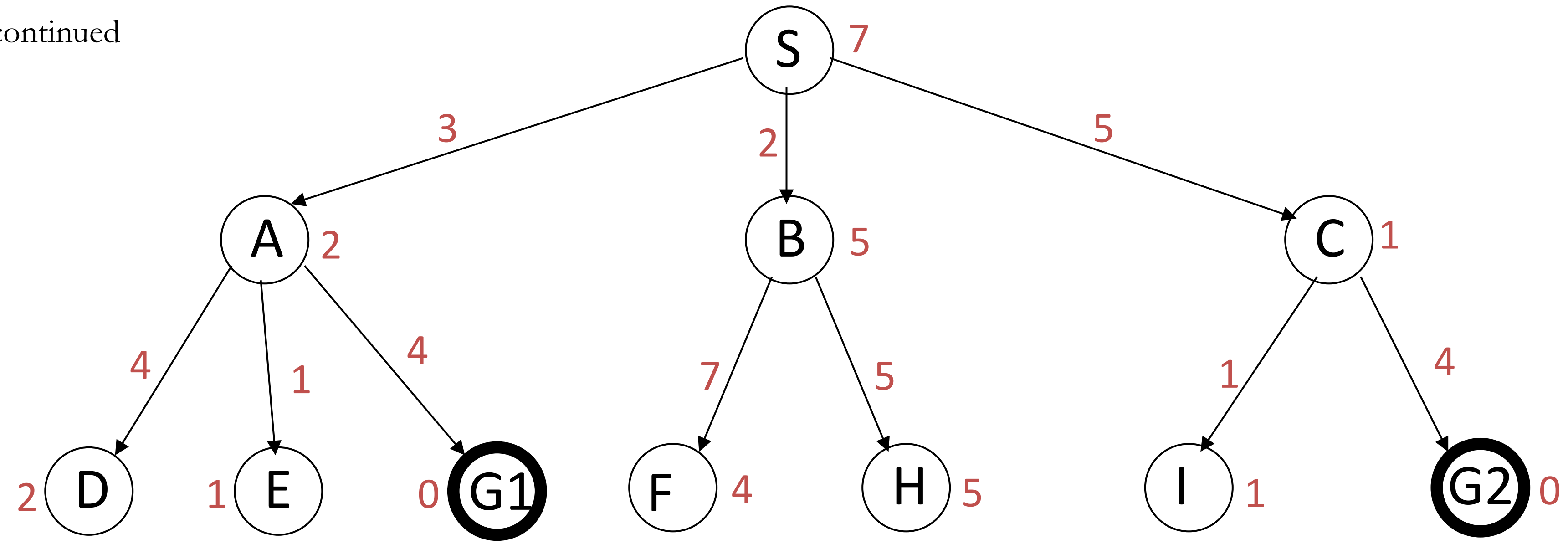


a) (1 pt) Is h consistent? Yes or No. Explain.

b) (2 pts) Is h admissible? Yes or No. Explain.

3 minutes

1 continued



c) (3 pts) Give the order in which nodes are visited (i.e., checked for goalness) by **heuristic depth first search**. In the case of two or more nodes with the same evaluation score on the frontier, break the tie by visiting the nodes in alphabetical order as labeled above – this same convention applies to the remaining parts of this question.

____

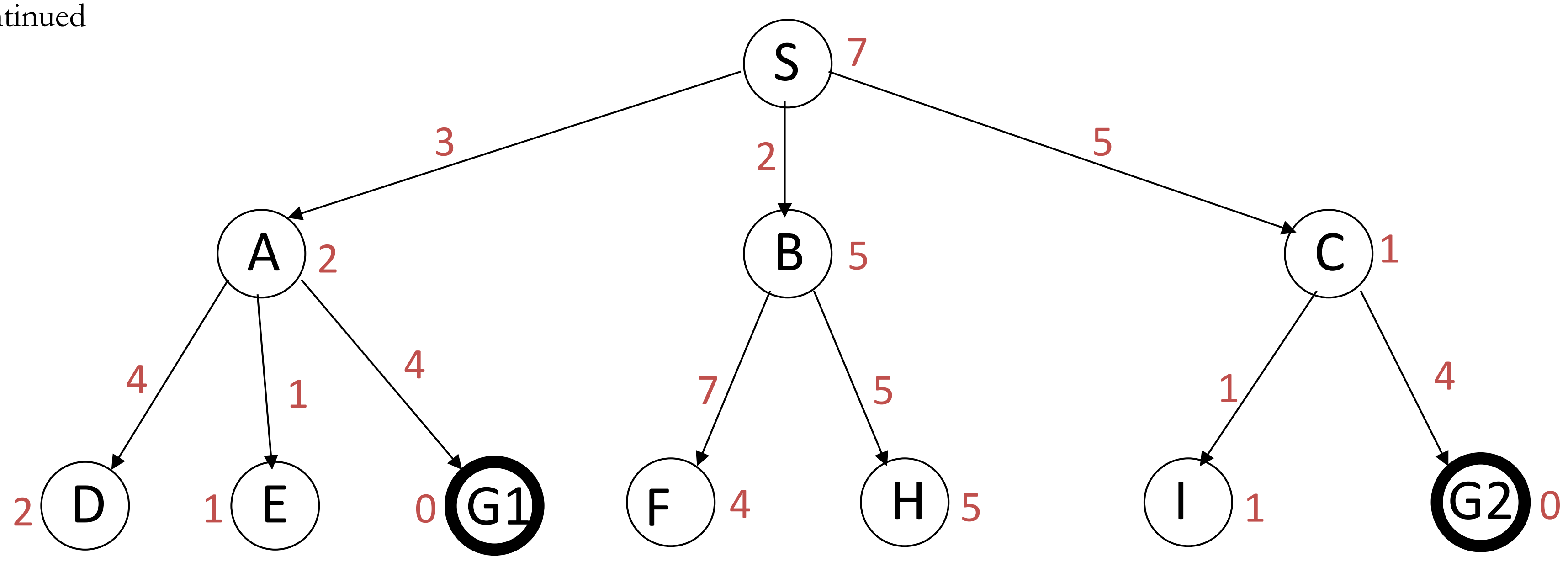
d) (3 pts) Give the order in which nodes are visited (i.e., checked for goalness) by **lowest cost-first search**.

____

e) (3 pts) Give the order in which nodes are visited (i.e., checked for goalness) by **greedy best-first search**.

____ 4 minutes

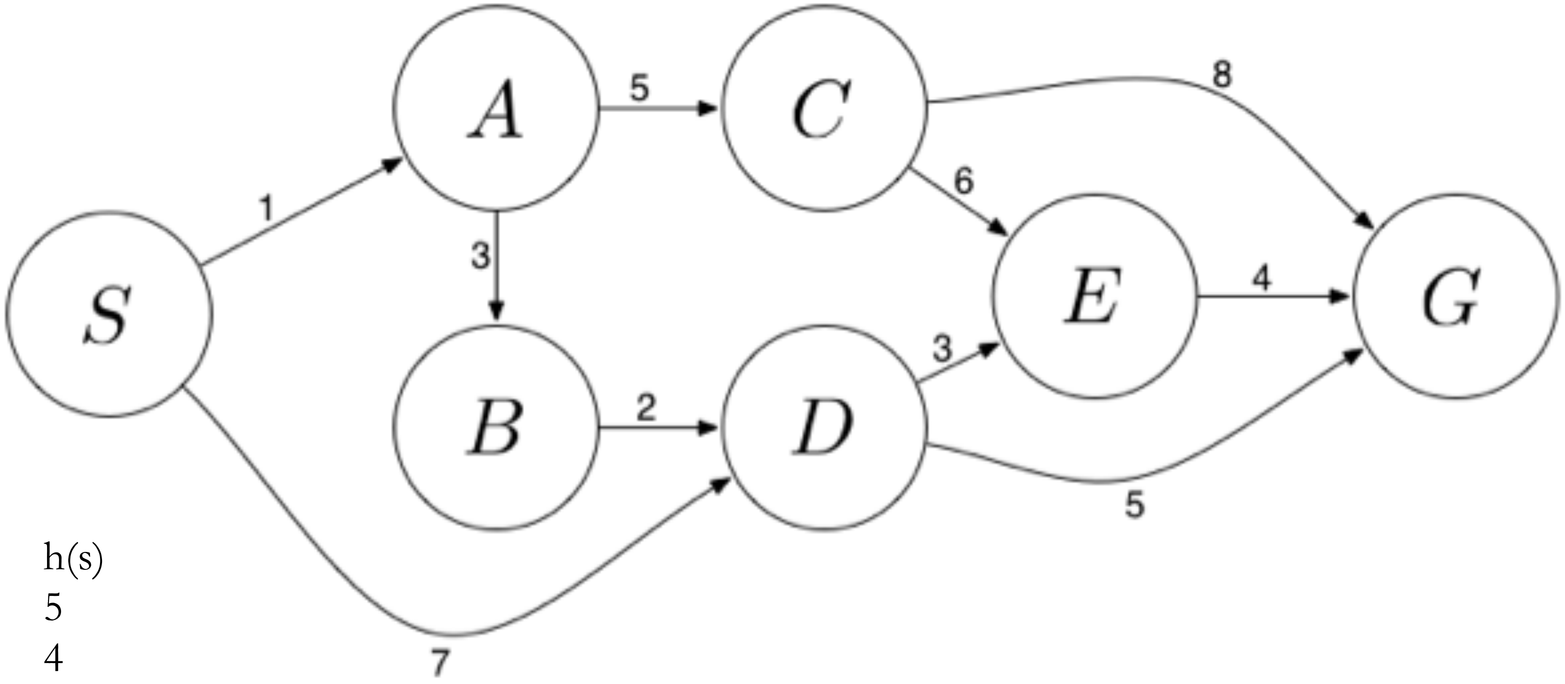
1 continued



f) (3 pts) Give the order in which nodes are visited (i.e., checked for goalness) by **A***.

3 minutes

2. (1 pt) Use this search graph (together with actual costs that label arcs, and the heuristic estimates in the table to the bottom left), as necessary, to answer this question. S is the start state and G is the only goal. This problem is worth 1 point – save it for last.



State s	h(s)
S	5
A	4
B	4
C	7
D	1
E	4
G	0

In question 1 you indicated the order in which states were visited, but the generic search algorithm in the book associates a path with each state removed from the frontier. In this question your answer will be a path. For example, a depth first search would return path S→A→B→D→E→G (assuming we break ties on the frontier alphabetically as labeled).

What path would **iterative deepening A*** return?

3. Consider the propositional knowledge base, KB:

$p \leftarrow q \wedge r \wedge s$

$q \leftarrow y \wedge u$

$m \leftarrow y \wedge z$

$y \leftarrow m$

$z \leftarrow m$

$z \leftarrow r \wedge x$

$s \leftarrow w$

r

w

y

u

a) (3 pts) Give a bottom-up proof of **p**, or explain why no such proof exists

b) (3 pts) Give a top-down proof of **m**, or explain why no such proof exists

d) (3 pts) Give a proof of **s** by contradiction using resolution (aka a resolution refutation proof). Convert any elements of the KB into clause form, as needed for this demonstration, and give the converted clauses.

5 minutes

4. Consider the following (feature-based) STRIPS-style operators:

Op1: precondition [p, q];
effects [~p, r]

Op2: precondition [q, r]
effects [~q, ~r, s]

Consider the goal to achieve is [r, q, w] and the **initial state** of the world is [p, s, w].

(a) (3 pts) Expand the **start state** of a *regression* planner, when run on this problem, giving the IMMEDIATE CHILDREN of the *start state* in the regression planner search:

(b) (2 pts) Of the two planning strategies, progression (forward) and regression (backward), which would you rely on to give you the quickest idea of what the "world" circumstances must be like if the planner is to achieve its goals:

__________________ ?

Which would give you the quickest idea of the way that the "world" circumstances can be in the near term:

_____________________________ ?

4 minutes

5. Consider the following macro/composite operator in STRIPS notation – this is a block stacking application of the type seen in lecture.

$$\begin{tikzcd}\textbf{Unstack-A-B} \arrow[r] & \textbf{PutDown-A} \arrow[r] & \textbf{Pickup-B} \arrow[r] & \textbf{Stack-B-C}\end{tikzcd}$$

on-A-B	~on-A-B	holding-A	~holding-A	onTab-B	~ onTab-B	holding-B	~holding-B
clear-A	~clear-A		clear-A	clear-B	~clear-B	clear-C	~clear-C
handEmpty	~handEmpty		handEmpty	handEmpty	~handEmpty		handEmpty
	holding-A		onTable-A		holding-B		clear-B
	clear-B						on-B-C
Preconditions	**Effects**	**Preconditions**	**Effects**	**Preconditions**	**Effects**	**Preconditions**	**Effects**

The basic operators making up the composite operator are labeled along the top (Unstack-A-B, PutDown-A, Pickup-B, Stack-B-C), with preconditions of each given below the operator name and to the left; effects to the right.

a) (2 pts) Give the preconditions of this macro/composite operator

b) (2 pts) Give the effects of this macro/composite operator. You need only list un-negated effects (because we can build in a KB that allows reason from un-negated propositions to obtain the relevant negated propositions. For example, handEmpty → ~holding-A; handEmpty → ~holding-B; …; on-A-B → ~clear-B; on-B-C → ~clear-C, …)

5 minutes

6. Consider a CSP with the variables X, Y, Z, each with domain {1, 2, 3, 4}. Suppose the constraints are X > Y and Y > Z.

a) (3 pts) Draw the constraint network after applying the generalized arc consistency (GAC) algorithm to this CSP.

b) (3 pts) Eliminate variable Y in the network of part (a) -- that is, assuming the reduced domains obtained through the GAC algorithm. Show the new constraint on X and Z that results. We will try to grade this so that we minimize the cascading of errors (i.e., if you get part a wrong).

10 minutes

7.(a) (1 pt) Suppose that you have a search *tree* with a (maximum) branching factor of B. How many nodes will BOUNDED depth-first search generate and place on the frontier in the worst case if the depth bound is D. Assume that when a node is expanded, all of its children are generated and placed on the Frontier (aka fringe) at once. Give an exact answer (not an O-notation expression), but you need not "simplify" your answer. Write clearly!

(b) (1 pt) Give the asymptotic TIME complexity (big-O expression) of a bounded depth-first search as a function of D and B assuming that time is proportional to number of states generated.

(c) (1 pt) Give the asymptotic SPACE complexity (big-O expression) of a bounded DFS as a function of D and B assuming that space is proportional to the maximum number of states that must be retained in memory simultaneously.

(d) (1 pt) Give the asymptotic TIME complexity (big-O expression) of a BREADTH-FIRST SEARCH (BFS) as outlined above (branching factor B to depth D) assuming that time is proportional to number of states generated.

(e) (1 pt) Give the asymptotic SPACE complexity (big-O expression) of a BFS as outlined above (branching factor B, max depth D) assuming that space is proportional to the maximum number of states that must be retained in memory simultaneously, as a function of D and B.

5 minutes

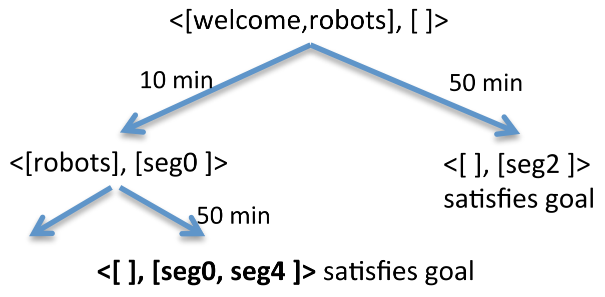
8. 4 pts) . This question investigates using graph searching to design video presentations. Suppose there exists a database of video segments, together with their length in seconds and the topics covered, set up as follows:

Segment	Length	Topics covered
seg0	10	[welcome]
seg1	30	[skiing, views]
seg2	50	[welcome, artificial_intelligence, robots]
seg3	40	[graphics, dragons]
seg4	50	[skiing, robots]

Represent a node as a pair:

$$\langle To\_Cover, Segs \rangle ,$$

Taken from Poole and Mackworth, Artificial Intelligence



<[welcome,robots], [ ]>
10 min
50 min
<[robots], [seg0]>
<[ ], [seg2]> satisfies goal
50 min
<[ ], [seg0, seg4]> satisfies goal

where $Segs$ is a list of segments that must be in the presentation, and To_Cover is a list of topics that also must be covered. Assume that none of the segments in $Segs$ cover any of the topics in To_Cover.

The neighbors of a node are obtained by first selecting a topic from To_Cover. There is a neighbor for each segment that covers the selected topic. [Part of this exercise is to think about the exact structure of these neighbors.]

For example, given the aforementioned database of segments, the neighbors of the node $\langle [welcome, robots], [] \rangle$, assuming that $welcome$ was selected, are $\langle [], [seg2] \rangle$ and $\langle [robots], [seg0] \rangle$.

Thus, each arc adds exactly one segment but can cover one or more topics. Suppose that the cost of the arc is equal to the time of the segment added.

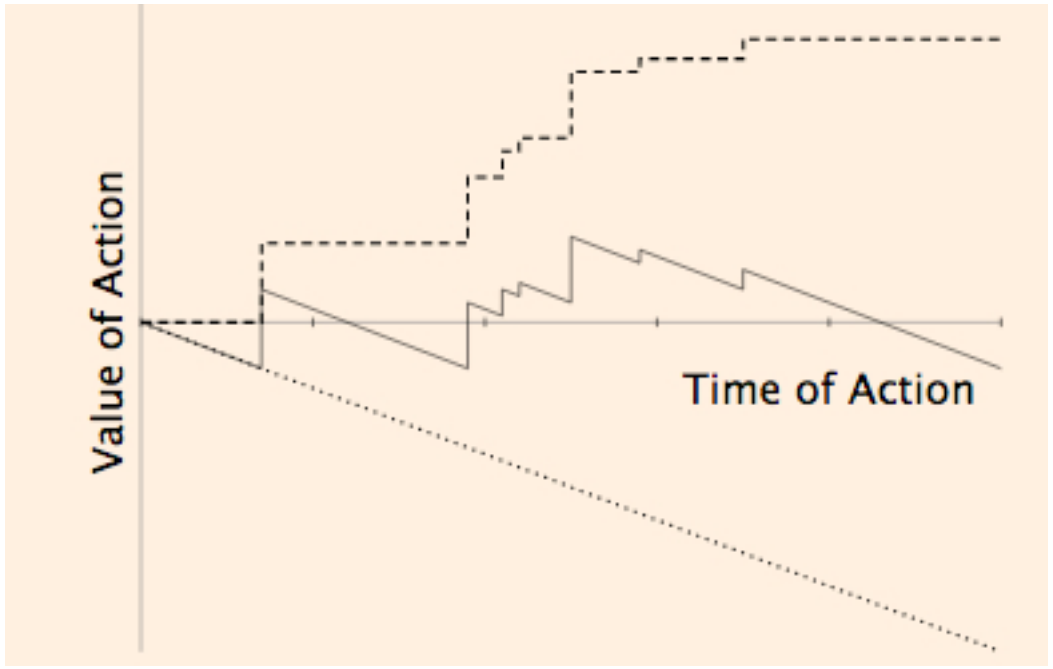
The goal is to design a presentation that covers all of the topics in $MustCover$. The starting node is $\langle MustCover, [] \rangle$, and the goal nodes are of the form $\langle [], Presentation \rangle$. The cost of the path from a start node to a goal node is the time of the presentation. Thus, an optimal presentation is a shortest presentation that covers all of the topics in $MustCover$.

(a) Suppose that the goal is to cover the topics $[welcome, skiing, robots]$ and the algorithm always select the leftmost topic to find the neighbors for each node. Draw the search space expanded for a lowest-cost-first search until the first solution is found. This should show all nodes expanded, which node is a goal node, and the frontier when the goal was found.

15 minutes

Answer to question 8 here

9. (2 pts) An anytime algorithm continues to search for solutions even after the first solution, and subsequent solutions, are found. It is called "anytime" because it can output (or execute) the best solution found so far, any time a solution is required. Unfortunately, there is often a penalty associated with waiting to output/execute a solution, which the agent may not know about. This graph, taken from an Example by Poole and Mackworth (our textbook), illustrates that the quality of solutions found through reasoning increase with time by the top-most dashed line (we assume that the agent remembers and can output the best solution found so far, so this line will never decrease). But the quality of solutions implied by this line don't take into account the cost of waiting to act on the solution. The bottom-most dotted line shows the penalty associated with waiting to act on a solution.



a) Put an 'X' along the timeline at the point where it is best for the agent to take action on the best solution found so far (even if the agent doesn't necessarily know that the time you indicate is best)

b) If the agent remembered solutions found on prior searches, how could these remembered solutions be used to increase the agent's performance when the same problems were encountered again?

3 minutes