Started on	Tuesday, 31 October 2023, 7:55 PM
State	Finished
Completed on	Tuesday, 31 October 2023, 9:03 PM
Time taken	1 hour 7 mins

Information

Frontier trace format

This information box appears in any quiz that includes questions on tracing the frontier of a graph search problem.

Trace format

- Starting with an empty frontier we record all the calls to the frontier: to add or to get a path. We dedicate one line per call.
- When we ask the frontier to add a path, we start the line with a `+` followed by the path that is being added.
- When we ask for a path from the frontier, we start the line with a `-` followed by the path that is being removed.
- When using a priority queue, the path is followed by a comma and then the key (e.g. cost, heuristic, f-value, ...).
- The lines of the trace should match the following regular expression (case and space insensitive): `^[+-][a-z]+(,\d+)?!?$`
- The symbol `!` is used at the end of a line to indicate a pruning operation. It must be used when the path that is being added to, or is removed from the frontier is to a node that is already expanded. Note that if a path that is removed from the frontier is pruned, it is not returned to the search algorithm; the frontier has to continue removing paths until it removes one that can be returned.

Precheck

You can check the format of your answer by clicking the "Precheck" button which checks the format of each line against a regular expression. Precheck only looks at the syntax of your answer and the frontier that must be used. It does not use the information in the graph object (so for example, `+ a` would pass the format check for a BFS or DFS question even if the graph does not have a node called `a`).

If your answer fails the precheck, it will fail the check. If it passes the precheck, it may pass the main check, it may not. You will not lose or gain any marks by using "Precheck".

Notes

- All state-space graphs are directed.
- In explicit graphs, the order of arcs is determined by the order of elements in `edge_list`.
- Frontier traces are not case sensitive - you don't have to type capital letters.
- There can be any number of white space characters between characters in a trace line and in between lines. When evaluating your answer, white spaces will be removed even if they are between characters.
- You can have comments in a trace. Comments start with a hash sign, #. They can take an entire line or just appear at the end of a line. Any character string appearing after # in a line will be ignored by the grader. You can use comments to keep track of things such as the *expanded* set (when pruning).
- In questions where the search strategy is cost-sensitive (e.g. LCFS) you must include the cost information on all lines. This means that in every trace line, the path is followed by a comma and a number.
- In questions where a priority queue is needed, the queue must be stable.

Question 1

Correct

Marked out of 6.00

Given the following graph, trace the frontier of a DFS search without pruning.

```
ExplicitGraph(
    nodes={'E', 'D', 'H', 'A', 'G', 'B'},
    edge_list=[('D', 'H'), ('E', 'D'), ('G','B'), ('E','A'), ('E','G')],
    starting_nodes=['E'],
    goal_nodes={'A'},
    )
```

Note that like other explicit graphs, the edges are directed.

Answer: (penalty regime: 20, 40, ... %)

```
+e
-e
+ed
+ea
+eg
-eg
+egb
-egb
-ea
```

+e (OK)
-e (OK)
+ed (OK)
+ea (OK)
+eg (OK)
-eg (OK)
+egb (OK)
-egb (OK)
-ea (OK)

Passed all tests! ✓

Question 2

Correct

Marked out of 7.00

Given the following graph, trace the frontier of an A-Star search **with multiple-path pruning**.

```
ExplicitGraph(
    nodes={'E', 'D', 'H', 'A', 'G'},
    estimates={'H':9, 'E':6, 'D':5, 'A':0, 'G':7},
    edge_list=[('G','D',2), ('D','A',6), ('H','E',6), ('H','D',5), ('H','G',2), ('E','A',2)],
    starting_nodes=['H'],
    goal_nodes={'A'},
    )
```

Notes

- This is a cost-sensitive search. All the trace lines must have a comma followed by a number.
- If pruning is necessary, finish the line with an exclamation mark.
- Priority queues must be stable.

Answer: (penalty regime: 20, 40, ... %)

```
+h,9
-h,9 #h
+he,12
+hd,10
+hg,9
-hg,9 #hg
+hgd,9
-hgd,9 #hgd
+hgda,10
-hd,10!
-hgda,10
```

+h,9 (OK)
-h,9 (OK)
+he,12 (OK)
+hd,10 (OK)
+hg,9 (OK)
-hg,9 (OK)
+hgd,9 (OK)
-hgd,9 (OK)
+hgda,10 (OK)
-hd,10! (OK)
-hgda,10 (OK)

Passed all tests! ✓

Question 3

Correct

Marked out of 2.00

We have an agent called `A` that operates in a grid environment. The agent can move in four directions: up, down, left, and right. The agent is provided with a map of the environment in the form of a multi-line string. The agent has to reach to a location marked with `G` and it can only pass through locations that are empty (blank).

We have implemented a Graph class and an A* frontier class that work together; they find optimal solutions (paths) when one exists. However, in scenarios like the following where there is no solution, the search does **not** halt.

```
+----------+
|    X    G|
| A  X     |
|    X  G  |
+----------+
```

Assuming that all arc costs are greater than or equal to 1, all heuristic values are non-negative, and both of them are finite, which of the following could be a reason for this bug?

Select one:

- [✓ a. No pruning is implemented.] ✓
- b. The frontier is not stable.
- c. The order of edges (returned by `outgoing_arcs`) is incorrect.
- d. The heuristic function is not admissible.
- e. The heuristic function is not monotone.

Question 4

Partially correct

Marked out of 8.00

In the context of propositional logic, consider the following knowledge base.

```
c :- e, a.
a :- d, b.
d.
b :- d.
c.
```

Answer the following questions with integers. Use numerical digits (not words or expressions).

- How many definite clauses are there in this knowledge base? 5 ✓
- How many distinct atoms are there in this knowledge base? 5 ✓
- In how many models is `a` true? 16 ✗
- In how many models is `c` true? 16 ✗
- In how many interpretations are both `c` and `e` true? 1 ✗
- In how many interpretations are both `b` and `d` false? 0 ✗
- In how many models are one or more atoms false? 1 ✓
- In how many models is `a` false? 0 ✓

Question 5

Correct

Marked out of 3.00

Write a predicate `even_length(?List)` that holds if `List` is a list with an even number of elements (i.e the length of the list is even).

You must write this predicate using the usual techniques covered in the course. Do not use the built-in `length` predicate or any solution that requires arithmetic. The Precheck button checks whether this condition is satisfied. Your answer will fail if it contains tokens/strings such as `'length'`, `'is'`, etc. Please note that the Precheck button does not run any of the (visible or invisible) test cases.

For example:

Test	Result
`test_answer :- even_length([]), even_length([foo, bar, zoo, log]), writeln('OK').`	OK
`test_answer :- \+ even_length([1]), \+ even_length(this_is_not_a_list), writeln('OK').`	OK

Answer: (penalty regime: 0, 20, ... %)

```
even_length([]).
even_length([_,_|L]) :- even_length(L).
```

	Test	Expected	Got	
✓	`test_answer :- even_length([]), even_length([foo, bar, zoo, log]), writeln('OK').`	OK	OK	✓
✓	`test_answer :- \+ even_length([1]), \+ even_length(this_is_not_a_list), writeln('OK').`	OK	OK	✓

Passed all tests! ✓

Question 6

Correct

Marked out of 4.00

Write a predicate `same_evens(?List)` that holds if `List` is a list whose length is even and every second element is the same (i.e. when available, the second, fourth, sixth, ... elements are all the same).

You must write this predicate using the usual techniques covered in the course. Do not use the built-in `length` predicate or any solution that requires arithmetic. The Precheck button checks whether this condition is satisfied. Your answer will fail if it contains tokens/strings such as '`length`', '`is`', etc. Please note that the Precheck button does not run any of the (visible or invisible) test cases.

For example:

Test	Result

`test_answer :-`
`    same_evens([a, b, c, b, d, b]),`
`    same_evens([a, b]),`
`    same_evens([]),`
`    writeln('OK').`
	`OK`

Answer: (penalty regime: 0, 20, ... %)

```
same_evens([]).
same_evens([_, _]).

second(_, []).
second(Y, [_,Y|_]).
same_evens([_, Y | T]) :- same_evens(T), second(Y, T).
```

	Test	Expected	Got	

✓	`test_answer :-`
	`    same_evens([a, b, c, b, d, b]),`
	`    same_evens([a, b]),`
	`    same_evens([]),`
	`    writeln('OK').`
		`OK`	`OK`	✓

Passed all tests! ✓

Question 7

Correct

Marked out of 5.00

Write a predicate `asbs(?List)` that holds if `List` is a list that starts with atom `a` repeated zero or more times followed by atom `b` repeated zero or more times.

For example:

Test	Result
`test_answer :- asbs([a,a,a,b]), writeln('OK').`	OK
`test_answer :- \+ asbs([a,b,a]), writeln('OK').`	OK

Answer: (penalty regime: 0, 20, ... %)

```
bs([b]).
bs([b | L]) :- bs(L).

asbs([]).
asbs([a]).
asbs([b]).
asbs(L) :- bs(L).
asbs([a|L]) :- asbs(L) ; bs(L).
```

	Test	Expected	Got	
✓	`test_answer :- asbs([a,a,a,b]), writeln('OK').`	OK	OK	✓
✓	`test_answer :- \+ asbs([a,b,a]), writeln('OK').`	OK	OK	✓

Passed all tests! ✓

Question 8

Incorrect

Marked out of 2.00

An instance of CSP can be transformed to an instance of an optimisation problem. Given an assignment, determine which of the following is a suitable objective function.

- a. number of variables that are False in the assignment
- b. number of variables in the assignment
- c. number of constraints in the CSP
- d. number of constraints that return False
- e. number of variables in the CSP that have False in their domains
- [✓ f. number of arcs in the constraint network] ✗
- g. number of variables in the CSP that have True in their domains
- h. number of variables that are True in the assignment

Question 9

Partially correct

Marked out of 6.00

Consider an instance of the CSP problem and its corresponding constraint network with four variables: *a*, *b*, *c*, and *d*. The domain of all the variables are *{1, 2, 3, 4}*. We have the following two constraints:

- $b < c$
- $b * c = d$

Answer the following questions with integers using only numerals. Do not use words or expressions.

1. How many arcs does the network have? 5 ✓
2. Suppose we apply the *General Arc Consistency* algorithm to this network. After the network has been made arc-consistent, what is the <u>size</u> (number of elements) of the domain of the following variables?
   - *a* 4 ✓
   - *b* 2 ✓
   - *c* 3 ✓
   - *d* 3 ✓
3. Suppose during the execution of the *General Arc Consistency* algorithm, when an arc is being made consistent, the domain of its variable is changed. What is the maximum number of arcs that will be added to the to-do list again? 4 ✗ [If the number varies depending on which arc has been made consistent, write the largest number here.]

Information

Another way of representing a CSP is by a collection of *relations*. There will be one relation per constraint in the problem. A relation is basically a table that holds zero or more rows. The columns of the table are the variables that are in the scope of the constraint. See the documentation of the `Relation` class in the `csp module.`

For example, the following instance of CSP

```
CSP(
   var_domains = {var:{0,1,2} for var in 'ab'},
   constraints = {
      lambda a, b:  a > b,
      lambda b: b > 0,
   })
```

can be represented in relational form as the following:

```
[
    Relation(header=['a', 'b'],
             tuples={(2, 0),
                     (1, 0),
                     (2, 1)}),

    Relation(header=['b'],
             tuples={(2,),
                     (1,)})
]
```

Notes

- Since the variable names and the effect of their domains appear in the relations, there is no need to specify them separately.
- Here we are using a list for the collection of relations instead of a set to avoid the the need for creating hashable objects (required for Python sets). From an algorithmic perspective, the order of relation objects in the collection does not matter, therefore, a set would have been a more natural choice.
- Single (one-element) tuples in Python require an extra comma at the end, for example, `(2,)`.
- You can write a short function that takes a CSP object and returns a collection of Relations equivalent to the object.

Question 10

Partially correct

Marked out of 6.00

Convert the following instance of CSP to an equivalent list of relations called `relations`. Then use the *variable elimination* algorithm to eliminate variable ***b*** and produce a new list of relations called `relations_after_elimination`.

```
csp = CSP(
   var_domains = {var:{1,2,3,4} for var in 'abc'},
   constraints = {
      lambda a, b, c: a < b < c,
      lambda b: b % 2 == 0,
      }
    )
```

Further notes and instructions

- In each relation, the variables must appear in alphabetic order.
- The order of relations in the two lists `relations` and `relations_after_elimination` is not important.
- This question does not have a check button. The precheck button runs a few tests with no penalty. Marking will be done after the exam. Partial marks will be awarded for test cases that pass.

Answer: (penalty regime: 0 %)

Reset answer

```
from csp import Relation

relations = [

    Relation(header=['a', 'b', 'c'],
             tuples={(1, 2, 3),
                     (1, 2, 4),
                     (1, 3, 4),
                     (2, 3, 4)
             }),

    Relation(header=['b'],
             tuples={(2,),
                     (4,)})

      ]

relations_after_elimination = [

    Relation(header=['a', 'b', 'c'],
             tuples={(1, 2, 3),
                     (1, 2, 4),
             }),

    Relation(header=['b'],
             tuples={(2,),
                     (4,)})

    ]

```

	Test	Expected	Got	

✗	`from student_answer import relations_after_elimination`
	`from csp import Relation`
	
	`print(len(relations_after_elimination))`
		1	2	✗

Some hidden test cases failed, too.

Show differences

Question 11

Not answered

Marked out of 4.00

This question is about the algorithm `greedy_descent(initial_state, neighbours, cost)` as it was described in one of the weekly quizzes.

▶ If you need to see the description of the algorithm (copied from the related question in the weekly quiz), click here to expand/collapse.
Assuming that we use this function to solve CSP problems and the state is an assignment and the cost function returns the number of unsatisfied constraints, determine the correctness of the following statements.

- If the function `neighbours` returns the current state itself, the algorithm can get stuck in an infinite loop. ✖
- If in a number of iterations, a sequence of values returned by the function `neighbours` form a cycle (from the current state to itself), the algorithm can get stuck in an infinite loop. ✖
- The algorithm always halts. ✖
- In all problems, all the neighbours of the current state are guaranteed to have a cost different from the cost of the current state. ✖

Question 12

Not answered

Marked out of 7.00

Write a function `num_crossovers(parent_expression1, parent_expression2)` that takes two expression trees as parents and returns the number of ways in which the two parent trees can be crossed over to produce two children.

The input parents have the same representation as what was used in the second assignment super-quiz. If you want to see the representation specification, click on the link at the end of this text. The return value must be an integer. You must compute the number of potential crossovers by considering every node in each tree as a potential crossover point. Note that the set of leaf symbols or the set of function symbols are not provided as they are not necessary in computing the required number.

▶ If you want to see the details of the representation, click here to expand/collapse.

For example:

Test	Result

`from student_answer import num_crossovers`
 	
`expression1 = ['+', 12, 'x']`
`expression2 = ['-', 3, 6]`
`print(num_crossovers(expression1, expression2))`
	9

Answer: (penalty regime: 0, 20, ... %)

```

```

Information

Representation of belief networks in Python

A belief (or Bayesian) network is represented by a dictionary. The keys are the names of variables. The values are dictionaries themselves. The second level dictionaries have two keys: `'Parents'` whose value is a list of the names of the variables that are the parents of the current variable, and `'CPT'` whose value is a dictionary again. The keys of the third level dictionaries are tuples of Booleans which correspond to possible assignments of values to the parents of the current node (in the order they are listed) and the values are real numbers representing the probability of the current node being true given the specified assignment to the parents.

Notes

- Variable names are case sensitive.
- If a node does not have any parents, the value of `'Parents'` must be an empty list and the only key of the third level dictionary is the empty tuple.
- For simplicity, we assume that all the variables are Boolean.

Example

The following is the representation of the *alarm network* presented in the lecture notes.

```
network = {
    'Burglary': {
        'Parents': [],
        'CPT': {
            (): 0.001,
         }
    },

    'Earthquake': {
        'Parents': [],
        'CPT': {
            (): 0.002,
        }
    },

    'Alarm': {
        'Parents': ['Burglary','Earthquake'],
        'CPT': {
            (True,True): 0.95,
            (True,False): 0.94,
            (False,True): 0.29,
            (False,False): 0.001,
        }
    },

    'John': {
        'Parents': ['Alarm'],
        'CPT': {
            (True,): 0.9,
            (False,): 0.05,
        }
    },

    'Mary': {
        'Parents': ['Alarm'],
        'CPT': {
            (True,): 0.7,
            (False,): 0.01,
        }
    },
}
```

Question 13

Not answered

Marked out of 6.00

Create a belief network called `network` with five random variables A, B, C, D, and E with the property that E and A both influence D; and they are both influenced by B. The value of the probabilities do not matter as long as they are valid and do not nullify influences.

For example:

Test	Result

```
from student_answer import network
from numbers import Number

# Checking the overall type-correctness of the network
# without checking anything question-specific

assert type(network) is dict
for node_name, node_info in network.items():
    assert type(node_name) is str
    assert type(node_info) is dict
    assert set(node_info.keys()) == {'Parents', 'CPT'}
    assert type(node_info['Parents']) is list
    assert all(type(s) is str for s in node_info['Parents'])
    for assignment, prob in node_info['CPT'].items():
        assert type(assignment) is tuple
        assert isinstance(prob, Number)

print("OK")
```
	`OK`

Answer: (penalty regime: 25, 50, ... %)

Question 14

Not answered

Marked out of 5.00

In the context of the *Inference by Enumeration* algorithm, determine the correctness of the following statements.

1. For a given number of variables, the topology of the belief network (the arcs) have an effect on which variables are hidden. ✗
2. The evidence variables given to the algorithm have an effect on which variables are hidden. ✗
3. The larger the number of evidence variables, the shorter the computation time. ✗
4. Before normalisation, the probabilities in the "raw" distribution can be negative. ✗
5. The time complexity of the algorithm grows exponentially with respect to the number of hidden variables. ✗

Question 15

Not answered

Marked out of 5.00

Suppose we want to predict the value of variable `Y` based on the values of variables `X1`, `X2`, `X3`. Assuming that we want to use a Naive Bayes model for this purpose, create a belief net for the model called `network`. The probabilities must be learnt by using the dataset given below. Use Laplacian smoothing with a pseudocount of 1.

X1	X2	X3	Y
T	F	F	F
F	F	F	F
T	F	F	F
F	F	F	F
F	F	F	T
F	F	F	T
T	T	F	T

Notes

- Node names are case sensitive.
- Since we are using Python syntax, you can use fraction expressions if you wish.
- This question does not have a check button. The precheck button runs some tests with no penalty. Marking will be done after the exam. Partial marks will be awarded for test cases that pass.

Answer: (penalty regime: 20, 40, ... %)

Question 16

Not answered

Marked out of 4.00

Suppose we use the k-Nearest Neighbours (kNN) algorithm with the following examples:

```
(-2, +), (-1, +), (0, -), (1, +), (2, -), (3, -)
```

Each example is a pair where the first element is of type input (which is a single real number) and the second element is of type output (a label which is either `+` or `-`).

Some details of the k-Nearest Neighbours (kNN) algorithm are as follows:

- the Euclidean distance is used for measuring the distance between two points;
- if after selecting k nearest neighbours, the distance to the farthest selected neighbour and the distance to the nearest unselected neighbour are the same, more neighbours are selected until these two distances become different or all the examples are selected;
- *majority voting* is used for combining outputs; that is, the label that is most common among the neighbours will be returned. If there is a tie, either $+$ or $-$ will be randomly returned.

Answer the following questions by completing the sentences.

1. When $k=3$, for input `1.2` the prediction is ✖ .
2. For input `0.8` the smallest value of k for which there will be a tie is ✖ .
3. For some values of k, a decision boundary is formed at point `0.5` so that every input greater than `0.5` is classified as `-` and every input less than `0.5` is classified as `+`. The smallest such value is ✖ and the largest such value is ✖ .

Question 17

Not answered

Marked out of 4.00

In the context of Artificial Neural Networks, complete or determine the correctness of the following statements.

1. Bias is equivalent to a weight for an input that is always 1. ✖
2. When using a perceptron to learn a classifier for points on a 2D plane, the number of weight parameters required depends on the number of points in the training data. ✖
3. The error of a perceptron on the training data can increase after a weight update. ✖
4. The main motivation for having a hidden layer is to ✖

Question 18

Not answered

Marked out of 7.00

Write a functions `num_parameters(unit_counts)` that takes a list of integers representing the number of units (nodes) in the layers of an MLP network and returns the total number of parameters (weights and biases together).

The input is guaranteed to have one or more elements. All the elements are integers and are greater than 0. The first element is the number of units (nodes) in the input layer, the second number (if present) is the number of nodes in the second layer, and so on. The last element is the number of nodes in the output layer.

For example:

Test	Result
`print(num_parameters([2, 4, 2]))`	`22`

Answer: (penalty regime: 0, 20, ... %)

```
1 
```

Question 19

Not answered

Marked out of 3.00

Consider the following explicit game tree.

```
[[3, -2], [2, 1, 4]]
```

Answer the following questions with integers. Use numerical digits (not words or expressions).

- If the first player is Max, what is the utility of the root node? ✗
- If the first player is Min, what is the utility of the root node? ✗
- How many actions are available to the first player? ✗

Question 20

Incorrect

Marked out of 6.00

Consider the following explicit game tree.

```
[[-1, [8 , 8], 15], [3, -3, 2]]
```

Assuming that the player at the root of the tree is Max, prune the tree (if necessary). Provide two variables: `pruned_tree` which is the pruned game tree and `pruning_events` which is a list of pairs of alpha and beta, one for each time a pruning event was triggered.

.

For example:

Test	Result

`import student_answer`
 	
`check_it_is_a_gametree(student_answer.pruned_tree)`
`check_it_is_a_pruning_events_list(student_answer.pruning_events)`
	OK
	OK

Answer: (penalty regime: 25, 50 %)

Reset answer

```
from math import inf

pruned_tree = # COMPLETE


pruning_events = [
    # (alpha, beta),
    ]
```

	Test	Expected	Got	

✗	`import student_answer`
	
	`check_it_is_a_gametree(student_answer.pruned_tree)`
	`check_it_is_a_pruning_events_list(student_answer.pruning_events)`
		OK
		OK
			`***Run error***`
			`Traceback (most recent call last):`
			`  File "__tester__.python3", line 153, in <module>`
			`    import student_answer`
			`  File "student_answer.py", line 3`
			`    pruned_tree = # COMPLETE`
			`                  ^^^^^^^^^^`
			`SyntaxError: invalid syntax`
				✗

Testing was aborted due to error.

Show differences

Question 21

Not answered

Not graded

This is not an actual exam question and carries no marks. You can ignore any warning that indicates this "question" is incomplete; you can safely leave this empty. However, if you have any comments that the examiners must be aware of, use the text area below.