Project 2 - Functional Dependencies and Normalisation

INSTRUCTIONS

1. This project is an individual project.
2. Submit only one ZIP file, `<student number>.zip` (for example: "`A012345L.zip`"), containing the completed Python file "`project2.py`" (follow the template indicated in the file and feel free to add other necessary functions) to the "`Project 2: Submission`" folder under Luminus

   "`Files > Project 2: Functional Dependencies and Normalisation`" by **Friday 28 February 2020, 18:30**.
3. Past this deadline and before **Friday 6 March 2020, 18:30**, you may submit to the "`Project 2: Late Submission`" folder (penalties apply).
4. If your submission file is not properly named, it might not be marked.
5. You may be asked to demonstrate the code that you submit. Make sure that your code is readable and commented properly. Highlight the major algorithmic design choices in your comments.
6. Do not copy any code or part of the code from a third party (e.g. friend, internet source, book) as it is plagiarism. If you borrow ideas from a source, add the necessary references in the comments of your code.

In this project, we use Python 3.8. Use the Python file "`project2.py`" from Luminus files to answer the questions. You may add additional functions to help you answer the questions. However, do not change the name and the parameters of the given functions in the template file. You may import additional Python standard libraries (e.g `itertools`, `copy`) but not external libraries.

For all questions, we use the following representations. The schema is represented as a Python list of non-empty strings. For example, the schema $\{A, B, C, D\}$ is represented as `['A', 'B', 'C', 'D']`. Even though it is represented as a list, the order of the elements in the list does not matter and it must have unique elements. A functional dependency is represented as a list of two lists. The order of the two elements matters. For example, the functional dependency $\{AB\} \rightarrow \{C\}$ is represented as `[['A', 'B'], ['C']]`. A set of functional dependencies is a list of functional dependencies. The order of the functional dependencies does not matters.

You may assume that the input to the questions is always valid (e.g. no empty string, no incomplete functional dependency). You do not need to validate the input. The code should be readable and commented properly.

Question 1 [3 marks]

(a) Write a function `closure` that takes three parameters: a schema, a set of functional dependencies and a set of attributes, which is the subset of the schema, and returns the closure of the set of attributes.

For example, the attribute closure of $\{A, B\}$, $\{A, B\}^+ = \{A, B, C, D\}$,

```
closure(['A', 'B', 'C', 'D'], [[['A', 'B'], ['C']], [['C'], ['D']]], ['A', 'B'])
      = ['A', 'B', 'C', 'D']
```

(b) Write a function `all_closures` that takes two parameters, a schema and a set of functional dependencies, and returns the closure of all subsets of attributes excluding super keys that are not candidate keys. The results is represented as a list of functional dependencies of the form $S \rightarrow S^+$ where S is a subset of attributes.

For example,

```
all_closures(['A', 'B', 'C', 'D'], [[['A', 'B'], ['C']], [['C'], ['D']]])
       = [[['A'], ['A']], [['B'], ['B']], [['C'], ['C', 'D']], [['D'], ['D']],
       [['A', 'B'], ['A', 'B', 'C', 'D']], [['A', 'C'], ['A', 'C', 'D']],
       [['A', 'D'], ['A', 'D']], [['B', 'C'], ['B', 'C', 'D']],
       [['B', 'D'], ['B', 'D']], [['C', 'D'], ['C', 'D']], [['A', 'C', 'D'],
       ['A', 'C', 'D']], [['B', 'C', 'D'], ['B', 'C', 'D']]]
```

Question 2 [7 marks]

(a) Write a function `min_cover` that takes two parameters, a schema and a set of functional dependencies, and returns a minimal cover of the functional dependencies. The results is represented as a list of functional dependencies.

For example,

```
min_cover(['A', 'B', 'C', 'D', 'E', 'F'],
          [[['A'], ['B', 'C']],[['B'], ['C','D']], [['D'], ['B']],
           [['A','B','E'], ['F']]])
      = [[['A'], ['B']], [['B'], ['C']], [['B'], ['D']],
         [['D'], ['B']], [['A', 'E'], ['F']]]
```

There might be more than one correct answer to this question.

(b) Write a function `min_covers` that takes two parameters, a schema and a set of functional dependencies, and returns all minimal covers reachable from the set of functional dependencies. The results is represented as a list of minimal covers. Make sure that you clearly explained the rationale of the algorithm for this question in the indicated comment section in the code.

For example,

```
min_covers(['A', 'B', 'C'], [[['A'], ['B']],[['B'], ['C']], [['C'], ['A']]])
      = [[[['A'], ['B']], [['B'], ['C']], [['C'], ['A']]]]
```

For the case above, we find only one minimal cover.

(c) Write a function `all_min_covers` that takes two parameters, a schema and a set of functional dependencies, and returns all minimal covers. The results is represented as a list of minimal covers. Make sure that you clearly explained the rationale of the algorithm for this question in the indicated comment section in the code.

For example,

```
min_covers(['A', 'B', 'C'], [[['A'], ['B']],[['B'], ['C']], [['C'], ['A']]])
= [
[[['A'], ['C']], [['B'], ['A']], [['C'], ['A']], [['A'], ['B']]],
[[['B'], ['C']], [['C'], ['A']], [['A'], ['B']]],
[[['C'], ['B']], [['A'], ['C']], [['C'], ['A']], [['B'], ['C']]],
[[['C'], ['B']], [['A'], ['C']], [['B'], ['A']]],
[[['B'], ['C']], [['A'], ['B']], [['B'], ['A']], [['C'], ['B']]]
]
```

For the case above, we find five minimal covers.

– END OF PAPER –