< Programming Assignment #1 >

Submission: **LMS** (learning.hanyang.ac.kr)

- **Until April 4 (Thursday), 23:59 PM.**

1. Environment

- OS: Windows, Mac OS, or Linux
- Languages: Python (any version is ok)

2. Goal: to find association rules using the **Apriori** algorithm

- step 1: find frequent itemsets using Apriori
- step 2: for each frequent itemset, find association rules

3. Requirements

The program must meet the following requirements:

- File name: studentID_name_hw1**.py** (**not ipynb!)**
  - You can put your name in either Korean or English. For english name students, do NOT put space between your first and last names. And only use plain alphabet, do not use like ñ
  - Example 1: **2008011666_채동규_hw1.py**
  - Example 2: **2025016242_albertkarlo_hw1.py**
- Python version: any version is fine, but **python3** is recommended.
- I will execute your code with **three** arguments: **minimum support (%)**, **input file name**, **output file name**
  - Example: apriori.py 10 input.txt output.txt
    - This means that **minimum support = 10%**, input file name = '**input.txt**', output file name = '**output.txt**'
    - I will test your assignment using several different minimum support values.

- Input file format (.txt)

  [*item_id*]\t[*item_id*]\n

  [*item_id*]\t[*item_id*]\t[*item_id*]\t[*item_id*]\t[*item_id*]\n

  [*item_id*]\t[*item_id*]\t[*item_id*]\t[*item_id*]\n

  - Each line: transaction
  - *item_id* is a numerical value
  - There is no duplication of items in each transaction
  - Example:

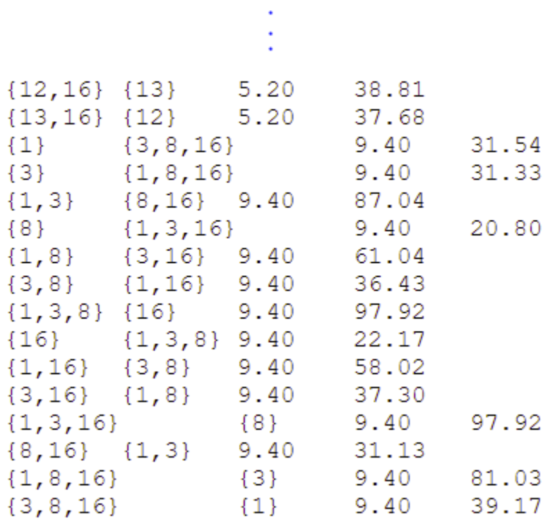
```
18      2       4       5       1
1       11      15      2       7       16
2       1       16
15      7       6       11      18      9
11      2       13      4
```

- Output file format (.txt)

  {*item_set*}\t{*associative_item_set*}\t*support*(%)\t*confidence*(%)\n

  {*item_set*}\t{*associative_item_set*}\t*support*(%)\t*confidence*(%)\n

  - {*item_set*}\t{*associative_item_set*}: association rules with minimum support (Minimum confidence is **not** considered. Only consider minimum support)
    - {*item_set*} → {*associative_item_set*}
    - Use **braces** ({   }) to represent item sets: {*item_id*, *item_id*, …} (*Important!!*)
  - *Support*: probability that a transaction contains {*item_set*} ∪ {*associative_item_set*}
  - *Confidence*: conditional probability that a transaction having {*item_set*} also contains {*associative_item_set*}
  - The order of lines in the output file is not important.
  - The value of support and confidence should be rounded to **two decimal places**.
    - e.g., 24.631 should become 24.63.
  - Result example:



```
                        ⋮
{12,16} {13}    5.20    38.81
{13,16} {12}    5.20    37.68
{1}     {3,8,16}        9.40    31.54
{3}     {1,8,16}        9.40    31.33
{1,3}   {8,16}  9.40    87.04
{8}     {1,3,16}        9.40    20.80
{1,8}   {3,16}  9.40    61.04
{3,8}   {1,16}  9.40    36.43
{1,3,8} {16}    9.40    97.92
{16}    {1,3,8} 9.40    22.17
{1,16}  {3,8}   9.40    58.02
{3,16}  {1,8}   9.40    37.30
{1,3,16}        {8}     9.40    97.92
{8,16}  {1,3}   9.40    31.13
{1,8,16}        {3}     9.40    81.03
{3,8,16}        {1}     9.40    39.17
```

  - Note: Please make sure to match the output format! If the format is not correct, you can't get any score.

4. Submission

- Please make **a single, runnable .py file** and upload it on LMS.

5. Penalty

- Late submission
  - 1 week delay: 20%
  - 2 weeks delay: 50%
  - Delay more than 2 weeks: 100%
- Requirements unsatisfied
  - Penalty up to 100% will be given depending on how the requirements are well-satisfied
- Plagiarim (from Internet, assisted by ChatGPT or any other AI-based generation, borrowed from someone else, etc)
  - Such plagiarim can be easily detected, and if detected, F grade will be given.
  - Please do it your own. This is not a difficult assignment.