National University of Singapore
School of Computing
CS1010X: Programming Methodology
Semester II, 2024/2025

Mission 9 - Side Quest
Funky Merge Spell

Release date: 17 March 2025
Due: 24 April 2025, 23:59

Required Files

- sidequest09.1-template.py
- sidequest09.1.zip

Background

It is a breezy evening. Satisfied that everything in your home is well-organised thanks to your newfound sorting abilities, a sense of wanderlust begins to tug at you, its faint whisper gently draws you outdoors. Before long, your feet settle into a steady cadence. Portal after portal you cross, the landscape is becoming more and more unfamiliar as you whiz across the vast distance.

As you are near another portal, you notice that it has a different structure than the usual. The dirt path leading to it appears to be peculiarly well-trodden given its obscure location. Gazing intently at the portal, your consciousness seems to enlarge and you start to gain an ineffable comprehension. You see paths. A network of paths, crossing and diverging. These are the paths embarked on by your fellow mages through their training and through life. Your consciousness enlarges further. Now you see yourself standing before the portal, and queer enough, your fellow mages are also standing before that same portal simultaneously, yet each one alone — as if in different dimensions of space.

It is too much to take. You look away and try to shake off the eerie and surreal feeling. Yet somehow, you know it is all profoundly true: that in all improbability, your paths are to *merge* at PIM, according some ordained order. But how? And why? You have much to ponder about...

Administrivia

The template file `sidequest09.1-template.py` has been provided for you to write your answers in. It also includes sample input and output. In addition to these samples, you should test your functions with your own test cases.

Require data files are found in `sidequest09.1.zip`.

In this mission, you will implement a variant of **merge sort** shown in the lecture. So as not to undermine your learning, you are barred from using Python built-in sorting functions like `sorted` and `.sort` in this mission.

This mission consists of **two** tasks.

About the Data

The data for this task comes in the form of JSON[1] strings which contain NUS course listings. Each listing contains three fields: `<module code>`, `<module name>`, and `<prof name>`. A function **`read_json(filename)`** has been provided for you to convert the JSON string to a Python list. (See Page 5 for an example!)

Note: A *module* in this mission generally refers to NUS courses, not Python modules.

Task 1: Merging Lists (6 marks)

So far, you have managed to merge two sorted lists together into a single sorted list. Why not we try to do the same for more than two lists?

(a) Given a list of n sorted lists, your task is to define a function **`merge_lists(lists)`** that will merge each list in `lists` into a single sorted list. **(4 marks)**

Example:

```
>>> all_lst = [[2, 7, 10], [0, 4, 6], [3, 11]]
>>> print(merge_lists(all_lst))
[0, 2, 3, 4, 6, 7, 10, 11]
```

Hint: You should try to imitate and generalise the method that you have seen in the lecture, which merges $n = 2$ lists. The method takes advantage of the fact that each of the n lists are sorted, which means we can find the smallest element of all the lists by scanning the first element of each of the n lists.

(b) By modifying `merge_lists` or otherwise, write a function **`merge(lists, field)`**. **(2 marks)**

Instead of taking as input a list of lists of numbers, `merge` takes in a list of lists of course listings and merges the inner lists to return a single list of course listings that is sorted according to one of the three categories (module code, module name or prof name).

It is assumed that each inner list of courses is already sorted according to the required category, which is specified by `field`. The argument `field` should take as input one of the following accessor methods that are provided in the template: `module_code`, `module_name` or `module_prof`. A sample execution is provided for you in page 5.

[1] More details will be provided on this format in a later mission. For now, it suffices to know how to convert JSON strings to lists so that we can use them. Do your own research if you wish to find out more!

Task 2: Merge Sort (4 marks)

Your second task is to define the function **`merge_sort(lst, k, field)`**.

The function `merge_sort(lst, k, field)` sorts the list of modules `lst` according to one of the three categories specified via the input `field` and returns the sorted list. Instead of dividing the data into two parts in each iteration of the sort as you have seen in the lecture, your version of merge sort should divide the data in `k` parts, where `k` is an input to the function.

You are allowed to write additional helper functions.

A small sample dataset has been provided in **`modules_small.txt`** to aid you with your debugging. Your code will also be tested on a larger dataset **`modules.txt`** which contains many more module listings.

Hints: You should be using the function `merge` from Task 1 to merge the k lists into one, after you have split them and recursively sorted them (using `merge_sort`). Your final answer should bear similarity to the code shown in the lecture.

Challenge (Not Graded)

What is the time complexity of your k-way merge sort algorithm? Is it faster than the one you have seen in lecture (when $k > 2$)?

Hint: The time complexity of the algorithm depends on both the size of the input list and on k.

Merge Sort Illustration

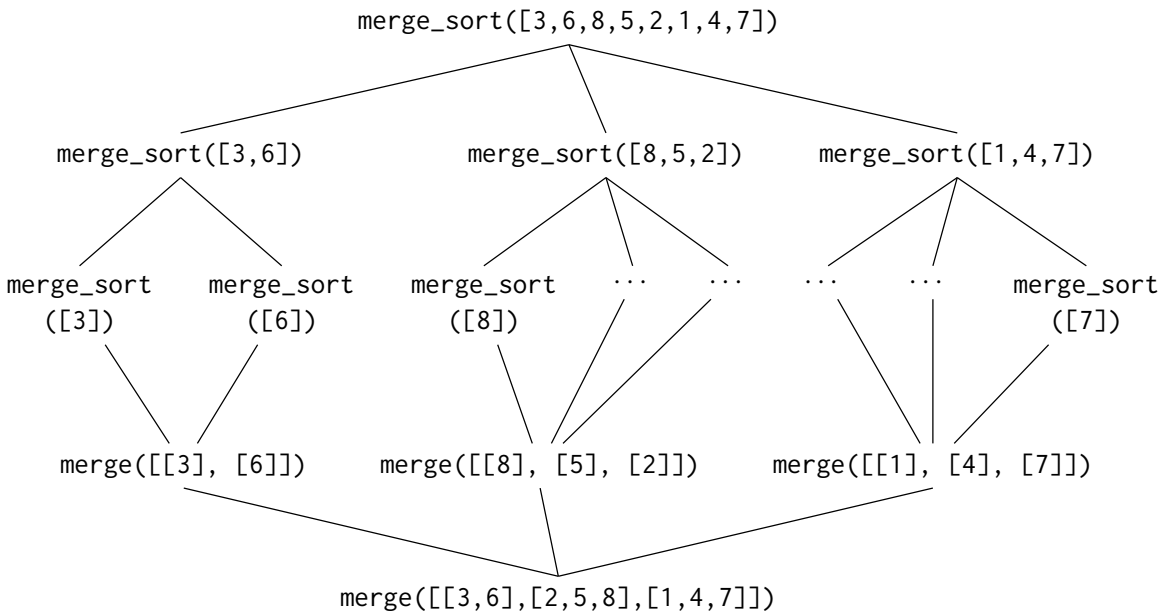
The original merge sort splits and merge a list in the following sequence:

```
[3,6,8,5,2,1,4,7]
[3,6,8,5],    [2,1,4,7]            #-> Splits into 'left' and 'right' lists
[3,6],[8,5],  [2,1],[4,7]          #-> Each list is split into two again
[3],[6], [8],[5], [2],[1], [4],[7] #-> Till they are individual elements
[3,6],[5,8],  [1,2],[4,7]          #-> Then the pairs of lists get merged
[3,5,6,8],    [1,2,4,7]
[1,2,3,4,5,6,7,8]
```

The merge sort with $k = 3$ splits and merge a list in the following sequence:

```
[3,6,8,5,2,1,4,7]
[3,6],[8,5,2],[1,4,7]      #-> Splits into three (not two) smaller lists
[3],[6],[8],[5],[2],[1],[4],[7] #-> Till they become individual elements
[3,6],[2,5,8],[1,4,7]      #-> Each smaller list eventually gets sorted
[1,2,3,4,5,6,7,8]          #-> The three lists are finally merged
```

To illustrate further, the following is a call tree showing the order that `merge_sort` and `merge` are called for $k = 3$. Note that some arguments have been supressed for readability.



Example of the data and outputs

Suppose that **`file.txt`** contains the JSON string:

```
[["CS3216", "SOFTWARE DEVELOPMENT ON EVOLVING PLATFORMS", "TAN KENG YAN, COLIN"],
["CS2010", "DATA STRUCTURES & ALGORITHMS II", "STEVEN HALIM"],
["CS1010S", "PROGRAMMING METHODOLOGY", "LEONG WING LUP, BEN"]]
```

then **`read_json(file.txt)`** will return the python list, which we shall call **`lst`**:

```
[
["CS3216", "SOFTWARE DEVELOPMENT ON EVOLVING PLATFORMS", "TAN KENG YAN, COLIN"],
["CS2010", "DATA STRUCTURES & ALGORITHMS II", "STEVEN HALIM"],
["CS1010S", "PROGRAMMING METHODOLOGY", "LEONG WING LUP, BEN"]
]
```

Result of **`merge_sort(lst, 2, module_code)`**:

```
[
["CS1010S", "PROGRAMMING METHODOLOGY", "LEONG WING LUP, BEN"],
["CS2010", "DATA STRUCTURES & ALGORITHMS II", "STEVEN HALIM"],
["CS3216", "SOFTWARE DEVELOPMENT ON EVOLVING PLATFORMS", "TAN KENG YAN, COLIN"]
]
```

Result of **`merge_sort(lst, 3, module_name)`**:

```
[
["CS2010", "DATA STRUCTURES & ALGORITHMS II", "STEVEN HALIM"],
["CS1010S", "PROGRAMMING METHODOLOGY", "LEONG WING LUP, BEN"],
["CS3216", "SOFTWARE DEVELOPMENT ON EVOLVING PLATFORMS", "TAN KENG YAN, COLIN"]
]
```

Result of **`merge_sort(lst, 10, module_prof)`**:

```
[
["CS1010S", "PROGRAMMING METHODOLOGY", "LEONG WING LUP, BEN"],
["CS2010", "DATA STRUCTURES & ALGORITHMS II", "STEVEN HALIM"],
["CS3216", "SOFTWARE DEVELOPMENT ON EVOLVING PLATFORMS", "TAN KENG YAN, COLIN"]
]
```

The following is an illustration of `merge`:

```
>>> sample_list_of_lists = \
[["CS1010S", "PROGRAMMING METHODOLOGY", "LEONG WING LUP, BEN"],
  ["CS3235", "COMPUTER SECURITY", "NORMAN HUGH ANDERSON"]],
 [["CS4221", "DATABASE DESIGN", "LING TOK WANG"],
  ["CS2010", "DATA STRUCTURES & ALGORITHMS II", "STEVEN HALIM"]]]

>>> merge(sample_list_of_lists, module_prof)

[['CS1010S', 'PROGRAMMING METHODOLOGY', 'LEONG WING LUP, BEN'],
 ['CS4221', 'DATABASE DESIGN', 'LING TOK WANG'],
 ['CS3235', 'COMPUTER SECURITY', 'NORMAN HUGH ANDERSON'],
 ['CS2010', 'DATA STRUCTURES & ALGORITHMS II', 'STEVEN HALIM']]
```

That is all for this task. Happy merging! (: