***Scheduling Shenanigans at The Vogon Poetry Corner***

Four steps to solve this problem.

1. Group all events information for each client.

Use ***function*** **info\_common\_client\_ids** (***sample\_clients, sample\_event\_occurrences***) to return an array of objects, each object contains client\_id, client\_name and events array, which contains event\_id, start and end information. For example:

Output:

[{client\_id: 313,

client\_name: “Prophet”,

events: [{event\_id: 1029,

start: "Mon, 29 May 2017 11:00:00 PDT -07:00",

end: "Mon, 29 May 2017 12:30:00 PDT -07:00"},

{event\_id: 923,

start: "Mon, 29 May 2017 12:00:00 PDT -07:00",

end: "Mon, 29 May 2017 13:00:00 PDT -07:00"},

{…},

…

{…}]},

{…},

{…},

…

{…}]

***Time complexity:*** O(m\*n). m is the length of ***sample\_clients***, n is the length of ***sample\_event\_occurrences.***

1. Find conflicting event pairs under same client.

In ***function*** **find\_rough\_Conflicts**(***events***), traverse event information and use

***function*** **intersection\_time**(***start1, end1, start2)*** to check if two events have time conflict, if so, add them to result array and return. For example:

Input: [{event\_id: 1029,

start: "Mon, 29 May 2017 11:00:00 PDT -07:00",

end: "Mon, 29 May 2017 12:30:00 PDT -07:00"},

{event\_id: 923,

start: "Mon, 29 May 2017 12:00:00 PDT -07:00",

end: "Mon, 29 May 2017 13:00:00 PDT -07:00"},

{…},

…

{…}];

Output: [[1029, 923], [923, 432], [1032, 1035], …]; *//[1029, 923] is conflicting event\_id pairs.*

***Time complexity:*** O(). n is the length of ***events.***

1. Merge sub-arrays if they contain same element.

***function*** **merge**(***arr***) returns a unique and largest set.

Input: [[1,2], [3,4], [1,3], [5,6], [6,5]];

Output: [[1,2,3,4], [5,6]];

***Time complexity:*** O(). n is the length of ***arr.***

1. ***function*** **findSchedulingConflicts**(***sample\_clients, sample\_event\_occurrences***)

Organize the output as required.