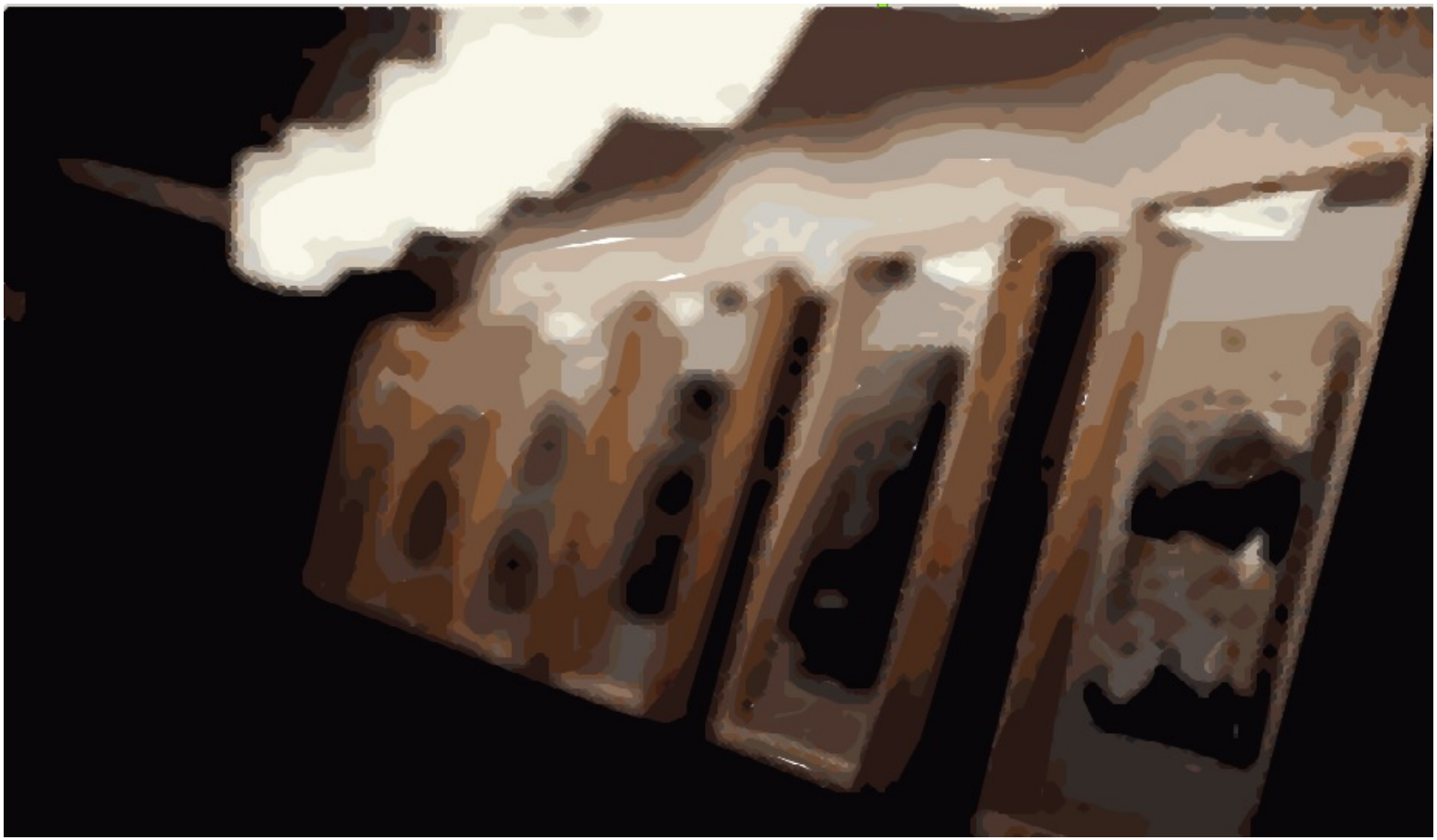
Python One Day Challenge (PODC)
The Optimized Exhibition Opening
— EPITA —

M. Angoustures & R. Dehak & R. Erra & A. Letois

April 2020



Your goal :

You are working in an art gallery and tonight, you have an exhibition opening. This one is very special, the attendees are robots and have very specific tastes. You are in charge of ordering the paintings in a very large corridor where the robots will look at them. To satisfy them, you have to follow some basic rules precised in the following slides.

Robots ?

Robots are integrating each painting with a simple list of tags. Each tag represents an element of the painting. For example, if a robot sees the painting *Mona Lisa* it will integrate it as a list of tags : [woman,smile].

Robots also separate two types of paintings :

- Landscape for an horizontal painting
- Portrait for a vertical painting

Landscape

Portrait

To better suit to robots preferences, paintings are placed behind a frameglass. Each frameglass contains **one** Landscape painting or **two** Portrait paintings.

Just like paintings, robots are integrating each frameglass with a list of tags. In a case of a Landscape, the list of tags is the same than the list of the painting tags. In a case of two Portraits, the list of tags of the frameglass is the list of all the tags present in any or both of the two paintings it contains. Each tag in a list is only counted **once** !

To help you with the ordering, you are given a file with all informations about the set of (different) paintings.

- An input data set is given as a plain text (only ASCII characters) ; all lines are terminated by a single '\n' character (UNIX- style line endings).
- The first line of the dataset contains one integer N ($1 \leq N \leq 10^5$) : the number of paintings for the exhibition.
- It is followed by N lines : line i contains a description of the painting with an ID i ($0 \leq i < N$). The description of a painting i contains the following data, separated by a single space :
  1. A single character 'L' if the painting is horizontal (Landscape), or 'P' if it is vertical (Portrait).
  2. An integer M_i ($1 \leq M_i \leq 100$) : the number of tags for this painting.
  3. M_i text strings : the tags for the painting i.

Each tag consists only of lowercase ASCII letters and digits, between 1 and 10 characters (max). The order of those tags is not important.

Example

animals, fear, war

smile, woman

woman, pearl

Input file	Description
4	The exhibition has 4 paintings
L 3 animals fear war	Painting 0 is a Landscape and has tags
P 2 smile woman	Painting 1 is a Portrait and has tags [sn
P 2 woman pearl	Painting 2 is a Portrait and has tags [wo

The good part with having robots attendees is that their satisfaction is very easy to determine : it is simply a *numerical value* : an integer !
We will call this value the score of the exhibition or more precisely : the **global robotic satisfaction**.
The score of the exhibition is based on how interesting the transitions between each pair of subsequent (neighboring) frameglass are. Robots like when the transitions have something in common to preserve continuity (the two frameglass should not be totally different), but they also want them to be different enough to stay interested. The similarity of two Portrait paintings on a single frameglass is not taken into account for the score. This means that in this case, two paintings can, but don't have to, have tags in common. For example, if a frameglass contains two Portraits with [smile, woman] and [woman, pearl], then the tags of this frameglass will be [smile, woman, pearl].

The local **robotic satisfaction**

For two subsequent frameglass F_i and F_{i+1}, the **local robotic satisfaction** is the minimum (the smallest number of the three) of the following three (3) integers :

1. The number of common tags between F_i and F_{i+1}
2. The number of tags in F_i but not in F_{i+1}
3. The number of tags in F_{i+1} but not in F_i.

The **global robotic satisfaction**

For a full set of ordered paintings the final score is the **global robotic satisfaction**, defined as the sum of all **local robotic satisfactions**. Your goal is to maximize it ! *Remark : the code of the computation of the* **global robotic satisfaction** *is given : score_checker.py*

The output file must start with a single integer F ($1 \leq F \leq N$) : the number of frameglass in the exhibition. This must be followed by F lines describing each frameglass. Each line should contain either :

- A single integer : ID of the single Landscape painting in the frameglass.
- Two integers separated by a single space : IDs of the two Portrait paintings in the frameglass (in any order).

Each painting can be used only one time or not at all.

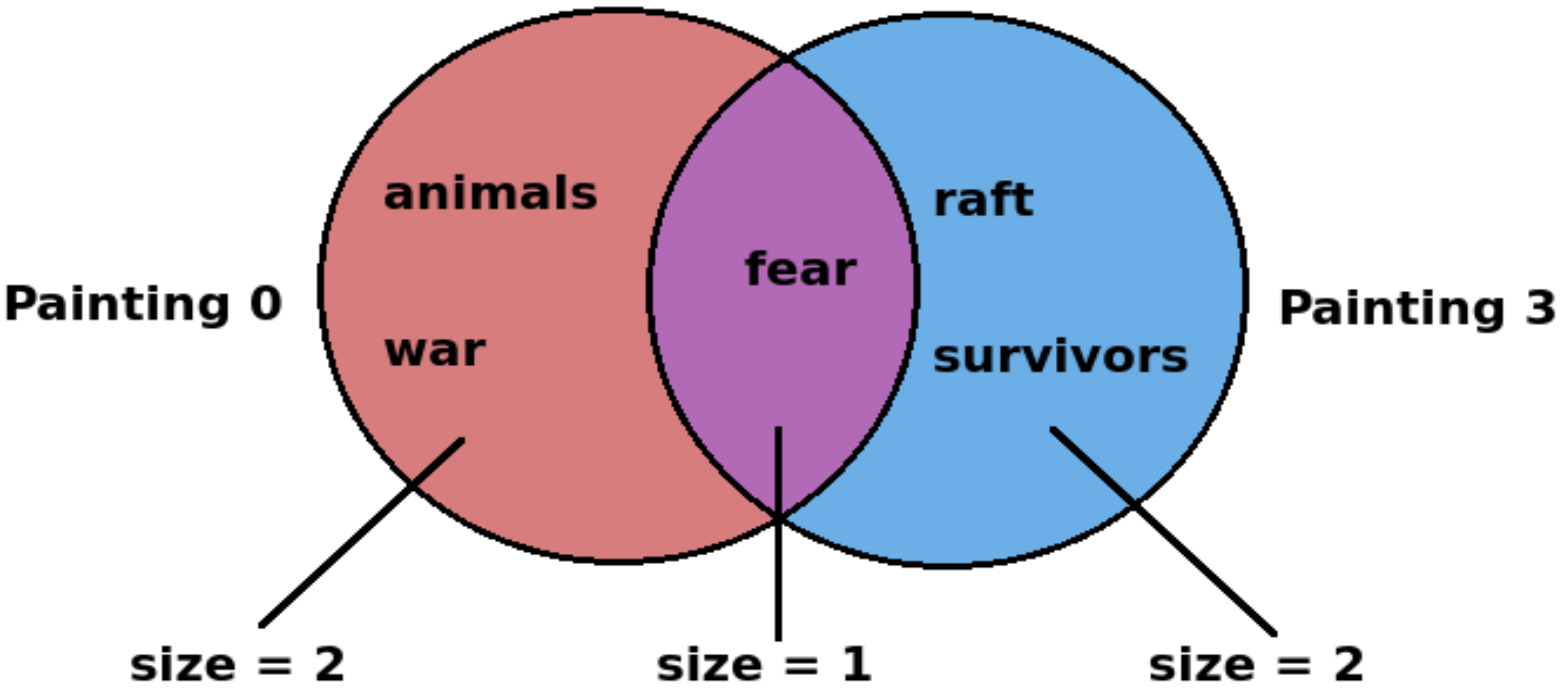
Example

frameglass F_0

frameglass F_1

frameglass F_2

Submission file	Description
3	The exhibition has 3 frameglass
0	First frameglass contains the painting 0
1 2	Second frameglass contains the paintings 1 and 2
3	Third frameglass contains the painting 3



Robotic satisfaction = min(2,1,2) = 1

The number of common tags is 1 -> [fear]
The number of tags in 0 but not in 3 is 2 -> [animals, war]
The number of tags in 3 but not in 0 is 2 -> [raft, survivors]

Some hints

1. The exhibition is tonight ! We need your help !
2. We don't have much time to think for the best ordering, so the computation need to be fast.
3. Begin with a simple solution, when it works . . .
4. . . . try to optimize it so it won't take too much time to execute, or it will be too late.
5. Use functions in your code !
6. Write comments if you want but be concise !

What we expect from you at the end

- 1 unique Python script that takes one input file and gives as output the adequate submission file.
- Try to Optimize the output files to get the biggest score.
- *Remark : to get the real "optimal" (maximal) score is probably an untractable problem, so think "approximation".*

Your final score for the "ranking"

1. We will compute the score for each of the 5 files
2. We will compute the sum of these 5 scores
3. This will be your final score for the "ranking"
4. Don't forget : think simple, begin with the "simplest" solution
5. Don't try to find first a too complex algorithm
6. Try to begin with a "very" simple solution
7. And only after try to find better strategies/tactics
8. Think "step by step", test different strategies/tactics

Where to sent your work

Sent to podcmsc@protonmail.com

1. Your python code please
2. **and** your Jupyter Notebook
3. and (mandatory) : add **all** your Family and First names both in your python code.
4. A unique email please !

Get the best score you can !

. . . **Congratulations and** *bon courage* **to all of you.**