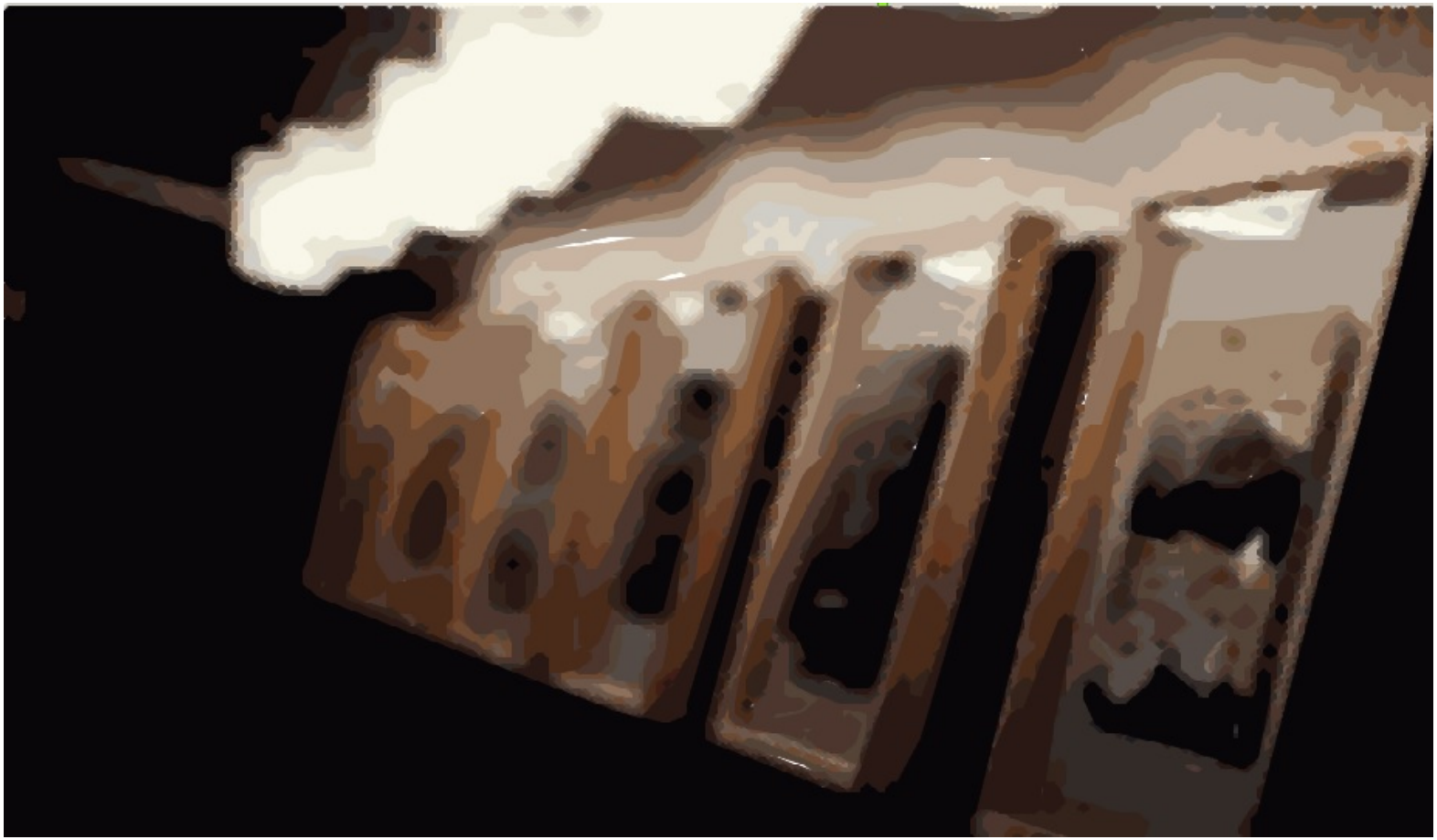
Python Two Days Challenge (P2DC)
The Optimized Exhibition Opening
— EPITA — MSc

M. ANGOUSTURES & R. DEHAK & R. ERRA & A. LETOIS

Mars 2022



Your Goal :

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

2 Problem description
- Robotic visualization
- Frames

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

2 Problem description
- Robotic visualization
- Frames

Frameglasses

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 information 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

Example

animals, fear, war

smile, woman

woman, pearl

fear, raft, survivors

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

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*

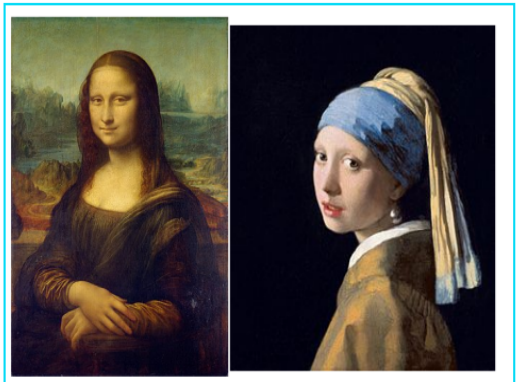
The output file must start with a single integer F $(1 \leq F \leq N)$: the number of frameglass in the exhibition. This must be followed by F lines describing each frameglass. Each line should contain either :

- 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

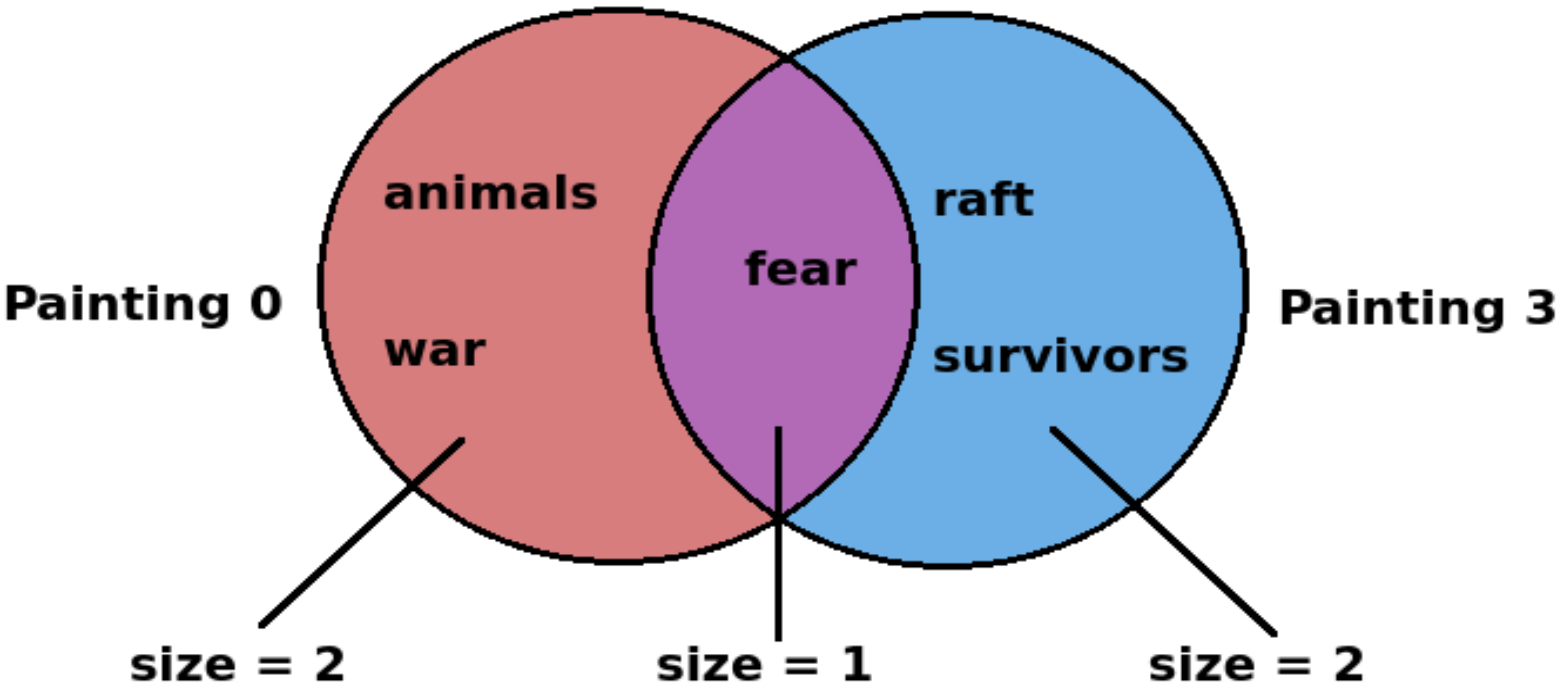
frameglass F_0



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]

Time and space constraints

Let us imagine you have a "classical" laptop : icore I7 with 8GB of RAM, then :

1. For an input file of one (1) MB the execution time has to be less than 1mn
2. For an input file of one (2) MB the execution time has to be less than 2mn
3. ...
4. For an input file of $n > 5$ MB the execution time has to be less than 10mn
5. All executions can be done with a 8GB RAM computer and **must be finished at max in an hour for all input file**.

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, once 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 ! (It is a challenge).
7. Use the **score checker file**, it will help you to evaluate quickly your strategies & tactics.
8. **FORGET THE BRUTE FORCE ATTACK !**

What we expect from you at the end

- 1 Python script, called **P2DC_Team_<your_team_number>.py** that takes one input file and gives as output the adequate submission file.
- 1 python script, called **P2DC_final_score_<your_team_number>.py** that will process all input files, easy to use, this script will also use the score-checker and will output the **Global Robotic Satisfaction**, *i.e.* the sum of the score for each input file.
- We will suppose that all input files will be in a directory called "Data".
- 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

Top score

1. You can access a site where you can upload your files and compare your scores with the other teams
2. The site adress is https ://pythonweek2022.fr
3. This website shows the different scores of each team for each uploaded file and the final score
4. The score is updated live
5. The maximum size of a file is 10Mo
6. You can upload one or multiple files at the same time
7. You can do only one upload each 5min

Presentation of your results

1. You will do a resume (at most one page A4, 11pt) in pdf, doc, docx, odt or txt that will describe your strategies and tactics, you can also use a jupyter notebook.
2. Your (main) python file will be named **P2DC_Team_<your team number>.py**
3. You will write a python file, named **P2DC_final_score_<your team number>.py** that will :
   - Run your main python file on each of the 5 given input files
   - Save all the outputs (on different files)
   - Print for each file the global score computed with the help of the scoring python file you have been given
   - Compute the sum of these 5 scores and print it, clearly.

Where to sent your work

An assignment will be created on Teams, you need to send your work there. You will create one zip file named Team_XX where XX is your team number. This zip file must contain :

1. Your **two** python codes
2. Your report file
3. **Mandatory** : add **all** your Family and First names as a commentary at the beginning of both your python code and your report file
4. **Mandatory** : add YOUR TEAM NUMBER in your python file name and in your report file name
5. **Only one** person in each group send the work for all the group

Get the best score you can !

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