VIETNAM NATIONAL UNIVERSITY
INTERNATIONAL UNIVERSITY

ꕥꕥꕥ ------ ꕥꕥꕥ

PROJECT

Minesweeper (Python)

Instructing Lecturer*: *Trần Thanh Tùng
Assessing Lecturer*: *Trần Thanh Tùng
***Course**: Data Structure & Algorithm (IT013IU)*

Repository Director: *Phạm Hoàng Minh* (ITITIU19031)
Colaborator #1: *Phạm Công Tuấn* (ITITIU19060)
Colaborator #2: *Trần Ngọc Tiến* (ITITIU19217)

Semester 02, 2020 – 2021

Table of Contents

I. Prologue

This game had been made for two purposes only: (1): To be the final project of the course Data Structure & Algorithm by using specific algorithms and data structure for implementation. (2): Supplementary Materials which can be put in CV to enhance competitivity among other candidates.

II. Abstract

Data Structure and Algorithm has been one of the critical classes that affects the level of any Computer Science or relevant IT students. Thus, since we have attended the course provided by International University in HCMC, we has built a small game serving as the final project for this course which was named "Minesweeper". This adaptive game was written in Python 3.7 along with some support from built-in modules (time, os, sys, logging, cProfile), Python-based packages (Numpy, memory_profiler), threading GUI (PyQt5) and distributed via BSD-3 Open-source License. Differed from many "Minesweeper" games online, we have implemented extra Undo/Redo Function via Stack, Graph Flowing which mathematical function for positional indexing. Moreover, an executable file is available for user's convenience.

III. Rules & Improvements

In this game, some features and design pattern has been added, designed, supplemented and consistently distributed via respective data structures and algorithms.

1) Features:

- *Graph Flowing*: This feature is executed when user clicks on an empty (zero-value) node. It would expand until the there are a boundary that blocks its expansion. This feature was built by the idea of Graph Theory Method (Depth First Search). By applying mathematical function inside, we accomplished of reducing memory size by N^2 in which N is the number of nodes in the matrix, and reducing the real-time complexity of using adjacency matrix from N^2 to N.
- *Flag & Question*: This function is a base core of "minesweeper": *Flag* in the node when you confirmed that there is a mine or *Question* if you find the node suspicious
- *Undo & Redo*: Player can undo or redo the step of previously clicked. However, the number of steps can undo or redo is limited (24 – 48), which is same as the Microsoft Office's Clipboard (24). Note that this feature is executed only if the player clicks on the bomb without the flag protection .
- *Time Counter*: We have also supplemented the Timing Counter which used *PyQt5* for display and Python "*time*" module for counting by real-time
- *Ranking*: We have also implemented Ranking Score to save every latest performance's record by using Pandas Library.
- *Solver Support*: We also build the alike-function that similar to minesweeper in Windows 7 to help player solve the matrix faster by click both left and right mouse at mean time to help the uncover matrix faster.

2) Design Pattern:

- *Singleton Pattern*: This pattern was implemented through out the project (all classes) to ensure that each object was run by its own and dependency state was transferred by message.
- *Observer Pattern*: Differed from normal situation when we need observer class to connect two object and transfer every messages, we used "Interface" as the Observer which binding the "minesweeper" logic

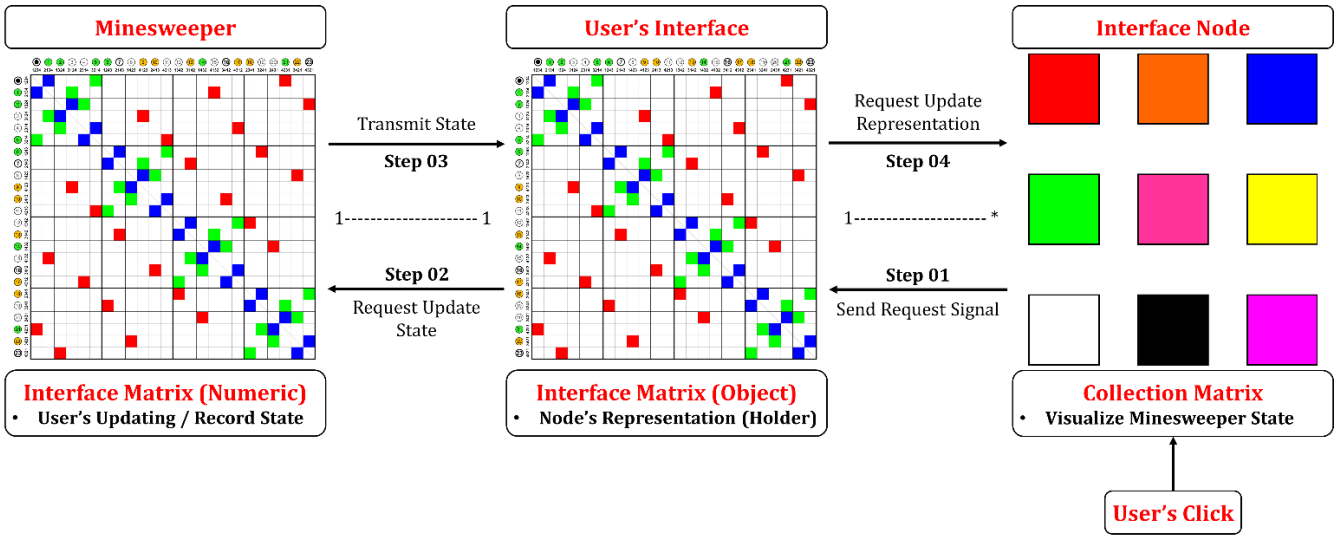

***Figure 1**: Representation of how objects (class) interact*

3) Data Structure:

- *HashTable (Python Dictionary)*: Store information that would be used at "*minesweeper*" class, "*node*" class, and probably "interface" class. The main use of this hashtable is to guarantee that all class having the same behaviour and consistency which would be easier for debug.
- *Matrix (Array)*: Storing objects or recorded states
- *Graph*: In fact, we don't actually build a graph representation due to memory inefficiency. In fact, a virtual graph was implemented based on the matrix indextation and mathematical function for conversion
- *Stack*: Record the previous state, which was used in the Undo or Redo features.
- *DataFrame*: Extension of 2D-matrix + 1D-array: Record the player's performance

4) Algorithm:

- *Binary Indexing (adaptive method of Binary Search)*: Store playing's record.
- *Depth First Flow (improved)*: Adaptive of Mathematical Function + Depth First Search + Graph Expansion. Time complexity: O(N)

5) Module / Class Explanation:

- *Interface*: Highest Interface to communicate.
- *Minesweeper*: Game Core that leverage Calculation & State Handling. Time complexity: O(1 – N)
- *Component_Interface*: Complicated class that cannot be instantiated
- *Config*: Main the project consistency for better calculation & Adaptation (Attached to Directory)

- *Preprocessing*.py: Extra Functions for Debug & Profile & Optimize

IV. Demo

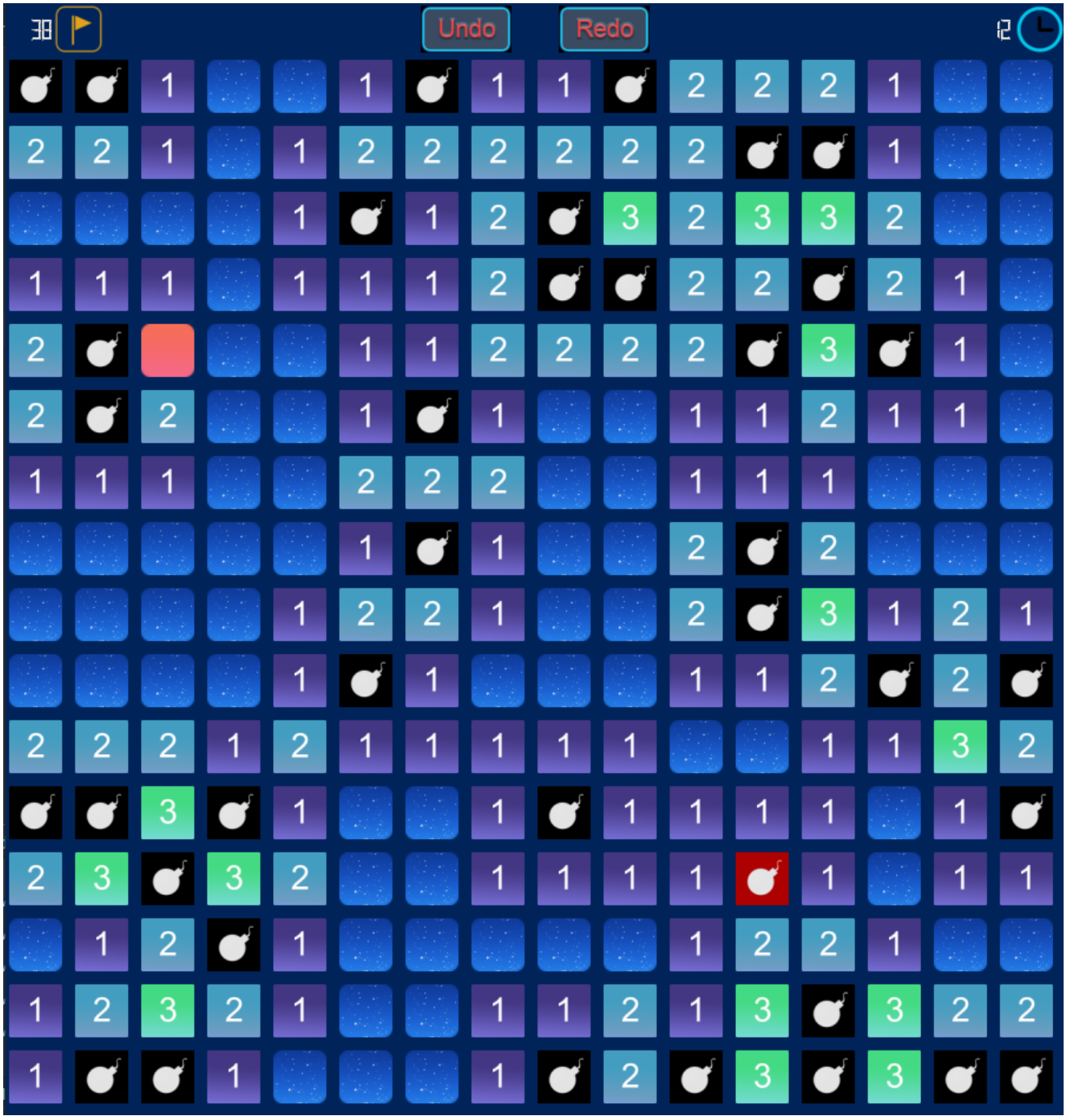

***Figure 2**: Demo*

V. UML Diagram

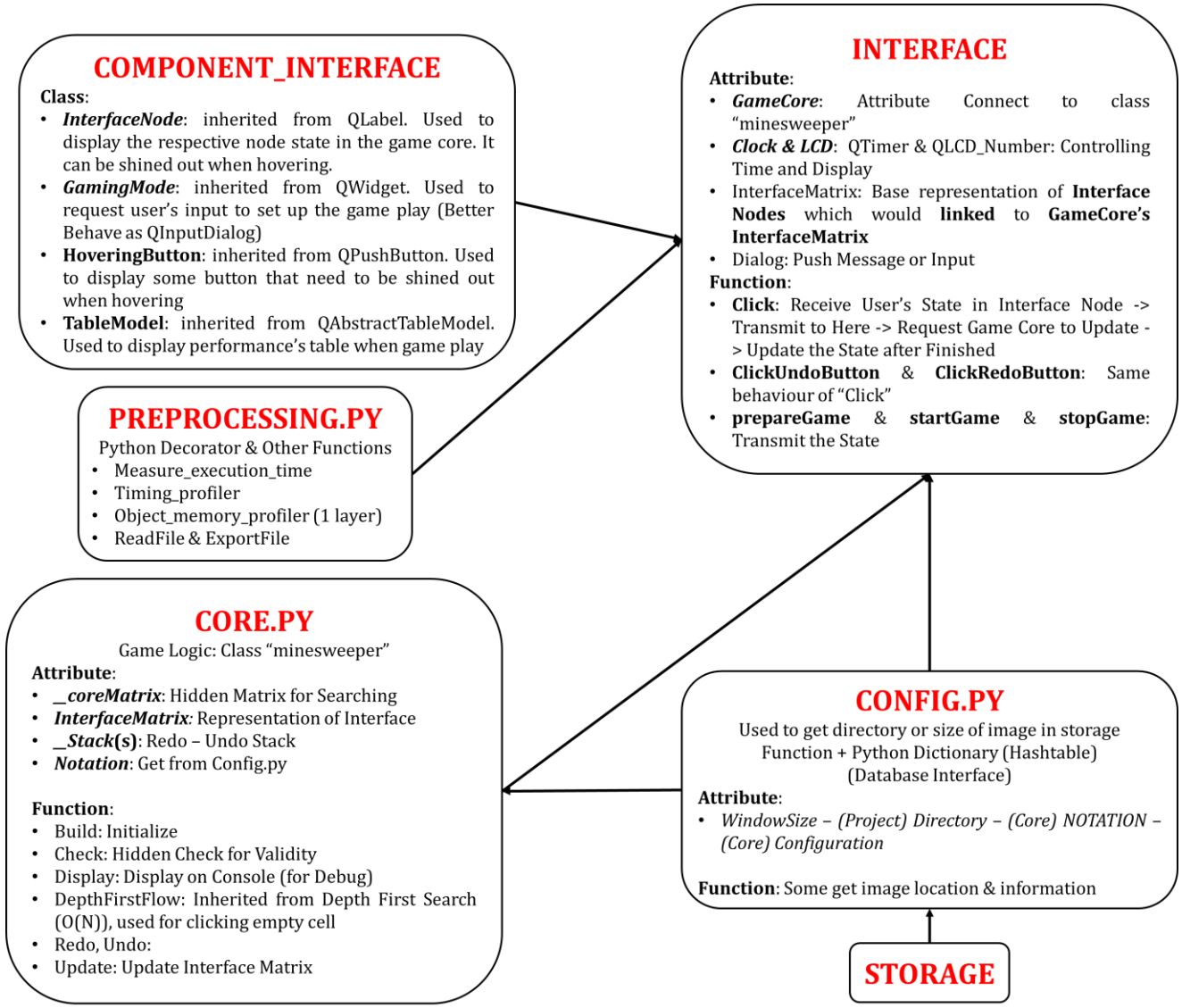

***Figure 3**: Class Diagram*

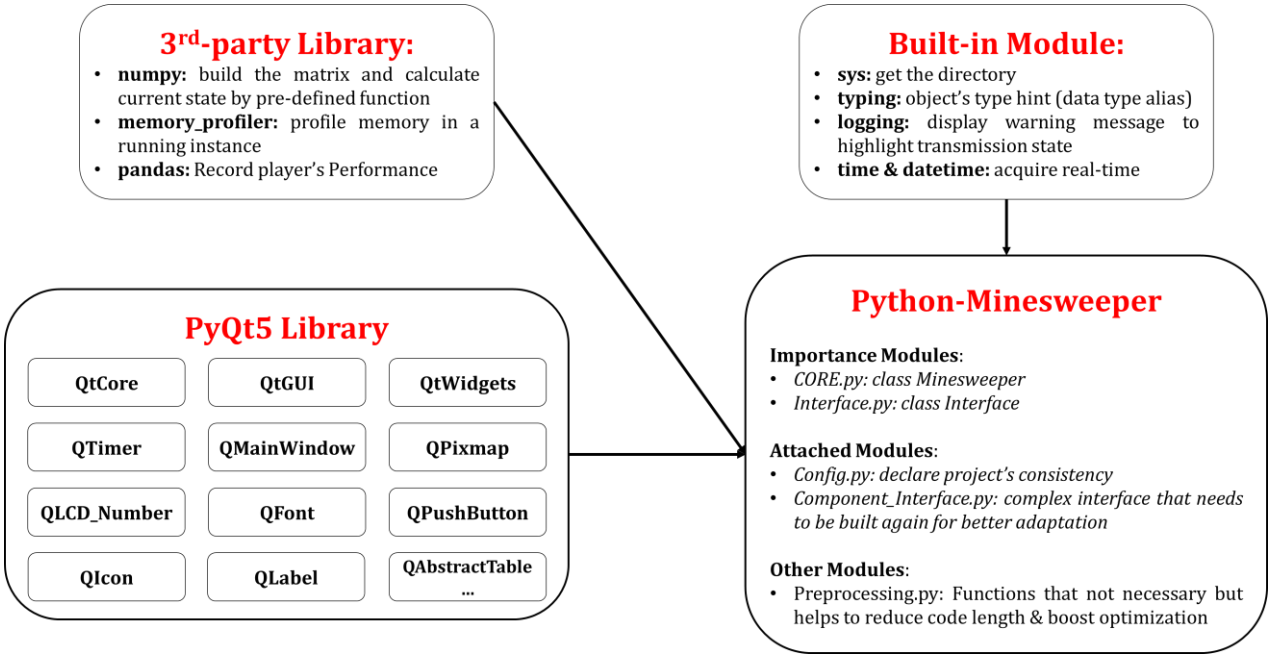

***Figure 4**: Library & Dependency*

VI. Timeline

***Table 1**: The working progess*

Week	Progress	Contributors
Feb 15th	Form team and receive topic from the instructor	Minh, Tuấn, Tiến
Mar 1st	Find information about the game	Minh, Tiến
Mar 8th	Initialize the idea & style, create drafts	Minh, Tuấn
Mar 22nd	Create class diagram, improve the design of game elements	Minh, Tuấn, Tiến
Apr 5th	Begin to code game logic & GUI	Minh, Tuấn
Apr 19th	Make progress in coding game logic & GUI	Minh, Tuấn, Tiến
May 3rd	Further improve the game	Minh
May 17th	Finishing the game & documents	Minh, Tuấn, Tiến
May 24th	Submission & Demo	Minh, Tuấn, Tiến

VII. System Requirements

Minimum Requirements

- Operating System: Windows 8.1
- CPU: Core i3
- Free Space: 1 GB
- RAM: 1.5 Gb

Recommended Requirements

- Operating System: Windows 10
- CPU: Core i3
- Free Space: 1 GB
- RAM: 2 Gb

VIII. Developers' Contribution

1. **Phạm Hoàng Minh** (ITITIU19031, Sophomore): 40%
   Nickname: Ichiru Take
   Department: School of Computer Science and Engineering
   Email: takeichiru2@gmail.com / ITITIU19031@student.hcmiu.edu.vn

2. **Phạm Công Tuấn** (ITITIU19060, Sophomore): 33%
   Department: School of Computer Science and Engineering
   Email: ITITIU19060@student.hcmiu.edu.vn

3. **Trần Ngọc Tiến** (ITITIU19217, Sophomore): 27%
   Department: School of Computer Science and Engineering
   Email: ITITIU19217@student.hcmiu.edu.vn

IX. References and Acknowledgments

- **Packages**: Numpy, Pandas, PyQt5, Memory_Profiler, PyInstaller
- **Built-in Modules:** os, sys, time, cProfile, logging

- **Programming Language:** Python 3.5 +
- **IDE:** PyCharm Pro, VS Code, …
- **Extra Tool:** Adobe Photoshop 2020