A Level Programming Project Report

Junrong Chen

May 24, 2021

Contents

1 Analysis **3**

1.1 Description of the problem 3

1.2 Stakeholders 3

1.2.1 Computer Science teachers 3

1.2.2 Computer Science students 4

1.3 Solve by computational methods 4

1.3.1 Thinking abstractly 4

1.3.2 Thinking ahead 5

1.3.3 Thinking procedurally and decomposition 5

1.3.4 Thinking concurrently 5

1.4 Research 5

1.4.1 LeetCode 6

1.4.2 Codeforces 10

1.5 Features 13

1.6 Limitations 13

1.7 Hardware and software requirements 14

1.8 Success criteria 15

2 Design **16**

3 Development **18**

3.1 Preparation 18
3.1.1 Code editor 18
3.1.2 Runtime environment 18
3.1.3 Version control 18
3.1.4 Project management 19
3.1.5 Create a project repository 19
3.1.6 Configure Git ignore files 20
3.1.7 Create the first commit 20
3.1.8 Create a virtual environment 20
3.2 Hello World! 21
3.2.1 Hello GUI application 21
3.2.2 Hello unit test 23
3.2.3 Hello coverage 25
3.2.4 Hello static check 26
3.2.5 Hello requirements.txt 27
3.2.6 Hello CI/CD 27
3.2.7 Hello Dependabot 28

4 Evaluation **30**

Chapter 1

Analysis

1.1 Description of the problem

A Level Computer Science students need to learn many algorithms and data structures during the course. In the final exam, they need to write pseudocode to solve computational questions. Many students find it is hard to achieve a high score on those questions due to the lack of efficient training. The general method used by students to learn and revise for Computer Science is to attempt and self-mark past paper questions. This works well for ordinary questions. However, for the algorithm questions, different students may produce completely different code solutions. This makes their self-marking very unreliable. It is also too much work for the teacher to mark their solutions one by one. So, in the end, students do not know whether they get things right, and teachers do not know how the students perform and how they can help - especially in this lockdown online learning era where no direct contact between teachers and students is possible.

Both the students and the teachers are looking for a more efficient method to learn and practice.

1.2 Stakeholders

There are two types of stakeholders, Computer Science teachers and Computer Science students.

1.2.1 Computer Science teachers

Computer Science teachers find it is difficult to monitor their students' ability to design and implement algorithms, so they cannot provide efficient help to their students. This software allows them to create coding questions and send

them to the students. After the students hand their solutions back, the software will automatically mark their answers and provide detailed statistical data with simple visualisations. This helps the teachers saving a lot of time and allows them to help the students better.

(TODO): Find actual stakeholders

1.2.2 Computer Science students

Computer Science students find that they tend to lose mark on the algorithms coding questions, so they want more practice. But unlike ordinary questions, they may take a completely different approach towards the questions comparing to the mark scheme, so they do not know whether they get it correct. Students may also think they have got things right, but actually, they have made some mistakes. The software provides a free practice space that automatically marks their solutions and points out their mistakes in real-time. So the students can learn and revise more efficiently.

(TODO): Find actual stakeholders

1.3 Solve by computational methods

1.3.1 Thinking abstractly

In reality, students use pen and paper to write their code solutions. This can be simplified by a code editor and the students can use the keyboard to type in their code. In this way, no 'text scanning' or 'handwriting recognization' is needed which makes the designing and programming much easier. The code editor will also provide a better user experience. Features such as syntax highlighting cannot exist on paper but are possible in the abstracted code editor.

In reality, the students' answer is sent to a teacher to mark it against the mark scheme. The teacher needs to read the code line by line and check whether it is correct. This process is simplified into a judger that marks the code against pre-generated test cases, which transforms a problem that originally cannot be solved by computational method into one which is very easy to be solved by a computer while saving time and costs. When creating a new question, instead of creating a mark scheme for marking, the teacher needs to provide test cases with the correct input and expected output. The judger will run the students' submissions with the input and check whether their output matches the expected one.

1.3.2 Thinking ahead

There are two types of input required.

For the teacher, the software requires them to input the question and test cases as a text file. A question editor need to be provided for this purpose.

For the students, the software requires them to input their code solutions to give them feedback. A code editor need to be provided for this purpose.

A relational database is need to store all the input data.

(TODO)

1.3.3 Thinking procedurally and decomposition

The program will be decomposed into several parts.

- Code editor
- Question editor
- Local database
- Data analyzer
- Judger
- GUI

(TODO)

1.3.4 Thinking concurrently

When auto judging the solution, many test cases can be executed at the same time to reduce the judging time for large test cases.

(TODO)

1.4 Research

There are many coding training websites on the market, most of them share the similar idea, so I will investigate two of the most popular ones.

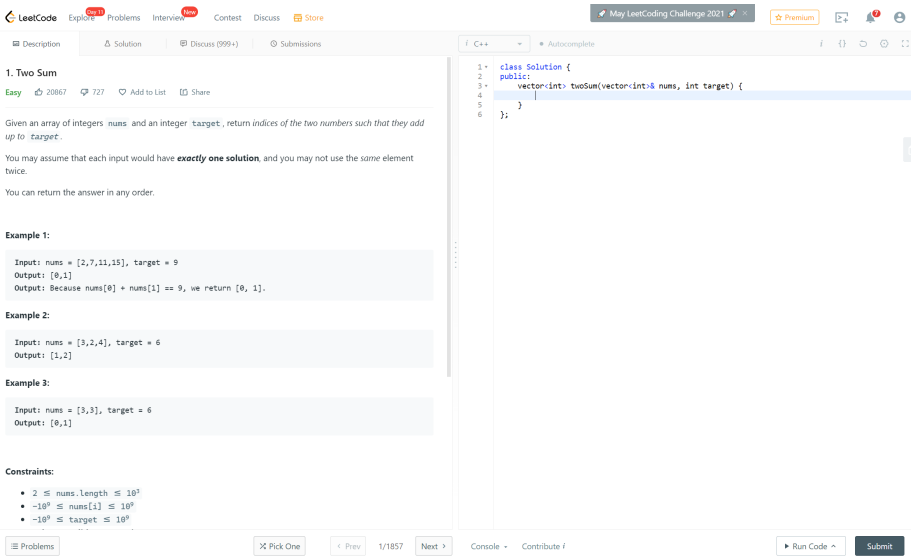
- LeetCode
- Codeforces

1.4.1 LeetCode

LeetCode is a platform for interview coding training, many large companies (Google, Facebook, ...) use it as a part of their interview.

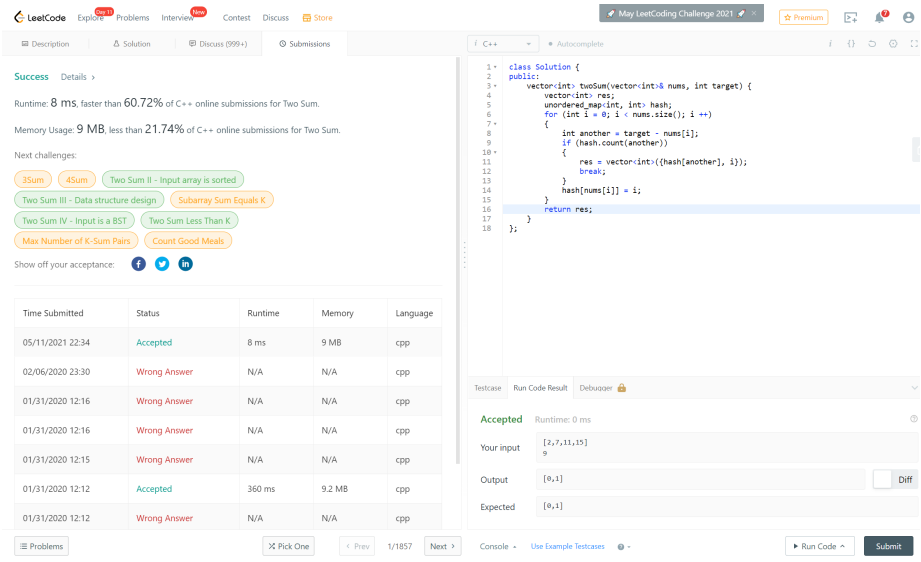
LeetCode provides a database containing more than 1000 coding questions.

Main coding layout



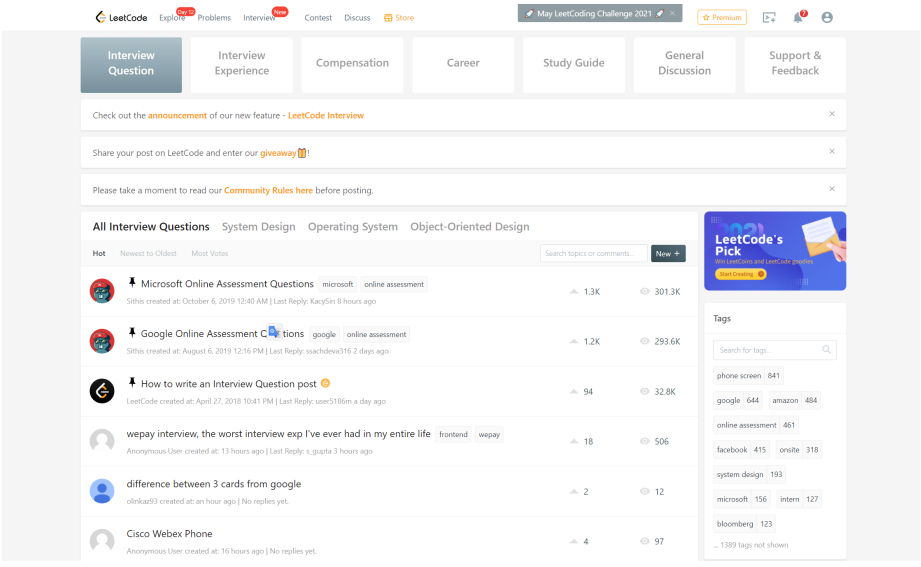
This is the main coding area of LeetCode. The question is on the left and the code editor is on the right, the user can write their code in the code editor and submit their solutions by clicking the button at the down right corner. This layout is a good design as the question and the code editor are displayed on the same screen, which makes it very convenient for the user to read the question and write the code solution. User can *run code* to check their solution against sample test cases in order to avoid stupid errors before formal submission for judging.

Submissions



The submission layout displays the running time and memory usage of the user's code solution. It also compares the result with all other submissions. It lists all history submissions down below. Being able to see the stats of the submission is an interesting feature for the user.

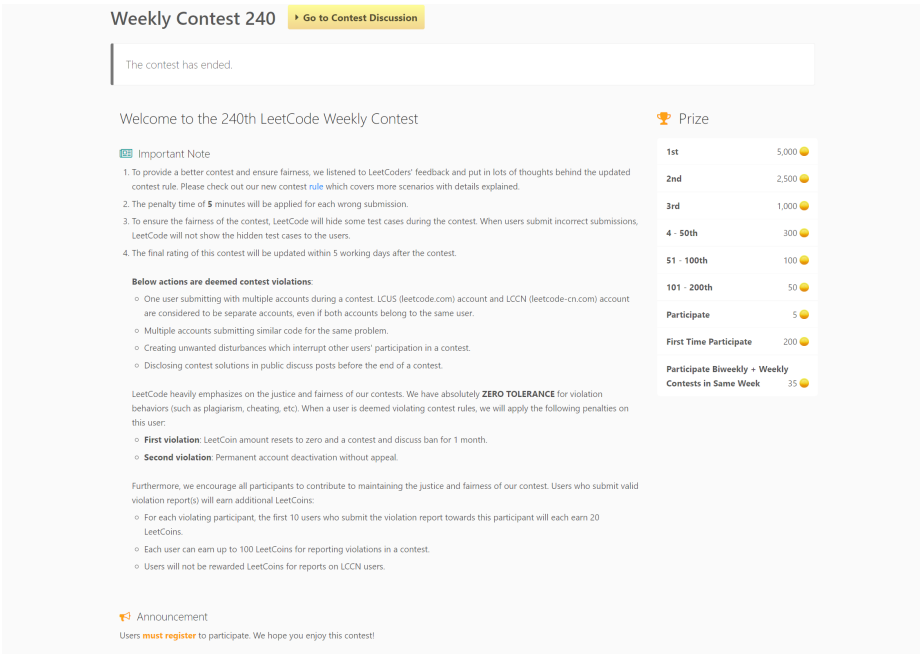
Discussion



There is a discussion page in LeetCode for all users to discuss the questions and their job interview experience. A good discussion environment is very helpful

for self-taught programming.

Contest



Weekly Contest 240 Go to Contest Discussion

The contest has ended.

Welcome to the 240th LeetCode Weekly Contest

Important Note

1. To provide a better contest and ensure fairness, we listened to LeetCoders' feedback and put in lots of thoughts behind the updated contest rule. Please check out our new contest rule which covers more scenarios with details explained.
2. The penalty time of **5** minutes will be applied for each wrong submission.
3. To ensure the fairness of the contest, LeetCode will hide some test cases during the contest. When users submit incorrect submissions, LeetCode will not show the hidden test cases to the users.
4. The final rating of this contest will be updated within 5 working days after the contest.

Below actions are deemed contest violations:

- One user submitting with multiple accounts during a contest. LCUS (leetcode.com) account and LCCN (leetcode-cn.com) account are considered to be separate accounts, even if both accounts belong to the same user.
- Multiple accounts submitting similar code for the same problem.
- Creating unwanted disturbances which interrupt other users' participation in a contest.
- Disclosing contest solutions in public discuss posts before the end of a contest.

LeetCode heavily emphasizes on the justice and fairness of our contests. We have absolutely **ZERO TOLERANCE** for violation behaviors (such as plagiarism, cheating, etc). When a user is deemed violating contest rules, we will apply the following penalties on this user:

- **First violation**: LeetCoin amount resets to zero and a contest and discuss ban for 1 month.
- **Second violation**: Permanent account deactivation without appeal.

Furthermore, we encourage all participants to contribute to maintaining the justice and fairness of our contest. Users who submit valid violation report(s) will earn additional LeetCoins:

- For each violating participant, the first 10 users who submit the violation report towards this participant will each earn 20 LeetCoins.
- Each user can earn up to 100 LeetCoins for reporting violations in a contest.
- Users will not be rewarded LeetCoins for reports on LCCN users.

Announcement

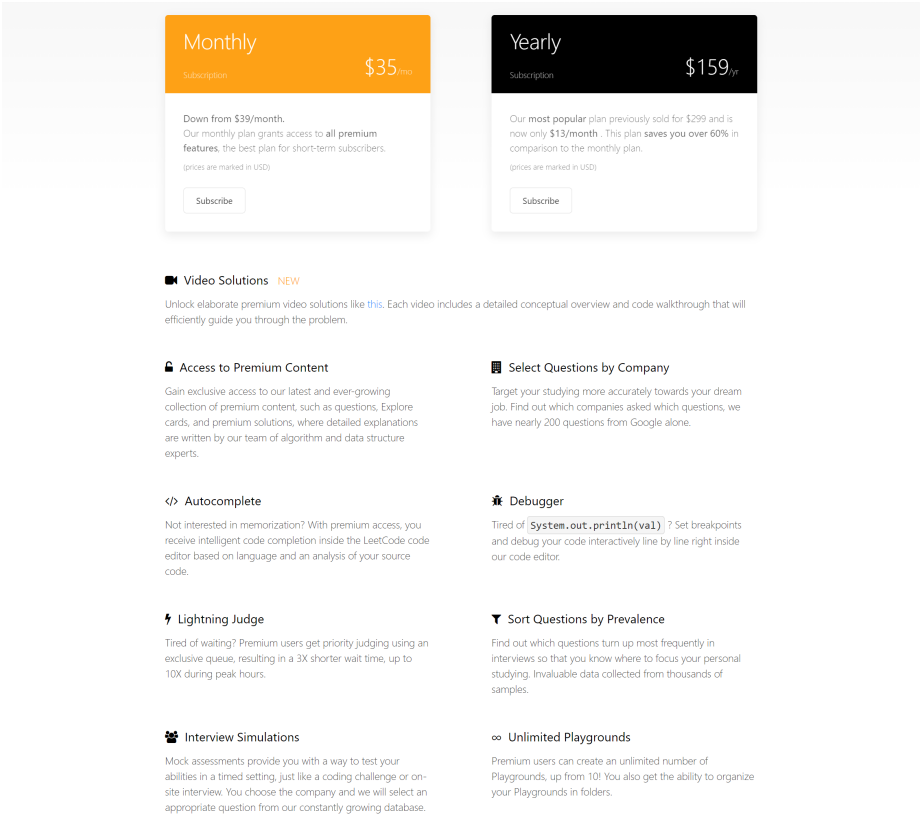
Users must register to participate. We hope you enjoy this contest!

Prize

1st	5,000
2nd	2,500
3rd	1,000
4 - 50th	300
51 - 100th	100
101 - 200th	50
Participate	5
First Time Participate	200
Participate Biweekly + Weekly Contests in Same Week	35

LeetCode helds a contest every week, people try to solve the coding questions as quickly as possible. This motivate people's passion of learning and practicing algorithms.

Pricing



The basic functions of LeetCode is free to use for all users and it charges a fee for premium subscriptions. The premium subscription provides a larger question database, better code editor, faster judger and more.

Analysis

Advantage

- LeetCode is fully web-based, so it works on any platform.
- LeetCode makes it easy to share questions and discuss them with other users.
- LeetCode has a clean and easy to use graphical interface.

Downside

- LeetCode does not allow users to create custom questions.

- There is no way for a teacher or a tutor to assign coursework and get statistics data.
- LeetCode charges a subscription fee for some essential functions.
- LeetCode requires a stable Internet connection to run and debug code.

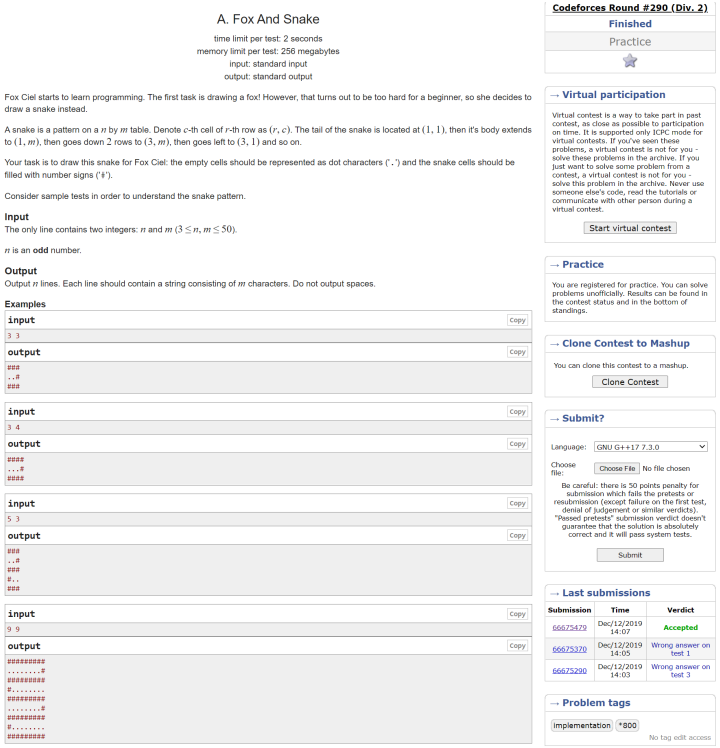
Ideas

- The layout of the coding area is good practice.
- The way LeetCode organizes its question database (Tagging each question with question type/difficulty/accept rate) is good practice.
- My software will be completely free and open sourced with a good editor and fast judger out of the box.
- The *run code* function for debugging before formal submission is a very useful feature.
- Some format of coding competition can be held by the user.

1.4.2 Codeforces

Codeforces is a competitive coding platform, it is mainly used by people to held coding competition. It takes a similar approach to judge code with test data. There is no code editor provided, user are required to write and debug their solution on their own IDE and only submit the source code for judging.

Main question layout



A. Fox And Snake

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Fox Ciel starts to learn programming. The first task is drawing a fox! However, that turns out to be too hard for a beginner, so she decides to draw a snake instead.

A snake is a pattern on a n by m table. Denote c-th cell of r-th row as (r, c). The tail of the snake is located at $(1, 1)$, then it's body extends to $(1, m)$, then goes down 2 rows to $(3, m)$, then goes left to $(3, 1)$ and so on.

Your task is to draw this snake for Fox Ciel: the empty cells should be represented as dot characters ('.') and the snake cells should be filled with number signs ('#').

Consider sample tests in order to understand the snake pattern.

Input
The only line contains two integers: n and m ($3 \leq n, m \leq 50$).

n is an **odd** number.

Output
Output n lines. Each line should contain a string consisting of m characters. Do not output spaces.

Examples

input
```
3 3
```
output
```
###
..#
###
```

input
```
3 4
```
output
```
####
...#
####
```

input
```
5 3
```
output
```
###
..#
###
#..
###
```

input
```
9 9
```
output
```
#########
........#
#########
#........
#########
........#
#########
#........
#########
```

Codeforces Round #290 (Div. 2)
Finished
Practice

→ Virtual participation

Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you - solve these problems in the archive. If you just want to solve some problem from a contest, a virtual contest is not for you - solve this problem in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.

Start virtual contest

→ Practice

You are registered for practice. You can solve problems unofficially. Results can be found in the contest status and in the bottom of standings.

→ Clone Contest to Mashup

You can clone this contest to a mashup.

Clone Contest

→ Submit?

Language: GNU G++17 7.3.0

Choose file: Choose File No file chosen

Be careful: there is 50 points penalty for submission which fails the pretests or resubmission (except failure on the first test, denial of judgement or similar verdicts). "Passed pretests" submission verdict doesn't guarantee that the solution is absolutely correct and it will pass system tests.

Submit

→ Last submissions

Submission	Time	Verdict
66675479	Dec/12/2019 14:07	Accepted
66675370	Dec/12/2019 14:05	Wrong answer on test 1
66675290	Dec/12/2019 14:03	Wrong answer on test 3

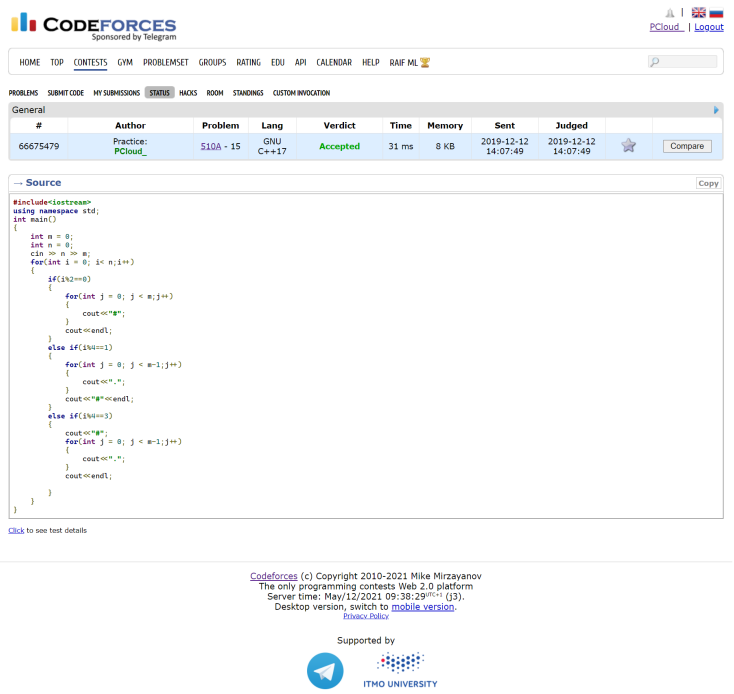
→ Problem tags

implementation *800

No tag edit access

The question layout displays the description of the question and provides sample test cases. It also shows the performance requirement for the code solution.

Submission



When the user submits the code, the code enters a queue waiting for judging, then the user can look up their result.

Contest

Codeforces helds contest regularly similarly to LeetCode, but with much harder quesitons.

Analysis

Advantage

- Codeforces is completely free and it is maintained by its community.
- Codeforces is fully web-based so it works on any platform.
- Codeforces supports all mainstream programming languages.
- Codeforces supports custom questions.

Downside

- Codeforces focuses more on coding competition purpose instead of learning and practicing.
- Codeforces does not provide out of box code editor. Users need to write code in their own code editor.
- Codeforces does not provide debugging feature, users need to run and debug their code in their own runtime environment.
- The juding process will be very slow when many users are submitting solutions at the same time.

Ideas

- The sample test cases are very useful for the user to debug their code solution.
- Set a time and space limit for judging to prevent malicious code from using up all system resources.
- Provide out of box code editor, runtime environment to make the software easy to use.
- Use a local judger to judge the user's solution instead of uploading it to a server to speed up the judging speed.

1.5 Features

(TODO)

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

1.6 Limitations

There are a few limitations of this software.

The software is written in Python instead of web-based which means extra software needs to be downloaded by the user. Because Python has good cross-platform compatibility, the software can still run on all mainstream platform (Windows, Mac OS and Linux) which minimize the inconvenience, but downloading an extra software is still inconvenient and may violate the IT security policy of some schools.

The judger can only accept code submission in Python. The reason for choosing Python is because it has a very similar grammar to the pseudocode and most students are already very familiar with it. Creating a compiler for 'Pseudocode Programming Language' is too complex for this project. So only Python is supported for now. The reason for not supporting other programming language is that an extra runtime environment needs to be installed (compiler/interpreter), so it is not possible to support them out of the box. But some extra configuration might be provided to allow submit code solution in other languages.

Unlike LeetCode, there is no Discussion pages for users to discuss questions because it is a Python program instead of a web one. But this is not a big problem, students and teachers should use an existing product such as Microsoft Teams which has very good support in sharing code snippets. It is unnecessary to rebuild the wheel.

Distribute the questions and assignments is still something inconvenient. Currently, distributing questions and assignments requires the teacher to first export the questions and assignments, then send them to the students through email or file-sharing platforms. When the students finish working, they need to send their result back through email or other apps. I have attempted to integrate the file-sharing function with the existing platform - the Microsoft Teams Assignment function. But very, unfortunately, the Graph API required for this operation is still in beta version, which means it can only be tested in the development environment and cannot be used in production. So for now, the users still have to use this inconvenient way to share questions and submissions. But the further, the integration with some existing platforms may improve the experience.

There are no good ways to maintain and distribute a large question database. Computer Science teachers are requied to maintain a database for their own students. But this is a difficult work. Creating good test cases is much time consuming than writing a mark scheme, it is very likely for a wrong solution to pass the judging if the test cases are not good enough. It relies on the teacher who creates the questions to consider everything clearly to minimize its impact.

The judger can only simply compare the students' output with the expected output, if there is a format error such as trailing space and extra newline in their output, which will not be considered as a mistake in a real exam, will be marked as a wrong answer by the judger. So students may need to spend extra time debugging their output format.

1.7 Hardware and software requirements

Due to the good compatibility of Python, it can run on all mainstream desktop operating systems. Linux, macOS and Windows. The software itself does not contain complex graphical effects or large scale computation. So it should be able to run under 1C CPU and 512M RAM. However, the code solutions provided by the students may need more resource to run and test. So 2C CPU and

2G RAM is recommended.

(TODO)

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

1.8 Success criteria

(TODO)

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

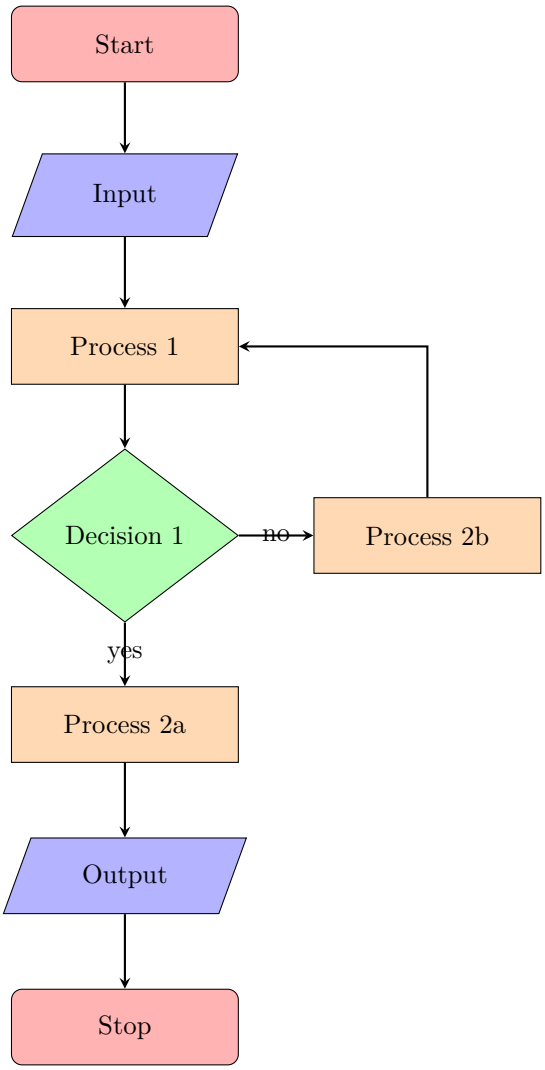
Chapter 2

Design

This is the design chapter.

(TODO)

Sample flow chart


Start
Input
Process 1
Decision 1
no
Process 2b
yes
Process 2a
Output
Stop

Chapter 3

Development

3.1 Preparation

3.1.1 Code editor

Download and install VS Code.

Instead of using a large IDE with everything pre-configured, I decide to use a code editor to write source code and a terminal to execute commands. This gives me more control on my project.

VS Code is a free and open source code editor which also have greate support for Python.

I decide to use VS Code as my code editor.

3.1.2 Runtime environment

Download and install Python 3.9.5.

I simply choose the latest Python release for this project. When I release the software, the Python interpreter will be packed with the binary files so there is no need to worry about the compatibility with the Python installation in users' environment.

3.1.3 Version control

Download and install Git.

Git is a free and open source distributed version control system. It records every *commit* I made to the source code and allows me to revert back to any previous *commit*. This makes it easy to roll back to a certain version and locate bugs. It also allow me to create new *branches* which is useful when experimenting new features without worrying about damaging the stable code.

I decide to use Git as the version control system for this project.

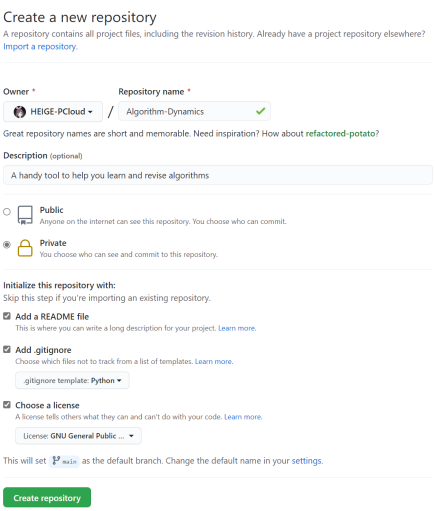
3.1.4 Project management

GitHub is a code hosting platform which supports many project management features. The *Issue* allows my stakeholders to report bugs and suggests features easily. The *Action* provides support for CI/CD. The *Project* provides support for managing and organizing the TODO list for the project.

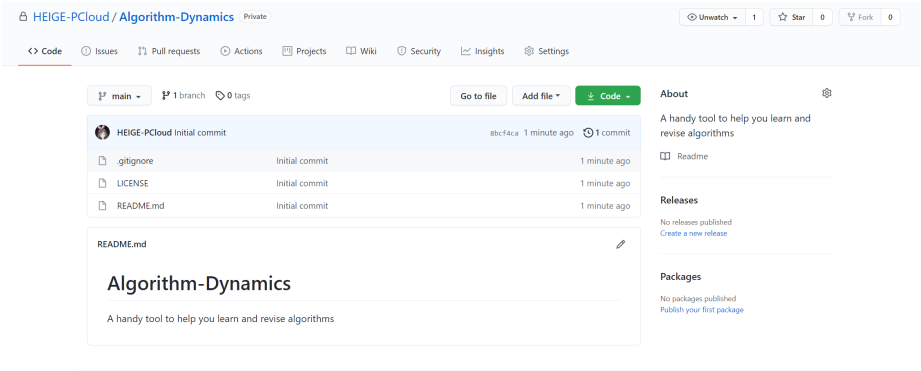
I decide to use GitHub as the code hosting platform for this project.

3.1.5 Create a project repository

I need to create a private repository to store and manage the source code.



This is the inital screenshot of this project repository, there is not many things there right now, but it will be much more vivid as time goes forward.



I need to `clone` this repository in order to add files and write code to it.

```
git clone https://github.com/HEIGE-PCloud/Algorithm-Dynamics.git
cd Algorithm-Dynamics
```

3.1.6 Configure Git ignore files

The `.gitigore` file lists files not being managed by the version control system. For example, I don't want to track the changes of the cache files or the log file.

GitHub has already generated a nice `.gitigore` file for this Python project, but I need to further ignore aditional two files, the config file from the VS Code and the pdf preview of this report.

3.1.7 Create the first commit

After I have made the changes, I need to create a *commit* to comfirm the chagnes and let Git record it, so I can go back here again in the future if needed.

```
$ git add .gitignore
$ git commit -m "chore(configure-environment): update .gitignore
- Ignore config files for VS Code
- Ignore pdf files for the report"
$ git push
```

Now, I have created the first commit and pushed it to the remote repository.

3.1.8 Create a virtual environment

A virtual environment is a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional

packages.

My Python program will use many external libraries, at the same time, there are other Python projects on my computer require the same library with differnt version requirements, so I need a virtual environment to isolate the dependicies for different projects.

First, I install the lastest release of virtualenv.

```
$ pip install virtualenv
```

Next, I create a new virtual environment under the project folder.

```
$ virtualenv env
```

Finally, I need to activate the virtual environment.

```
$ ./env/Scripts/activate
```

Now I have a clean environment to install and manage all the dependicies and packages for this project.

3.2 Hello World!

3.2.1 Hello GUI application

I will create a simple GUI application to go through all the development stages to verify the environment is ready.

I will use PyQt6 as the GUI framework. PyQt is a set of Python bindings for The Qt Company's Qt application framework and runs on all platforms supported by Qt including Windows, macOS, Linux, iOS and Android.

PyQt is licensed under GNU GPL v3, which means the program needs to licensed under GNU GPL v3 as well.

Fisrt, I install PyQt6 in the virtual environment.

```
$ (env)  pip install PyQt6
```

I then create `helloworld.py` under `src` folder as the source code file for the Hello World application.

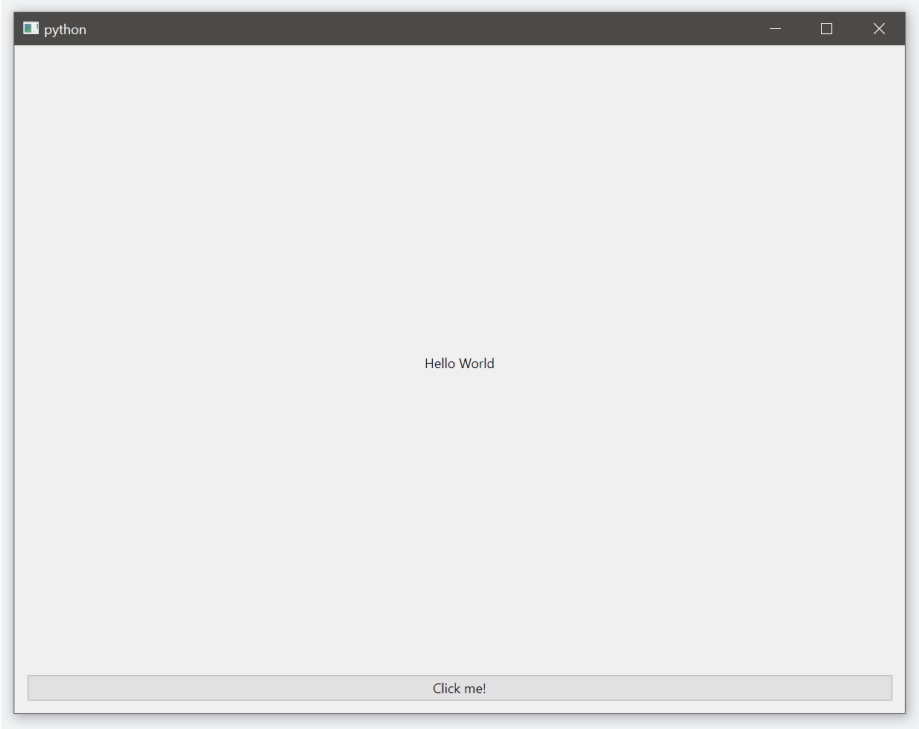
In this hello world program, I will create a custom widget contains a text label and a button. The text label will display a hello world sentence. When the button is clicked by the user, the text will randomly change to the hello world message in another language.

```
import sys
import random
from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import (QApplication, QLabel, QWidget, QPushButton,
                             QVBoxLayout)


class MyWidget(QWidget):
    def __init__(self):
        super().__init__()
        # List for all hello text
        self.hello = ["Hallo Welt", "Hei maailma", "Hola Mundo"]
        # Add a button for changing the hello text
        self.button = QPushButton("Click me!")
        # Connect the button to magic function
        self.button.clicked.connect(self.sayHello)
        # Add a label to display the text
        self.text = QLabel("Hello World", )
        # Set alignment to center
        self.text.setAlignment(Qt.AlignmentFlag.AlignCenter)
        # Set layout of the widget
        self.layout = QVBoxLayout(self)
        self.layout.addWidget(self.text)
        self.layout.addWidget(self.button)

    def sayHello(self):
        self.text.setText(random.choice(self.hello))


if __name__ == "__main__":
    # Create a new application
    app = QApplication(sys.argv)
    # Create a new widegt
    widget = MyWidget()
    # Resize the widget
    widget.resize(800, 600)
    # Show the widget
    widget.show()
    # Run the app
    sys.exit(app.exec())
```

I run the GUI Application with the following command.

```
$ (env) pythonw ./src/helloworld.py
```

An hello world window shows up correctly. When the I click the button, the text in the middle will change randomly.



3.2.2 Hello unit test

I am going to automate the test process with Pytest unit test framework. It can run through my pre-set test data automatically without me clicking each button or inputing each value by hand. This saves a lot of time and improves the quality of my tests.

I need to install Pytest into my virtual environment first.

```
$ (env) pip install Pytest
```

Next I need to write the test code for the hello world program. Under `test` folder, I create `test_helloworld.py`.

Here I create three tests for the hello world program. First I test its initial state, this ensure the widget loads up with the correct text and layout. Second I test its `sayHello` function to make sure the text is changed correctly (white box test). Third I test its button, I simulate the click with `QTest.mouseClick` to make sure the button is working (black box test).

```
import sys
```

```
from PyQt6.QtWidgets import QApplication
from PyQt6.QtCore import Qt
from PyQt6.QtTest import QTest
from src.helloworld import MyWidget


app = QApplication(sys.argv)
hello = ["Hallo Welt", "Hei maailma", "Hola Mundo"]


def test_initWidget():
    """
    Test the initial state of the widget.
    """
    widget = MyWidget()
    assert widget.text.text() == 'Hello World'
    assert widget.button.text() == 'Click me!'
    assert widget.hello == hello


def test_sayHello():
    """
    Whitebox test the sayHello function.
    Execute sayHello 10000 times, remove each random result from the list.
    The final list should be empty.
    """
    widget = MyWidget()
    for i in range(10000):
        widget.sayHello()
        text = widget.text.text()
        if text in hello:
            hello.remove(widget.text.text())
    assert len(hello) == 0


def test_mouseClick():
    """
    Blackbox test the sayHello function.
    Use QTest.mouseClick to click the button 10000 times.
    Remove each random result from the list.
    The final list should be empty as well.
    """
    widget = MyWidget()
    for i in range(10000):
        QTest.mouseClick(widget.button, Qt.MouseButton.LeftButton)
        text = widget.text.text()
        if text in hello:
            hello.remove(widget.text.text())
    assert len(hello) == 0
```

3.2.3 Hello coverage

Coverage is a measure used to describe the degree to which the source code of a program is executed when a test runs. A program with high test coverage has had more of its source code executed during testing, which suggests it has a lower chance of containing undetected software bugs compared to a program with low test coverage. I will use the coverage to provide evidence for the quality of my tests.

I need to install Pytest-cov to calculate the coverage of the tests.

```
$ (env) pip install Pytest-cov
```

I need to add a config file for the Pytest-cov to exclude the test and debug code from the coverage calculation.

Create `.coveragerc`.

```
[run]
source = src/
branch = True

[report]
exclude_lines =
    # Have to re-enable the standard pragma
    pragma: no cover

    # Don't complain about missing debug-only code:
    def __repr__
    if self\.debug

    # Don't complain if tests don't hit defensive assertion code:
    raise AssertionError
    raise NotImplementedError

    # Don't complain if non-runnable code isn't run:
    if 0:
    if __name__ == .__main__.:
```

Now, I run the unit test with this command. `--cov` configures the folder of my source code. `--cov-report` configures the format of the coverage output, `term` lets it to be printed directly to the termial. '-vv' shows the details of my tests. `--cov-config` configures the location of our config file for coverage, which is the `.coveragerc` file I just created.

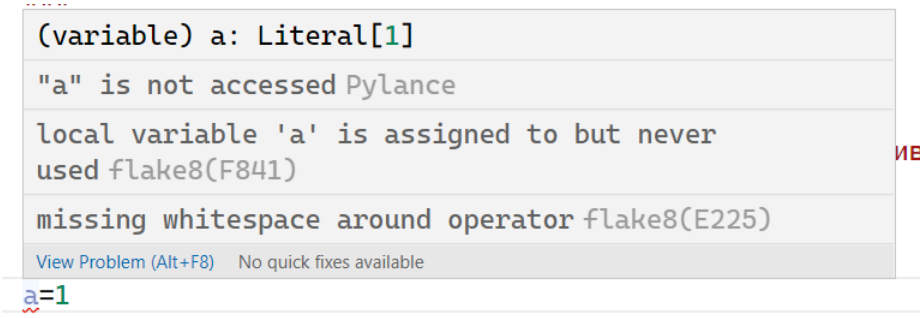
```
$ (env) pytest --cov=src -vv --cov-report=term --cov-config=./.coveragerc
```

Here is the output of the test. My 3 tests `test_initWidget`, `test_sayHello`, `test_mouseClick` are executed and passed correctly. And the coverage report shows that the coverage of my tests is 100% which means all code is executed during the tests. I aim for a 95%+ coverage for the formal project.

```
========================== test session starts ==========================
platform win32 -- Python 3.9.5, pytest-6.2.4, py-1.10.0, pluggy-0.13.1 --
cachedir: .pytest_cache
plugins: cov-2.12.0
collected 3 items

tests/test_helloworld.py::test_initWidget PASSED [ 33%]
tests/test_helloworld.py::test_sayHello PASSED  [ 66%]
tests/test_helloworld.py::test_mouseClick PASSED [100%]

------------ coverage: platform win32, python 3.9.5-final-0 ------------
Name                 Stmts   Miss Branch BrPart  Cover
-------------------------------------------------------
src\__init__.py          0      0      0      0   100%
src\helloworld.py       17      0      0      0   100%
-------------------------------------------------------
TOTAL                   17      0      0      0   100%


============================ 3 passed in 0.53s ============================
```

3.2.4 Hello static check

The style of the code is important as well. It will make the maintenance much easier if all variables have meaningful names, no trailing whitespace, proper blank lines, etc. I will use the tool flake8 to perform static check of my code. flake8 nicely works with VS Code, so I will have useful notifications so these issues can be fixed quickly.



Another useful tool is autopep8, it auto formats my source code and decides the

whichspaces and blank lines wisely.

I install these two tools to monitor and improve the style of my code.

```
$ (env) pip install flake8 autopep8
```

3.2.5 Hello requirements.txt

I have installed a lot of packages for my project. `requirements.txt` is a file records all the packages I have installed, so the packages can be easily managed.

```
$ (env) pip freeze > requirements.txt
```

3.2.6 Hello CI/CD

Continuous integration (CI) and continuous delivery (CD) embody a culture, set of operating principles, and collection of practices that enable application development teams to deliver code changes more frequently and reliably.

I will use GitHub Actions to auto test, build and deliver my application.

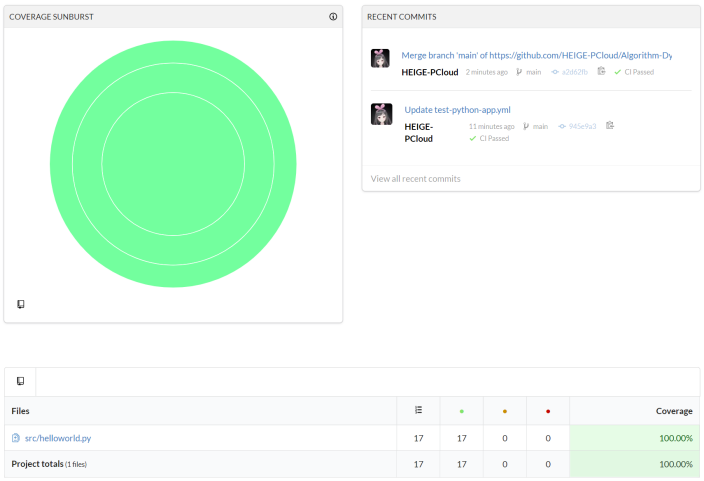
I will use Codecov to monitor the test quailty and coverage.

Create workflow file at `.github/workflows/test-python-app.yml`.

```
name: Test Python Application

on: [push, pull_request, workflow_dispatch]

jobs:
  build:
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [macos-latest, windows-latest]
      fail-fast: false
    steps:
    - uses: actions/checkout@v2
    - name: Set up Python 3.9
      uses: actions/setup-python@v2
      with:
        python-version: 3.9
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt
```

```
    - name: Lint with flake8
      run: |
        flake8 src --count --show-source --statistics
    - name: Test with pytest
      run: |
        pytest --cov=src --cov-config=.coveragerc
    - name: Upload coverage to Codecov
      uses: codecov/codecov-action@v1.5.0
      with:
        fail_ci_if_error: true
        token: ${{ secrets.CODECOV_TOKEN }}
```

I can view the coverage report at Codecov.



3.2.7 Hello Dependabot

The dependicies I install may get upgraded by their maintainer later during my developing process. I need them to be managed and upgraded automatically. I use GitHub Dependabot to manage the dependicies.

Configure Dependabot.

Create Dependabot config file at `.github/workflows/dependabot.yml`.

```
version: 2
updates:
  - package-ecosystem: "pip"
    directory: "/"
    schedule:
      interval: "daily"
```

So the dependicies will be automatically checked everyday and the bot will create a new pull request when a new update is detected.

Chapter 4

Evaluation

This is the Evaluation chapter.

(TODO)