Programming assignments#1

Sorting

1. Description

You start with an initial power value of power, an initial score of 0, and a collection of tokens represented by an integer array tokens, where each tokens[i] indicates the value of the i-th token.

Your goal is to maximize your total score by strategically playing these tokens. In each move, you can select an unplayed token and choose one of the following two actions (you cannot perform both actions on the same token):

1. **Play Face-up:** If your current power is at least tokens[i], you can play the i-th token face-up. This action costs you tokens[i] power but increases your score by 1.
2. **Play Face-down:** If your current score is at least 1, you can play the i-th token face-down. This action decreases your score by 1 but increases your power by tokens[i].

Return the maximum total score you can achieve after playing any number of tokens.

Example1:

Input: initial power = 50, tokens = [100]

1. Since your score is 0 initially, you cannot play the token face-down. You also cannot play it face-up since your power (50) is less than token[0] (100).
2. The maximum score achievable is 0.

Example2:

Input: initial power = 150, tokens = [200,100]

1. Play token[1] (100) face-up, reducing your power to 50 and increasing your score to 1.
2. There is no need to play token[0], since you cannot play it face-up to add to your score.
3. The maximum score achievable is 1.

Example3:

Input: initial power = 200, tokens = [100,200,300,400]

1. Play token[0] (100) face-up, reducing power to 100 and increasing score to 1.
2. Play token[3] (400) face-down, increasing power to 500 and reducing score

to 0.

3. Play token[1] (200) face-up, reducing power to 300 and increasing score to 1.
4. Play token[2] (300) face-up, reducing power to 0 and increasing score to 2.
5. The maximum score achievable is 2.

2. Input Format

The input file is composed of two parts. The first part contains the initial power (an integer) you start with, and the second part contains an array of tokens (integers). **When the input is -1, it signifies the end of the input.** The numbers before -1 are considered as the tokens.

$0 < tokens.length <= 10^3$

$0 <= tokens[i]<=1000$, $power < 10^4$

```
200
100 200 300 400 -1
```

3. Output Format

Print the maximum possible score.

```
2
```

4. Program

1. Program in C/C++.
2. Do not use built-in sorting functions.

5. Grading

1. TA will randomly check your code. If it's found that you use any built-in sorting function, you will get **a penalty of -40 on your grade.**
2. Score:
   - Q1~Q10: Each contributes 10%
   - Q1~Q7 are open case
   - Q8~Q10 are hidden case
3. The score of questions is evaluated by the OJ system. **TA will evaluate your grades based on the most recent version of your submissions.**

6. E3 Submission

1. Submit your source file to the E3 system

[Student_ID_Number]_hw1.cpp (.c)

2. Please submit the source code of your latest submission on the OJ. Please make sure that all characters of the filename are in **lowercase**. For example, if your student number is 313510224, the name of your source file should be "313510224_hw1.cpp".
3. Remember the submission rules mentioned above, or you will be penalized by **-15 points** on your grade.

7. Policy

1. The upload deadline would be at **23:59 on October 15, 2024.** The score will be **30% off** if your submission is late.
2. **Plagiarism: -100 points.**

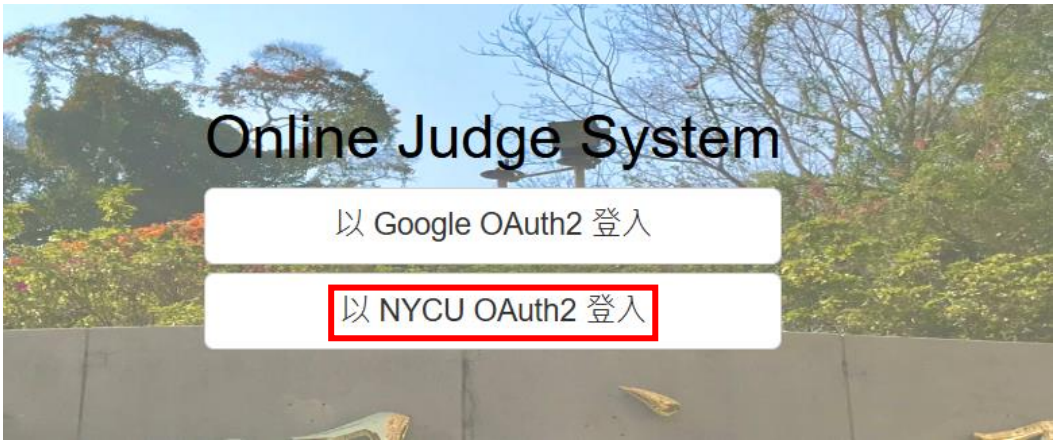
8. Problems

1. If you have any problem with this project, please post it on the E3 forum.
2. If you have any private problem, you can send emails to: jasontsai.ee13@nycu.edu.tw

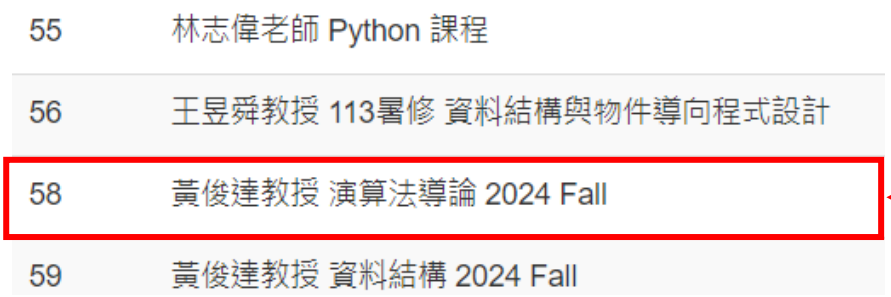
↓ **Tutorial for Formosa OJ** ↓

Formosa OJ

1. Link: https://formosa.oj.cs.nycu.edu.tw/
2. Login: Use your NYCU Portal Account



3. Joining the group “黃俊達教授 演算法導論 2024 Fall”.



4. After entering, you will see the following page. Click “Problems” to enjoy your assignment. For more information, please check “使用說明”.