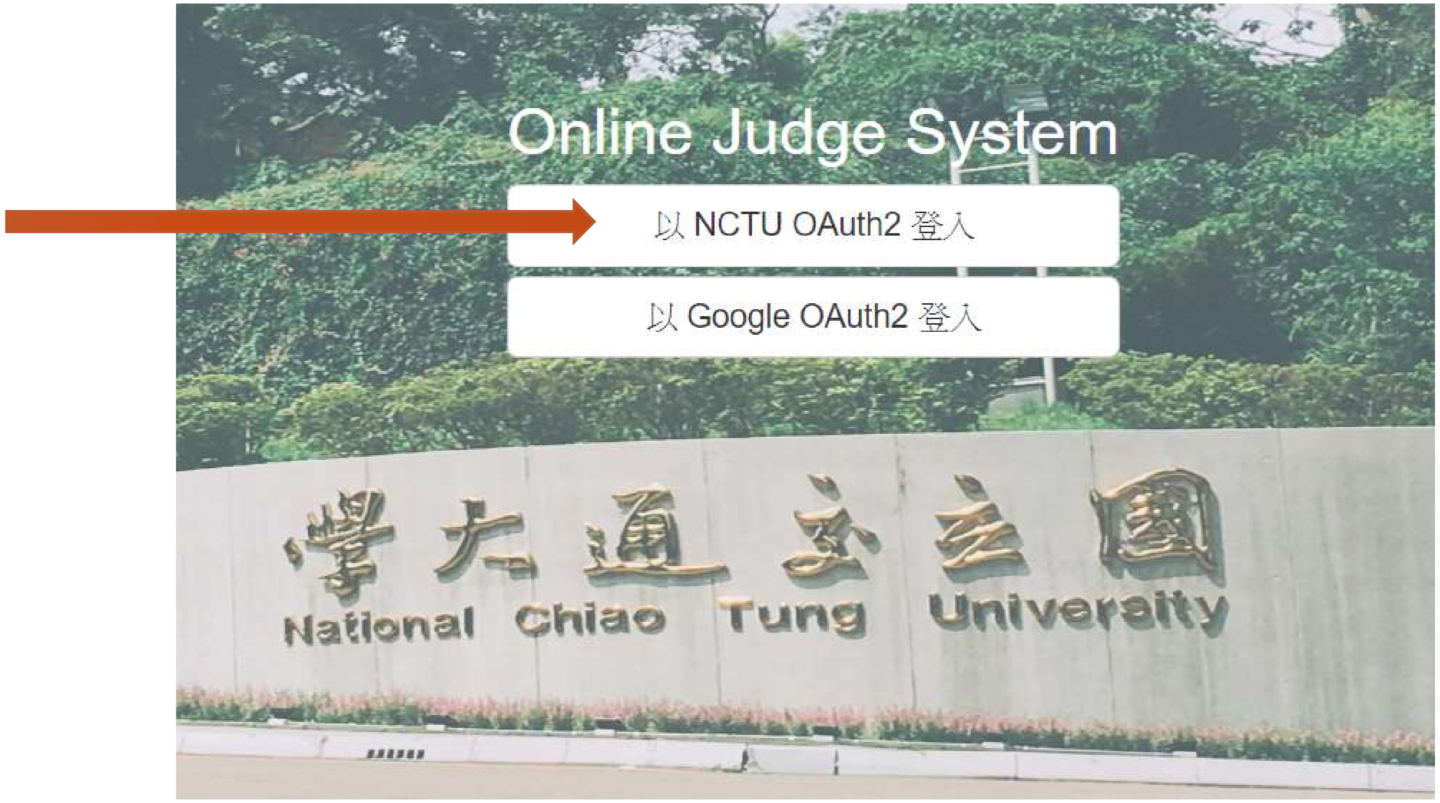
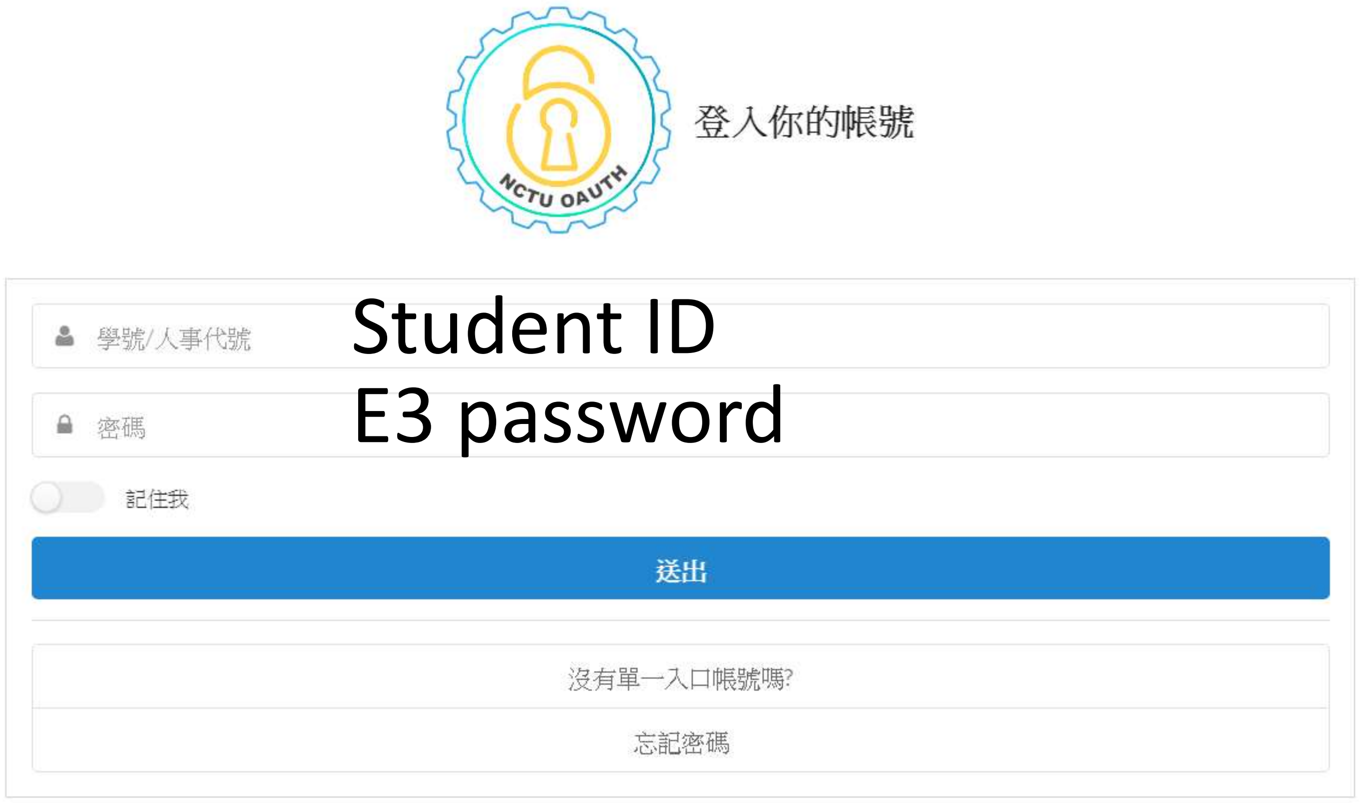
Final Project of Discrete Mathematics

Formosa OJ

Link: https://oj.nctu.me/



Formosa OJ



登入你的帳號

學號/人事代號

Student ID

密碼

E3 password

記住我

送出

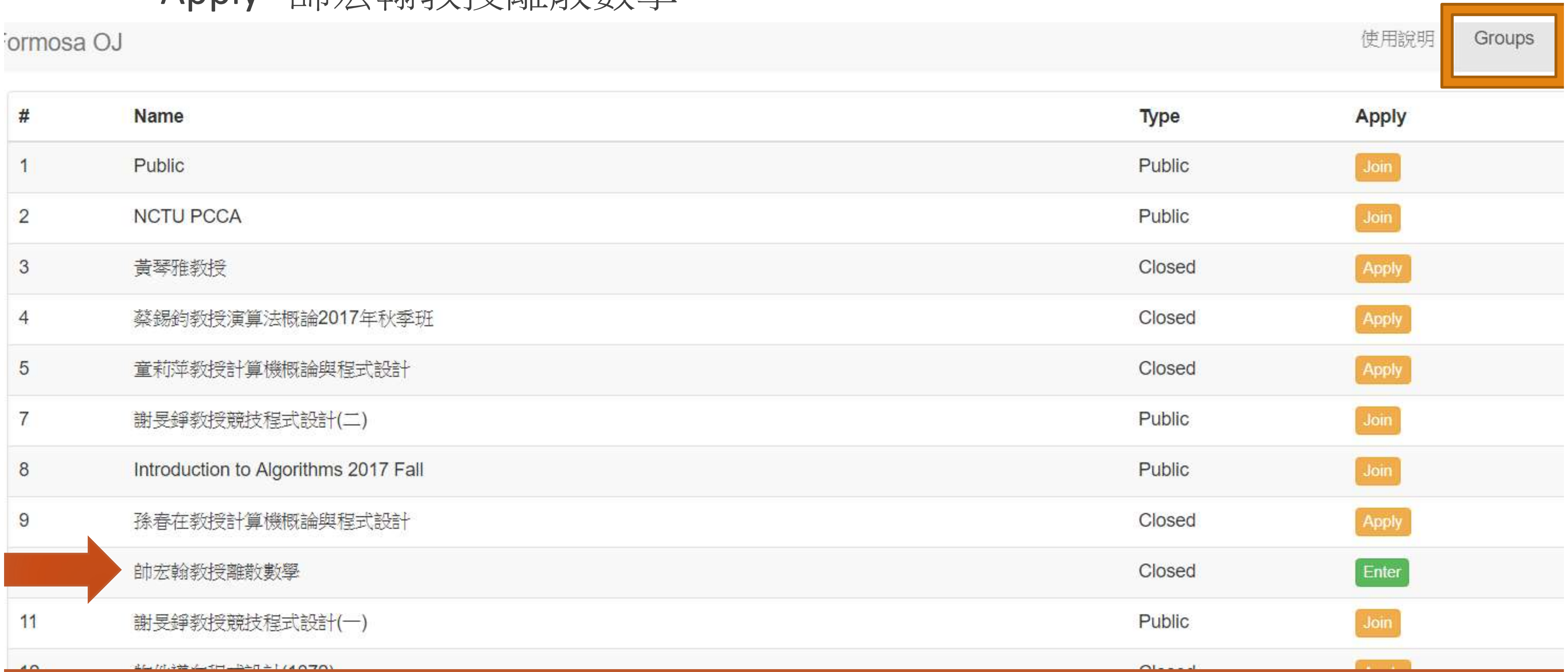
沒有單一入口帳號嗎?

忘記密碼

Formosa OJ

Link: https://oj.nctu.me/groups/

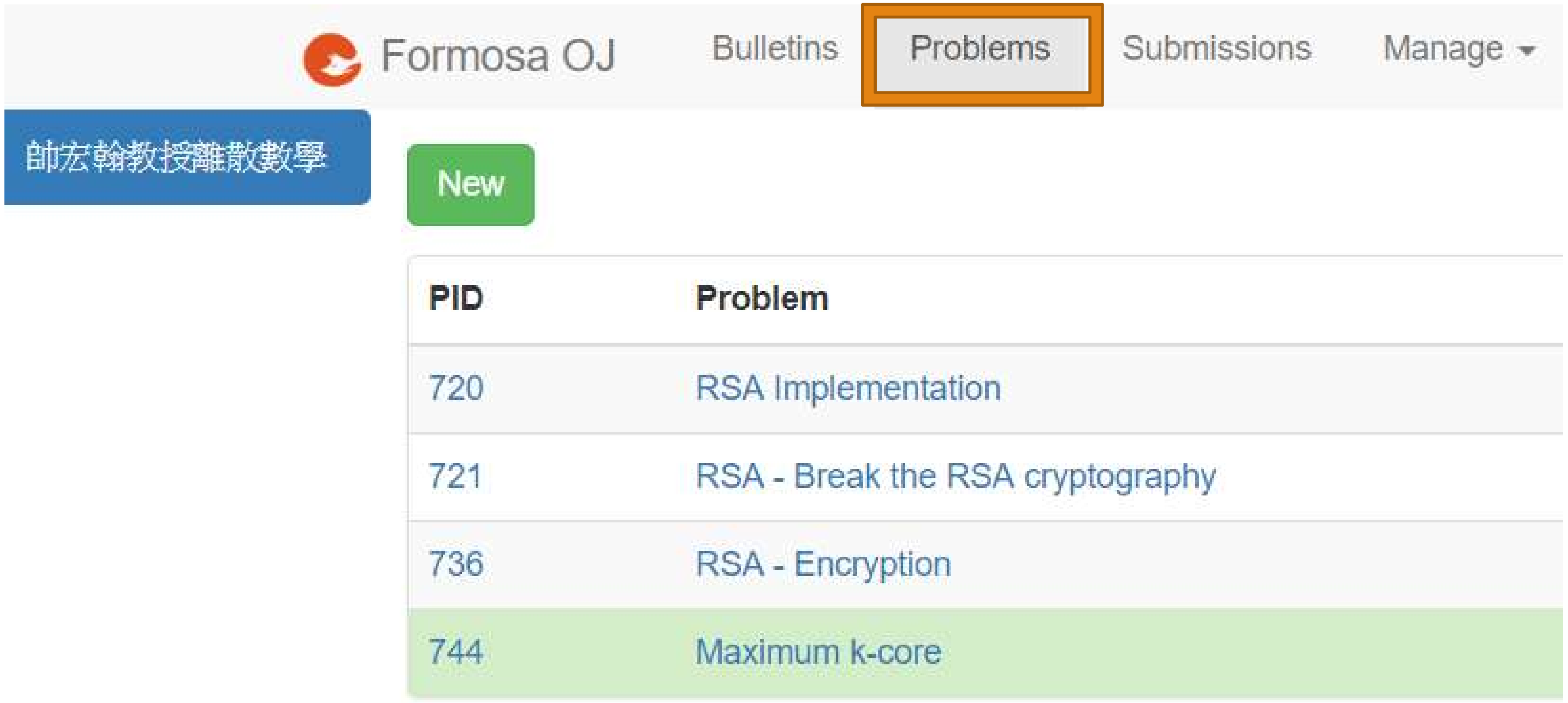
Apply “帥宏翰教授離散數學”


ormosa OJ

#	Name	Type	Apply
1	Public	Public	Join
2	NCTU PCCA	Public	Join
3	黃琴雅教授	Closed	Apply
4	蔡錫鈞教授演算法概論2017年秋季班	Closed	Apply
5	童莉萍教授計算機概論與程式設計	Closed	Apply
7	謝旻錚教授競技程式設計(二)	Public	Join
8	Introduction to Algorithms 2017 Fall	Public	Join
9	孫春在教授計算機概論與程式設計	Closed	Apply
	帥宏翰教授離散數學	Closed	Enter
11	謝旻錚教授競技程式設計(一)	Public	Join
[illegible]	[illegible]	[illegible]	[illegible]

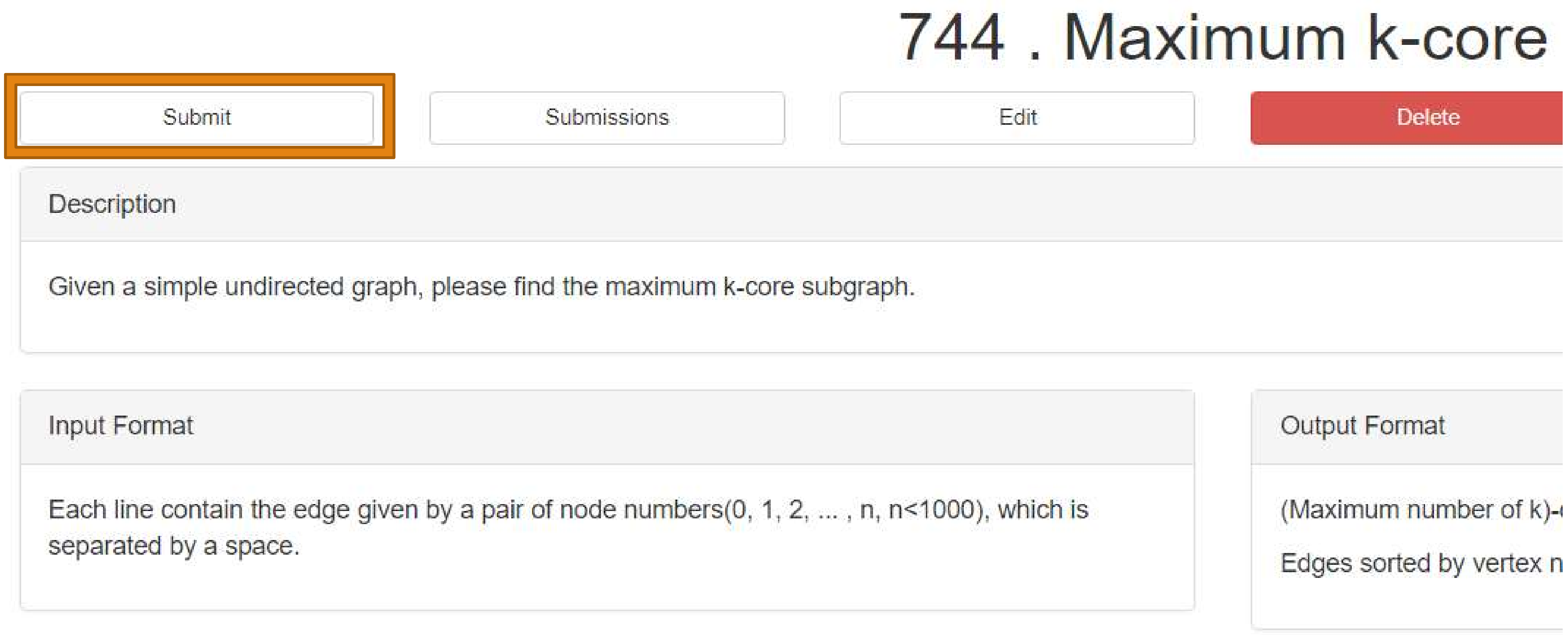
Formosa OJ

Link: https://oj.nctu.me/groups/10/problems/



PID	Problem
720	RSA Implementation
721	RSA - Break the RSA cryptography
736	RSA - Encryption
744	Maximum k-core

Formosa OJ



744 . Maximum k-core

Description

Given a simple undirected graph, please find the maximum k-core subgraph.

Input Format

Each line contain the edge given by a pair of node numbers(0, 1, 2, ... , n, n<1000), which is separated by a space.

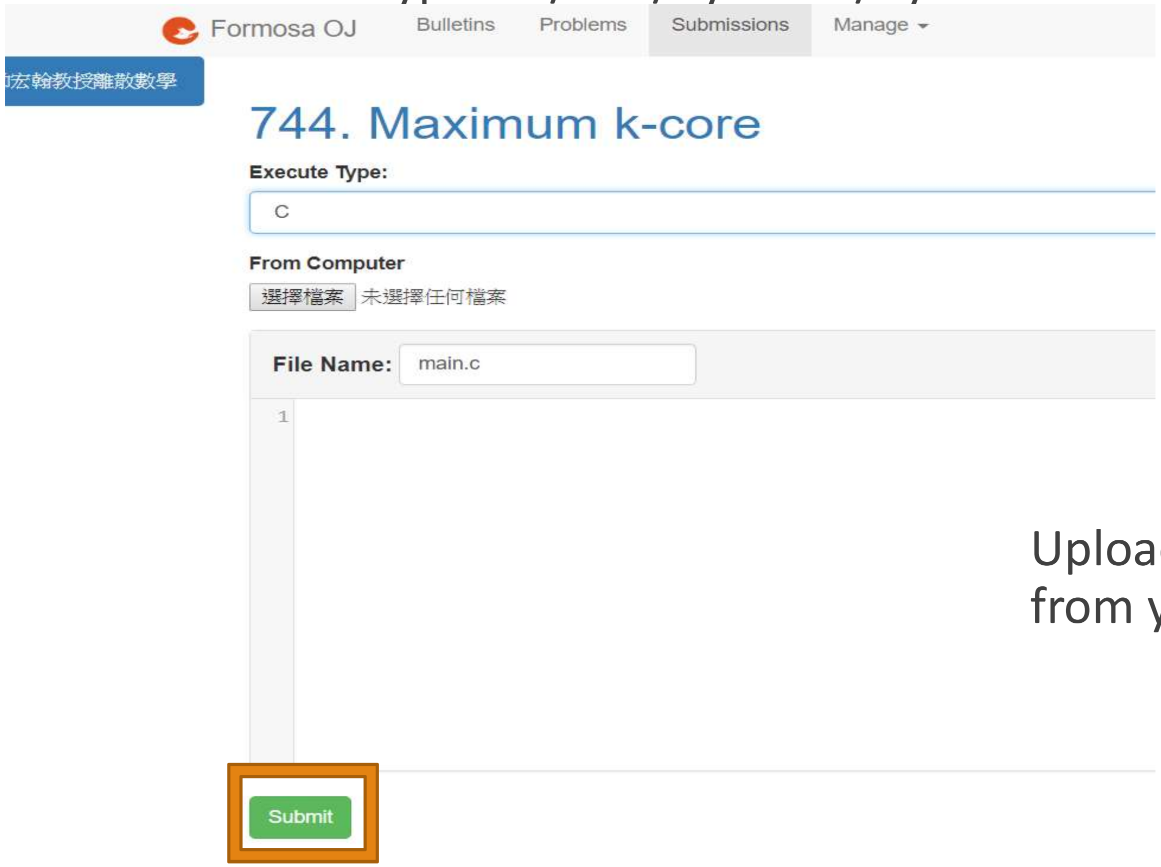
Output Format

(Maximum number of k)-

Edges sorted by vertex n

Formosa OJ

Select “Execute Type”: C/C++/Python2/Python3



744. Maximum k-core

Execute Type: C

From Computer

選擇檔案 未選擇任何檔案

File Name: main.c

Upload your source code from your computer

Formosa OJ - Verdict

- AC: Accepted (答對)
- WA: Wrong Answer (答案錯)
- CE: Compilation Error (編譯時錯誤)
- RE: Runtime Error (執行時錯誤)
- TLE: Time Limit Exceeded (超過時間限制)
- MLE: Memory Limit Exceeded (超過記憶體限制)
- SE: System Error (系統錯誤)

Formosa OJ – IO (Python3) Case 1

Input:

77 7 HI

Python 3:

```
s1 = input()                 #s1='77 7 HI'
s2 = input().split('  ')   #s2[0]='77' s2[1]='7'
                             #s2[2]='HI'
```

Output:

30,17

Python 3:

```
c = list[30,17]
code = ','.join([str(integer) for integer in c])
print(code)
```

Formosa OJ – IO (C) Case 1

Input:

77 7 HI

Output:

30,17

C:

```
int n1, n2;
char s[30];
scanf("%d%d%s", &n1, &n2, s);
//n1=77, n2=7, s=HI
```

C:

```
int n1=30, n2=17;
printf("%d,%d", n1, n2);
```

Formosa OJ – IO (C++) Case 1

Input:

77 7 HI

Output:

30,17

C++:

```
int n1, n2;
string s;
cin >> n1 >> n2 >> n3; //n1=77, n2=7, s=HI
```

C++:

```
int n1=30, n2=17;
cout << n1 << “,” << n2 << endl;
```

Formosa OJ – IO (Python3) Case 2

Input:

```
0 1
0 2
1 2
1 3
……
```

Python 3:

```
import sys
for line in sys.stdin:
        a=line.split(' ')
        n1=int(a[0])
        n2=int(a[1])
```

Formosa OJ – IO (C) Case 2

Input:

```
0 1
0 2
1 2
1 3
……
```

C:

```
int n1, n2;
while(scanf("%d%d", &n1, &n2)!=EOF){
      ……
}
```

Formosa OJ – IO (C++) Case 2

Input:

```
0 1
0 2
1 2
1 3
……
```

C++:

```
int n1, n2;
while(cin >> n1 >> n2){
    ……
}
```

Project Candidates: #1

RSA - Encryption

- The objective is to use Public key (n,e) and message (m) to compute ciphertext (c).

RSA - Break the RSA

- The objective is to use Public key (n,e) and ciphertext (c) to break RSA cryptography and compute message (m).

Tools are allowed BUT you need to implement your own modular exponentiation function (x^y mod n).

No plagiarism.

EXAMPLE

- $p = 61, q = 53$
- $n = 61 \times 53 = 3233$
- $\varphi(n) = 60 \times 52 = 3120$
- $e = 17 \Rightarrow ex + \varphi(n)y = 1$
- $17 \times 2753 - 3120 \times 15 = 1$
- $A = 123, \boxed{A^{17} \equiv 123^{17} \equiv 855 \ (mod\ 3233)}$ => encrypt
- $\boxed{855^{2753} \equiv 123 (mod\ 3233)}$ => decrypt

Public key: (n,e)
Private key: (n,d)

gcd(17,3120)=1
=> e = 17
d = inverse of 17 mod 3120
=> d = 2753

Description(RSA - Encryption)

Use Public key (n,e) and message (m) to compute ciphertext (c).

Input:

A test case consists of one line, which contains two integers n, e, and a string of the message (one word without any space).

Output:

Your program must produce a single line, containing ASCII code of the message separate by comma (,).

Format (RSA - Encryption)

Input: (n e m)

77 7 HI

Output: (c)

30,17

Solve:

H = 72 ; I = 73

72^7%77 = 30

73^7%77 = 17

Testdata (RSA - Encryption)

Number of test cases: 8

Time Limit: 1000ms

Per test case = 3% ; All 8 test cases = 24%

Description(RSA - Break the RSA)

Use Public key (n,e) and ciphertext (c) to break RSA cryptography and compute message (m).

Input

A test case consists of one line, which contains two integers n, e, and a string of the ciphertext (numbers separated by comma).

Output

Your program must produce a single line, containing original message.

Format (RSA - Break the RSA)

Input: (n e c)

221 5 89,99

Output: (m)

HI

Testdata (RSA - Break the RSA)

Number of test cases: 4

Time Limit: 10000ms

Per test case = 6% ; All 4 test cases = 24%

Speed of Breaking Encryption(12%)

- Top 25% : 12%
- Top 50% :   9%
- Top 75% :   6%
- The rest :   3%

Grading policy (RSA)

Correctness (48%):
- Encryption: 24%
- Breaking Encryption: 24%

Speed of Breaking Encryption(12%)
- Top 25% : 12%
- Top 50% :   9%
- Top 75% :   6%
- The rest :   3%

Report (40%)
- English/Chinese
- Novelty
- Comprehensiveness of experiments
- Theoretical results

Project Candidates: #2

Graph problem-- Maximum k-core problem

- A k-core of a graph G is a maximal subgraph of G in which all vertices have degree at least k.
- Find the maximum k-core in the simple graph G.

- Tools are allowed while existing source codes are forbidden.
- Packages for graph or network are also forbidden

  (Ex. NetworkX).
- No plagiarism.

Format (Maximum k-core)

Input(graph):	Output(maximum k-core):
0 1	3-core
0 2	1 2
1 2	1 4
1 4	1 5
1 5	2 4
2 3	2 5
2 4	4 5
2 5	
4 5	

Example

input:

0 1

0 2

1 2

1 4

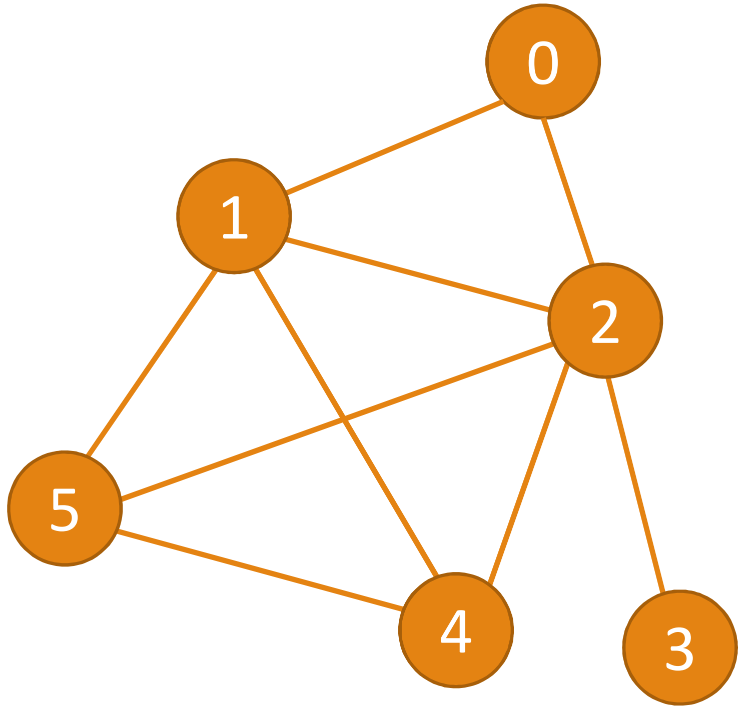
1 5

2 3

2 4

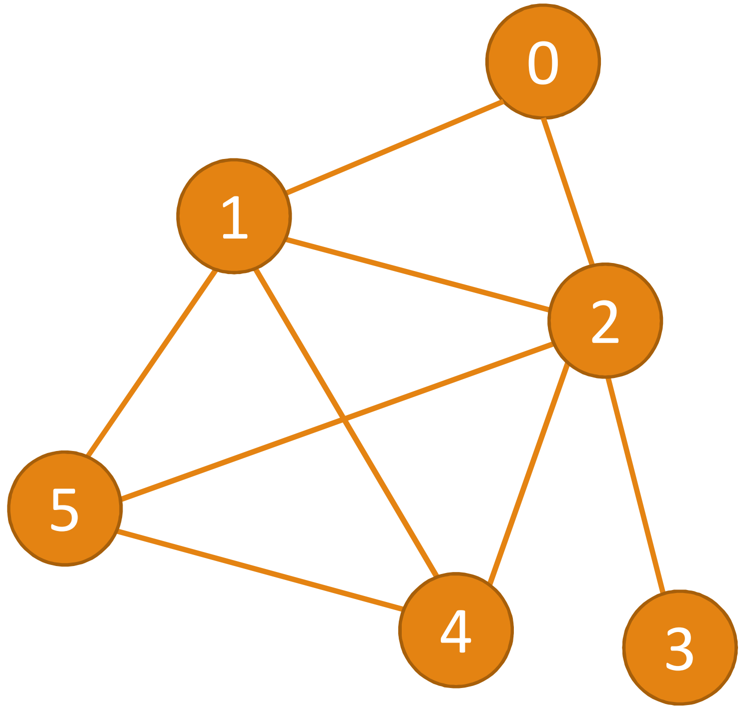
2 5

4 5



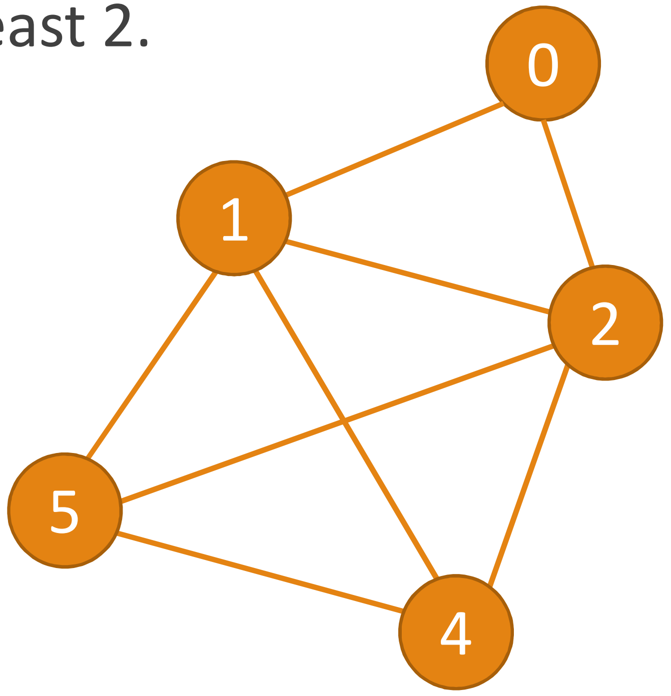
Example(create 2-core graph)

Remove the vertex with degree 1.



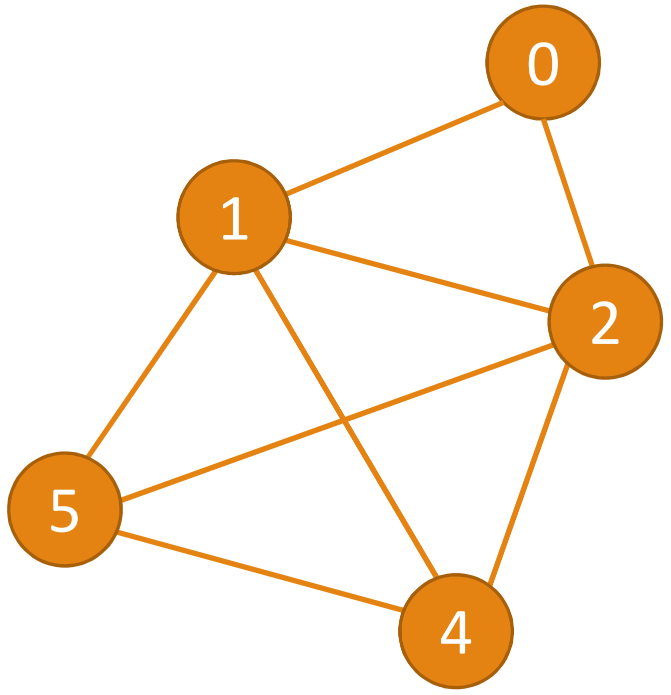
Example(2-core graph)

All vertices in 2-core graph
have degree at least 2.



Example(create 3-core graph)

Remove the vertex with degree 2.

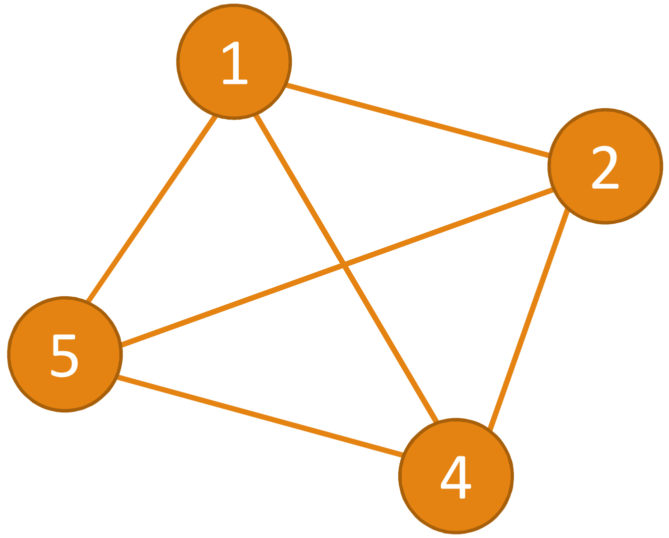


Example(3-core graph)

All vertices in 3-core graph

have degree at least 3.

The maximum k-core is 3-core.



Test data

Number of test cases: 10

Time Limit: 4000ms

Memory Limit: 1,000,000 (KiB)

4% for each test case, all 10 test cases = 40%

Grading policy

1. Correctness (40%)

2. Speed (20%):
- Top 25%: 20%
- Top 50%: 15%
- Top 75%: 10%
- The rest: 5%

3. Report (40%)
- English/Chinese
- Novelty
- Comprehensiveness of experiments
- Theoretical results

Deadline

1/16 23:59 | Formosa OJ關閉

1/17 23:59 | 繳交report與程式檔 (E3)