


# 演算法與資料結構（一）

講者：大叫

# 自我介紹

- 綽號：大叫 🐼
- 成大資工 大二（特選甲組）
- SCIST  演算總召
- 程式設計一 助教
- CPE 6 / 2403 名
- ICPC 外站（橫濱、河內）
- 兩次 YTP 少年圖靈計畫 專題指導資格



少年圖靈計畫  
Young Turing Program



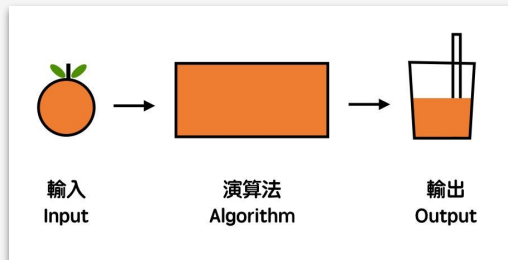
# 目錄

- 演算法與資料結構
- OJ
- 時間複雜度
- 枚舉

# 演算法與資料結構

# 演算法與資料結構

- 什麼是演算法？
- 將「輸入」轉變為「輸出」的一連串計算步驟
- 朋友問你怎麼去火車站
- 你打開 google 地圖 -> 搜尋 -> 告訴朋友怎麼走
- 你就是一個演算法！
- 一個好的演算法，能在達到同樣效果的前提下，使用更少的資源和時間。



# 演算法與資料結構

- 什麼是資料結構？
- 我們在生活中，會把東西分類放好。
- 圖書館的書會根據類別和編號排序。
- 資料結構就是**組織和儲存資料的方式**。它決定了資料如何被放置，以及如何被存取和操作。
- 在設計演算法時，搭配資料結構是非常重要的一環。

# 演算法與資料結構

- 學好演算法和資料結構有什麼好處？
- 常用且實用
- 相關競賽檢定多 -> 競程

# 競程介紹

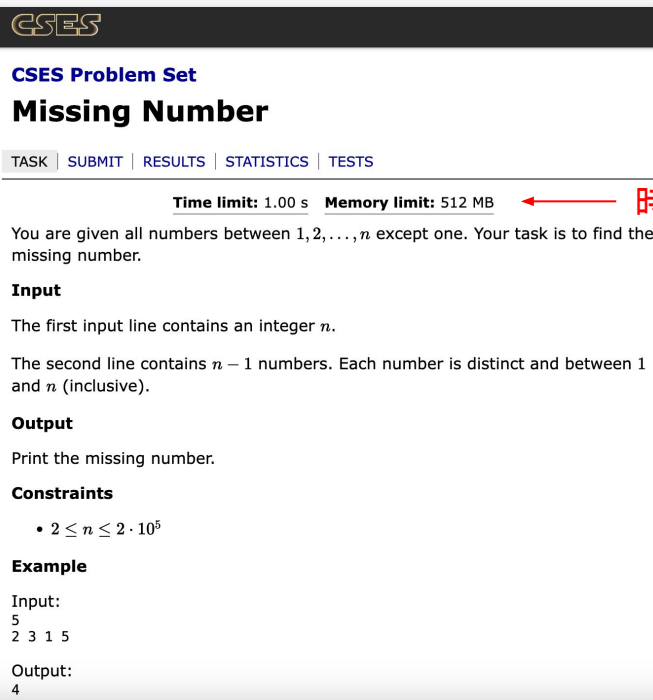
- 競程是什麼？
- 也叫競技程式、演算法競賽
- 是資訊領域的一個分支，有很多對高中升學有幫助的比賽和檢定都是競程形式的
- 例如 TOI、學科能力競賽、APCS、成大暑期高中生邀請賽、YTP 等
- 一場通常有數道題目
- 評分指標：題數/速度





- 題目架構

題目敘述



The screenshot shows the CSES Problem Set page for 'Missing Number'. It includes navigation links (TASK, SUBMIT, RESULTS, STATISTICS, TESTS), time and memory limits (1.00 s, 512 MB), a problem description, input/output specifications, constraints, and an example.

**CSES Problem Set**  
**Missing Number**

[TASK](#) | [SUBMIT](#) | [RESULTS](#) | [STATISTICS](#) | [TESTS](#)

**Time limit:** 1.00 s **Memory limit:** 512 MB

You are given all numbers between  $1, 2, \dots, n$  except one. Your task is to find the missing number.

**Input**

The first input line contains an integer  $n$ .

The second line contains  $n - 1$  numbers. Each number is distinct and between 1 and  $n$  (inclusive).

**Output**

Print the missing number.

**Constraints**

- $2 \leq n \leq 2 \cdot 10^5$

**Example**

Input:  
5  
2 3 1 5

Output:  
4

時間、空間限制

輸入、輸出說明與限制

範例

# 競程介紹

- 如何練習和驗證學習成效？
- 就在 OJ (Online Judge) 上拿到一發 AC 吧！

# Online Judge

- OJ (Online Judge) 是什麼？
- 線上解題/評測系統
- 裡面有許多題目，可上傳自己的解並即時得到回饋
- 本課程會用到的 OJ：
  - CSES
  - CodeForces
  - ZeroJudge

- 讀懂題目後寫一寫

**CSES**

**CSES Problem Set**

**Missing Number**

TASK | SUBMIT | RESULTS | STATISTICS | TESTS

**Time limit:** 1.00 s **Memory limit:** 512 MB

You are given all numbers between  $1, 2, \dots, n$  except one. Your task is to find the missing number.

**Input**

The first input line contains an integer  $n$ .

The second line contains  $n - 1$  numbers. Each number is distinct and between 1 and  $n$  (inclusive).

**Output**

Print the missing number.

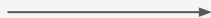
**Constraints**

- $2 \leq n \leq 2 \cdot 10^5$

**Example**

Input:  
5  
2 3 1 5

Output:  
4

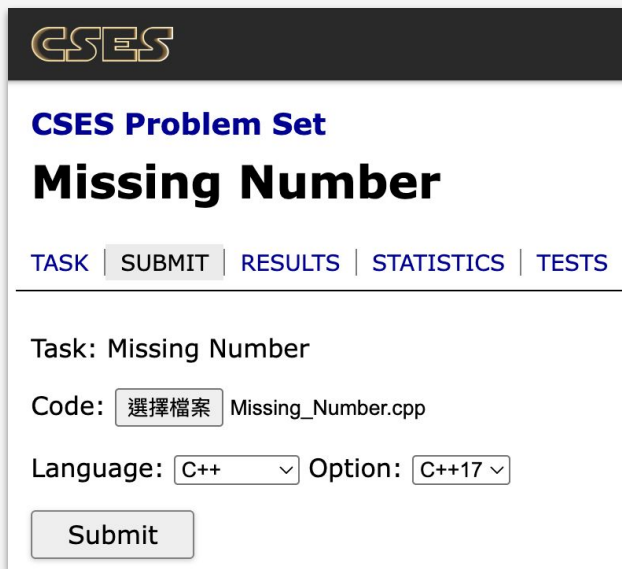


```
#include<bits/stdc++.h>
using namespace std;

const int MAX_N = 2e5+5;
int arr[MAX_N];

int main() {
    int n;
    cin >> n;
    for(int i=1;i<n;i++) {
        int x;
        cin >> x;
        arr[x] = 1;
    }
    for(int i=1;i<=n;i++) {
        if(!arr[i]) {
            cout << i << "\n";
            return 0;
        }
    }
}
```

- 按下 Submit 上傳



The screenshot shows the CSES (Competitive Programming) website interface for the 'Missing Number' problem. At the top is the CSES logo. Below it, the title 'CSES Problem Set' is in blue, followed by the problem name 'Missing Number' in large black font. A navigation bar contains links: TASK, SUBMIT (highlighted), RESULTS, STATISTICS, and TESTS. The task description area shows 'Task: Missing Number'. The 'Code:' label is followed by a file selection button labeled '選擇檔案' and the filename 'Missing\_Number.cpp'. The 'Language:' dropdown is set to 'C++' and the 'Option:' dropdown is set to 'C++17'. A 'Submit' button is at the bottom.

CSES

CSES Problem Set

## Missing Number

TASK | SUBMIT | RESULTS | STATISTICS | TESTS

Task: Missing Number

Code: 選擇檔案 Missing\_Number.cpp

Language: C++ Option: C++17

Submit

- 提交後可能遇到的狀況
- AC(Accepted)：輸出正確
- WA(Wrong Answer)：輸出錯誤
- TLE(Time limit exceed)：執行時間超過限制
- MLE(Memory limit exceed)：使用過多記憶體
- RE(Runtime error)：比較可能是戳到陣列外的空間
- CE(Compile error)：你的程式無法正常編譯

- 接下來請各位試著成功上傳一份程式到

## CSES : Missing Number 上

### CSES Problem Set

### Missing Number

[TASK](#) | [SUBMIT](#) | [RESULTS](#) | [ANALYSIS](#) | [STATISTICS](#) | [TESTS](#) | [QUEUE](#) | [SOLUTION](#)

#### Submission details

Task:	<a href="#">Missing Number</a>
Sender:	sakinu9487
Submission time:	2025-06-28 21:00:06 +0300
Language:	C++ (C++17)
Status:	READY
Result:	ACCEPTED

#### Test results ▲

test	verdict	time	
#1	ACCEPTED	0.00 s	»
#2	ACCEPTED	0.00 s	»
#3	ACCEPTED	0.00 s	»
#4	ACCEPTED	0.00 s	»
#5	ACCEPTED	0.00 s	»
#6	ACCEPTED	0.01 s	»
#7	ACCEPTED	0.01 s	»
#8	ACCEPTED	0.04 s	»

#### Tags

► Show Problem Tags

#### Tips

► Show Tip 1

#### Introductory Problems

Weird Algorithm	✓
Missing Number	✓
Repetitions	✓
Increasing Array	✓
Permutations	✓
Number Spiral	✓
Two Knights	✓
Two Sets	✓
...	

#### Your submissions

2025-06-28 21:00:06	✓
---------------------	---

← Your submissions 下要有東西 !



# 時間複雜度

# 時間複雜度

- 維基百科的定義：

在電腦科學中，演算法的時間複雜度 ( time complexity ) 是一個函數，它定性描述該演算法的執行時間。

- 衡量程式執行時間的一個指標
- TLE 的原因
- $O(N)$ ， $N$  代表資料量的大小， $O$  代表這是考慮最差狀況
- 一個原因是，一個正常的題目應該在符合輸入限制條件的同時讓提交的程式TLE

# 時間複雜度

- 那要如何計算一個程式的複雜度呢？
- 最簡單的方式：數迴圈

```
int main(){  
    int n;  
    cin >> n;  
    for(int i=1;i<=n;i++){  
        cout << "?\n";  
    }  
}
```

$O(n)$

```
int main(){  
    int n;  
    cin >> n;  
    int x;  
    for(int i=1;i<=n+5;i++){  
        cin >> x;  
        cout << x;  
    }  
}
```

$O(n)$

- 為何兩者都是  $O(n)$ ？

# 時間複雜度

- 那要如何計算一個程式的複雜度呢?
- 最簡單的方式：數迴圈

```
int main(){
    int n,m;
    cin >> n >> m;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cout << "?\n";
        }
    }
}
```

$O(nm)$

```
int main(){
    int N;
    cin >> N;
    while(N){
        N>>=1;
    }
    cout << "?\n";
}
```

$O(\log N)$

# 時間複雜度

- 還有很多時間複雜度的估算方式，例如均攤分析、主定理等等，但這邊不會講到，有興趣者可以自行研究
- OJ 一秒可以跑多少操作？

# 枚舉

# 枚舉目錄

- 枚舉介紹
- 減少枚舉數量
- 全排列枚舉
- 位元枚舉
- 折半枚舉
- 課後練習題

# 枚舉介紹

- 什麼是枚舉？
- 簡單來講，枚舉就是考慮各種可能情況來得到答案的技巧
- 基本的技巧，也能許多變化。



# 例題

- 有兩疊張數不超過10的牌堆A、B，每次操作可選擇以下一種執行
    - 1.從 A 中抽兩張卡，並從 B 中抽一張牌
    - 2.從 A 中抽一張卡，並從 B 中抽兩張牌
  - 求兩種操作各做幾次才能將兩疊卡片同時抽完？保證卡牌總和為3的倍數
- 
- 最簡單的方式就是直接枚舉操作 1 和操作 2 要做幾次
  - [解說](#)

# 減少枚舉數量

- 剛剛對程式的優化，就是在減少枚舉的數量
- 再來一題

# 減少枚舉數量

- 給一正整數  $X$ ，問  $X$  是否是質數
- $X \leq 10^{14}$
- 如果像右邊的程式一樣直接枚舉  $2 \sim X-1$ ，檢查其能否整除  $X$ ，最大到  $10^{14}$  的計算量 how to AC
- 此時就可以利用數學性質，只檢查到  $\text{sqrt}(X)$  來有效減少需要枚舉的量，以  $10^7$  量級的計算量避免 TLE 的未來

```
int main() {
    long long x;
    cin >> x;
    for(long long i=2; i<x; i++) {
        if(x%i == 0) {
            cout << "No\n";
            return 0;
        }
    }
    cout << "Yes\n";
    return 0;
}
```

# 全排列枚舉

- 顧名思義，就是枚舉所有排列來得到答案(例如數列) 複雜度 $O(n!)$
- `next_permutation(first, last)`，會回傳是否存在、並把 `[first,last)` 之間的東西變成下一個排列
- `prev_permutation(first, last)`，則相反，變為上一個排列

```
int main(){
    int arr[3];
    for(int i=0;i<3;i++) arr[i]=i;    //arr={0,1,2}
    do{
        for(int i=0;i<3;i++) cout << arr[i];
        cout << "\n";
    }while(next_permutation(arr,arr+3)); //只要還有後一個排列就重複
}
```

```
012
021
102
120
201
210
```

# 全排列枚舉

- 練習題：[e446. 排列生成](#)
- 練習題：[d299. 程式設計師的面試問題](#)

# 位元枚舉

- 利用二進位的特性來枚舉所有狀態
- 例如決定要不要買某兩個東西時，可以用00 01 10 11來枚舉所有買與不買的組合
- 枚舉複雜度 $O(2^n)$

# 位元枚舉

- 練習題：[CSES 1623 Apple Division](#)
  - 給 $n$ 顆蘋果，問若將其分成任意兩堆，最小的重量差
  - $1 \leq n \leq 20$
- 
- 觀察到蘋果只有在與不在其中一堆的可能
  - 從0枚舉到 $2^{20}-1$ ，判斷當前狀況兩邊的重量差
  - [範例](#)

# 折半枚舉

- 聽名字就可以想像，折半枚舉在做的就是東西折一半再枚舉
- 位元枚舉在  $n > 25$  左右時就會很容易 TLE，但若拆成兩半，複雜度就會變成  $O(2^{(n/2)} + \text{從兩邊枚舉出的東西得出答案的複雜度})$



# 折半枚舉

- 給 $n$ 顆蘋果，問若將其分成任意兩堆，最小的重量差
  - $1 \leq n \leq 40$
- 從上一題做變化，想像成兩個 $n \leq 20$ 的狀況，記錄下所有可能出現的重量
  - 利用一些方法從其中一半，快速的找到另一半最適合的配對(例如這題就是找與自己大小最近的重量)
  - [範例](#)

# 課後練習題

- ABC 196C Doubled
  - 給一整數 $n$ ，並詢問在 $[1 \sim n]$ 之間有多少長度為偶數的整數(無前導0)，其左半部分與右半部分相同
  - $1 \leq n < 10^{12}$
- [提示](#)

# 課後練習題

- ABC 100D Patisserie ABC
  - 有N個蛋糕，每個蛋糕有三個參數 A、B、C，問從中選M個蛋糕使  $|A\text{的和}| + |B\text{的和}| + |C\text{的和}|$  的最大值是多少
  - $1 \leq n \leq 1000$ 、 $0 \leq m \leq n$ 、 $-1e10 \leq A、B、C \leq 1e10$
- 
- 這題的難點是，由於最終選的是絕對值，所以我們無從判斷選某個蛋糕對答案的影響是好是壞
  - [提示](#)

# 課後練習題

- [CF 1494B. Berland Crossword](#)
- 詢問t次，每次給一個n\*n個小方塊組成的方形拼圖，問有沒有可能上下左右四邊被塗色的拼圖與輸入相同
- $1 \leq t \leq 1000$ 、 $1 \leq n \leq 100$
- 這和位元枚舉有什麼關係？
- [提示](#)

# 課後練習題

- [CF 1554B. Cobb](#)
  - 給長度為 $n$ 的數列 $A$ 與整數 $k$ ，設  $f(i,j)$  為  $i*j - k*(a[i] \mid a[j])$ ，求對於所有  $1 \leq i < j \leq n$  中最大的  $f()$ 。
  - $2 \leq n \leq 1e5$ 、 $1 \leq k \leq \min(n, 100)$ 、 $0 \leq A$  中的數  $\leq n$
- 
- 不得不說，這題蠻難的！！！！
  - 如果直接枚舉是 $O(n^2)$ ，有沒有什麼好方法減少枚舉的範圍？
  - [提示](#)

# 如果想這方面變強...

- 學習資源有營隊、社團、社群、課程、網路教材等等
- 練習方式：刷題
  - 知道如何對剛學會的演算法做變化
  - 先實際碰過一些坑，避免賽中燒雞

- 例如：
- 對前面的內容有哪裡想問？
- 其他...

- 演算法與資料結構
- OJ
- 時間複雜度
- 枚舉
  - 減少枚舉次數
  - 位元枚舉
  - 全排列枚舉
  - 折半枚舉



# 謝謝各位

講者：大叫