Chapter 2.
複雜度分析
Complexity Calculate

台南女中資訊研究社 38th C++ 進階班課程

TNGS IRC 38th C++ Advanced Course

講師：陳俊安 Colten

一切的起點 - 時間複雜度

- 一個可以用來估計程式效率的指標
- 你可能常常會看到有人說他的時間複雜度是 O(N)、O(NM)
- 這一個 O 的意思其實是表示 "漸進時間複雜度" 的意思
- 通常我們打競程所估的時間複雜度都是以最壞情況下為主
- 而 O 括號裡面數字越大，則表示效率越差

為什麼都要以最壞的情況下為主

- 通常一個 "正常" 的題目都會給你題目範圍
- 而一個 "正常" 的出題者會在測試資料中加入一些極限測資
- 因此我們在估計的時候當然也要考慮到最極限的狀況

如果不正常的題目或比賽怎麼辦

- 自己通靈
- 對，真的是這樣
- 如果有裁判的話可以問裁判
- 但通常你不會得到你想要的回覆的
- 競程界有很多相關的傳奇故事，可以去打聽看看XD

1 秒能執行多少次

- 1 秒大約能執行 10^8 次左右
- 估完時間複雜度後把數字代進去看看
- 大家要有一個習慣，設計完一個演算法後都請先估計看看時間複雜度的好壞
- 必須要先對自己的程式執行的效率有一個底

時間複雜度最簡單的估計方式

- 算迴圈執行的次數
- 幾次就是多少
- O(N)

```
 5 int main()
 6 {
 7     int n;
 8     cin >> n;
 9
10     for(int i=0;i<n;i++)
11     {
12         cout << "Hello world!\n";
13     }
14
15     return 0;
16 }
```

時間複雜度最簡單的估計方式

- 算迴圈執行的次數
- 幾次就是多少
- O(NM)

```
int main()
{
    int n;
    cin >> n >> m;

    for(int i=0;i<n;i++)
    {
        for(int k=0;k<m;k++)
        {
            cout << "Hello world!\n";
        }
    }

    return 0;
}
```

時間複雜度最簡單的估計方式

- 算迴圈執行的次數
- 幾次就是多少
- O(logN)

```
 5 int main()
 6 {
 7     int n;
 8     cin >> n;
 9
10     while( n ≠ 0 )
11     {
12         n /= 2;
13     }
14
15     return 0;
16 }
```

時間複雜度最簡單的估計方式

- 通常我們會把一些不太重要的常數去掉
- O(N)

```
 5 int main()
 6 {
 7     int n;
 8     cin >> n;
 9     for(int i=0;i<n+5;i++)
10     {
11         cout << "Hello world\n";
12     }
13
14     return 0;
15 }
```

時間複雜度最簡單的估計方式

- 基本上沒執行幾次
- O(1)

```
 5 int main()
 6 {
 7     cout << "Colten\n";
 8
 9     return 0;
10 }
```

一些要記得的

- C++ 內建的 std::sort 的時間複雜度是 O(NlogN)
- O(N/1 + N/2 + N/3 + ⋯ + N/N) 約等於 O(NlogN)
    - 這個是調和級數，有興趣的可以參閱維基百科
    - 要證明的話要用 積分 : (

空間複雜度

- 跟時間複雜度一樣，我們通常都探討最壞的情況
- 時間複雜度探討的是效率、而空間複雜度探討的是記憶體的使用量

空間複雜度

- 通常會大量使用到空間的地方是陣列或一些 STL 資料結構
- 最簡單的估計方式是看看這一些東西塞到全滿的狀態會需要多少數字的空間
- 影響空間複雜度的因素其實還有型態 (int, long long, double)