Time Complexity

時間複雑度

時間複雜度 – 定義

- 描述一個演算法的效率的工具。

- 根據輸入規模，來表達演算法**需要執行的步驟的增長趨勢**。

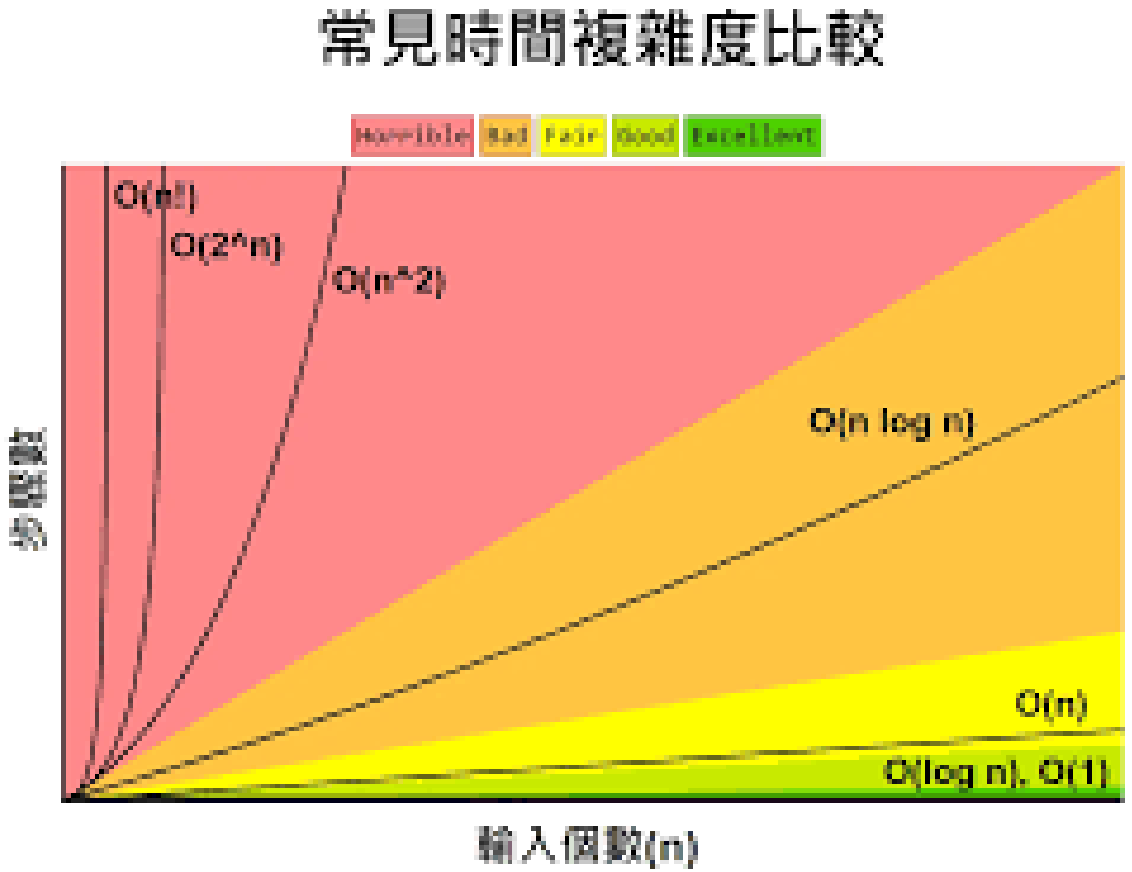
時間複雜度 – 含義

- O(n) 的複雜度 -> 要執行的步驟數和 n 呈現**近線性關係**
- O(n^2) 的複雜度 -> 要執行的步驟數和 n^2 呈現**近線性關係**

時間複雜度 – 簡化

- O(n) 的複雜度 -> 要執行的步驟數和 n 呈現**近線性關係**：
- 因此 O(2n + k) = O(n)，其中 k 是常數
- O(n^2) 的複雜度 -> 要執行的步驟數和 n^2 呈現**近線性關係**：
- 因此，O(n^2+n) = O(n^2)，因爲相比 n^2，n 太小了

時間複雜度 – 比較



- O(n) 線性時間。
- O(log(n)) 對數時間，注意，log 的底不重要（只是常數。
- O(1) 常數時間，無視輸入，是能實現的最小時間複雜度。

時間複雜度－計算（一）：計算最壞的情況

```
1   //A為長度n的陣列
2   for j = 2 to A.length do
3       data = A[j]
4       i = j-1
5       while i > 0 and A[i] > data do
6           A[i+1] = A[i];
7           i = i-1;
8       end while
9       A[j+1] = data;
10  end for
```

時間複雜度－計算（一）：計算最壞的情況

```
1   //A為長度n的陣列
2   for j = 2 to A.length do
3       data = A[j]
4       i = j-1
5       while i > 0 and A[i] > data do
6           A[i+1] = A[i];
7           i = i-1;
8       end while
9       A[j+1] = data;
10  end for
```

$$\sum_{j=2}^{n} O(j-1) = O(n^2)$$

O(j - 1) * O(1) = O(j - 1)

O(1)

時間複雜度－計算（二）

```
1  //A為長度n的陣列
2  for i = 0 to A.length do
3      t=a[i]
4      while t>=1 do
5          t/=2
6      end while
7  end for
```

時間複雜度－計算（二）

```
1  //A為長度n的陣列
2  for i = 0 to A.length do
3      t=a[i]
4      while t>=1 do
5          t/=2
6      end while
7  end for
```

O(nlogM)

O(logA[i]) = O(logM)

O(1)

時間複雜度 – 運用

電腦一秒鐘大概執行 5 * 10^8 個指令

因此，如果你的複雜度是 O(n)，n 最高可以到 10^8；
如果你的複雜度是 O(n^2)，n 最高只能到 10^4；
如果你的複雜度是 O(n^3)，n 最高只能到 500。

時間複雜度可以幫我們判斷，給定範圍內，你的演算法超時的可能性。
當然，雖然說時間複雜度不在乎常數，但我們實際上在計算的時候，還是儘可能的把會影響的常數列入考量。