<https://blog.csdn.net/NXHYD/article/details/71716539> (LinkedList reverse)

<https://github.com/apachecn/awesome-algorithm/tree/master/docs/Leetcode_Solutions/Python>

<https://noob.tw/data-structure/>

<https://github.com/apachecn/awesome-algorithm/tree/master/docs/Algorithm_Implementation/Python>

<https://github.com/apachecn/awesome-algorithm/tree/master/docs/%E5%89%91%E6%8C%87offer/Python>

data structrue

時間複雜度

一個程式的時間複雜度是指完全地執行程式所需的計算機時間。

如果一個演算法執行的步驟是固定的，無關輸入的值而改變，那我們會記成 O(1)，例如：

function(int n){

print(n);

}

不管 n 輸入多少，這個程式永遠只會執行一次。

而下面這個演算法：

function(int n){

for(i=0;i<n;i++){

print(i);

}

}

這個演算法則是依據輸入的 n 的數量會跑 n 次，所以是 O(n)。

空間複雜度

而一個程式的空間複雜度是指完全地執行程式所需的記憶體量。

所需的記憶體量，大概可以看成所用的變數量。

例如下面這個函式，不管程式跑了幾遍，都不會影響使用的變數數量：

function(int n){

for(int i=0;i<n;i++){

print(i);

}

}

故該函式的空間複雜度記做 O(1)。

但下面這個函式，會隨著丟進去的數字而影響變數的量，例如：

function(int n){

int c[n];

for(int i=0;i<n;i++){

c[i] = i;

}

}

丟進去 n，就換產生 n 個變數，故該函式空間複雜度為 O(n)。

時間複雜度標記法，也完全忽略了 I/O 處理和記憶體管理的問題。要是資料結構複雜一點、龐大一點，讀取資料就會變慢。

時間複雜度標記法，也完全忽略了程式語言特性和平台特性。平平同一個演算法，用 C 寫執行較快，用 Java 、 Python 寫執行較慢。因為後者的記憶體管理機制更加複雜，而且牽扯到系統運作架構。

時間複雜度標記法再怎麼不可靠，也比不上實作的不可靠。平平同一個演算法，不同人寫出來的程式碼，執行效率都不一樣，相差十倍都有可能。

<https://xiaozhuanlan.com/topic/1538476290> github登入