演算法 DIY Problem

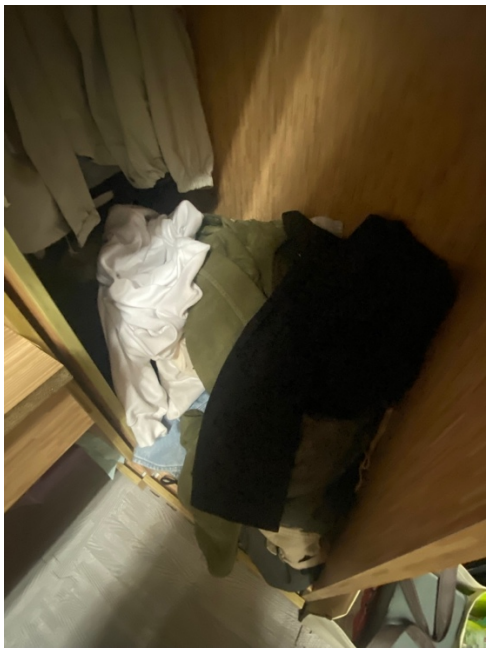
B09901062 黃宥翔

問題緣起

最近一個月來我一直在煩惱 DIY Problem 要做什麼題目，一直覺得要做一個很難很複雜的東西，直到有一天，我躺在衣櫃裡面，突然靈光一閃，發現其實用演算法去解決生活中的小困難也是一種方式，於是我就想到了這個題目：用 BFS 提升找衣服紀律性！

背景介紹

下面圖片是我現在衣櫃的樣子：

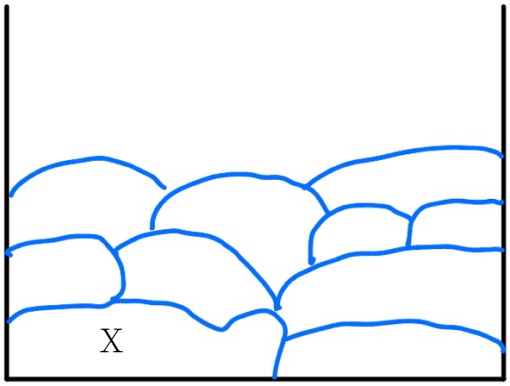


因為我懶惰，每次洗完衣服都隨手一丟，造成現在的窘境。這樣會造成的問題是，每次早上醒來，因為室友們都還在睡覺，燈都還沒開（如右圖），所以我要在看不見衣服們的情況下找到我想要的，也就是說我要一個一個拿出來 check，

還有一個問題是，因為從圖片可見，我的衣服的堆疊情形加上衣櫥的設計，導致我每次只能拿最上面的來 check。最後就是因為我每次都不假思索地拿，所以我目前的找法是看完一件就往衣櫥旁邊丟，十分沒有紀律性也沒有效率，常一直找到重複的，就導致我每天早上都很煩躁。

問題定義

下圖是我衣櫃的橫頗面示意圖：



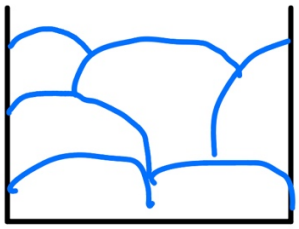
我每天早上的目標是找到某件衣服 X，而且根據上述背景介紹，每次只能找最上面的幾件衣服而已，那我要如何有規律且有效率地找到我要的衣服呢？

演算法選擇

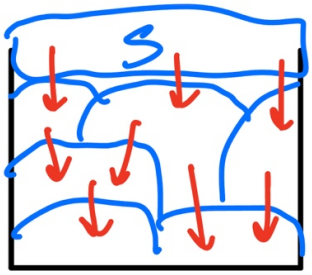
根據上面的問題定義，我發現課堂中教過的一個演算法非常適合解決這個問題，那就是 BFS。只能找最上面幾件衣服就像是 BFS 中的 Frontier 一樣，沒有把衣服拿起來（等價於 BFS 中的 color = BLACK）就不會搜到後面的東西。而為什麼不用另一個 Search Algorithm DFS 呢？其實是因為如果要用 DFS 的話我發現會有諸多限制，譬如說重力，我會在下個作出解釋。

演算法細節

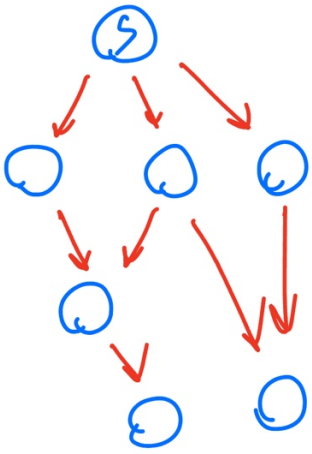
首先我要先做出 Graph，在這個演示中，我選擇一個 minor case 如下：



首先，根據問題，我每次只能看到最上面裸露出來的衣服，所以我把衣服有上下交疊的部分畫上箭頭。至於最上面和空氣相接的部分，則是補上一個 source node，並且畫上箭頭，結果就如下圖所示：



接下來，我把每件衣服視為一個 node，把他展開來後，就成了一個 Directed Graph：



從上圖可知，如果我對這樣的結構採用 DFS，那當我走到最下面的 node 的時候，會有上面的 node 還沒被搜到（或是在這個問題中，被拿起來）就會導致衣服（或 node）會因重力往下掉，改變問題的構造，這是我們所不樂見的，因此我最後選擇使用 BFS。

下面是我演算法的 pseudo code：

```
Cloth Finding (G, s):
    s.color = GRAY
    for each vertex u ∈ G.V - {s}
        u.color = WHITE
    Q = ϕ
    Enqueue (Q, s)  //Frontier
    while (Q ≠ ϕ)
        u = Dequeue (Q)
        for each vertex v ∈ G.Adj[u]
            if (v.color = WHITE)
                v.color = GRAY

                Check if v is the targeted cloth ? ?
                ⇒ if true, end algorithm

                Enqueue (Q, v)
    u.color = BLACK
```

在 pseudo code 中，node 的 color，GRAY 表示被看到，BLACK 則表示已經拿起來，並把他的 neighbor 都看過（標為 GRAY）。

實際使用

實際使用這個演算法的情形大致如下，我先走到 source node 把 neighbor 也就是最上面幾件衣服都看過(標為 Gray，Enqueue)，沒找到就拿最上面隨便一件(Dequeue)，把他拿起來(標為 BLACK)並把他周遭的衣服(white node)都看過並標為 GRAY(Enqueue)，沒找到就找和 source node 接壤的下一件(Dequeue)，拿起來後一樣 check 周遭，標為 GRAY，Enqueue，然後把一件看過的(標為 GRAY 的)的衣服拿起來（Dequeue 然後成為 BLACK），把他周遭 check……。

就是這樣的不斷輪迴，直到找到 X，就可以有規則的找到我要的衣服了！

時間複雜度

這個演算法的時間複雜度就和 BFS 一樣式 O(V+E)！