Algorithm PA3 Report

b06702064 / 會計五 / 林聖硯
EDA Union: Port 40055

一、不同 case 之 runtime 以及記憶體使用量比較

Case	Runtime (ms)	Memory usage (KB)
public_case_1.in	2.998	12,684
public_case_2.in	2.998	12,680
public_case_3.in	4.998	12,680
public_case_4.in	1.999	12,684
public_case_7.in	3.998	12,684
public_case_8.in	6.998	12,820

二、程式設計概念

1. undirected graph

在 undirected graph 的部分，由於我們最後要找的是「能用最少的 cost 讓 graph acyclic 的 edges set」，所以這個問題其實可以看成一個 maximum spanning tree 的問題。在 implementation 的細節上，我的程式會先判斷如果這個 graph 的 edge weight 的每一個是否皆為"1"。如果不是(代表這個 graph 是 weighted 的)，就會做 counting sort，來讓 edges 從"大"排到"小"。選擇 counting sort 的原因是我們已經知道 edges weight 的上下限，用 counting sort 演算法可以再把時間複雜度壓得更低(只需要 theta(n+k))。接下來，我再使用 Kruskal algorithm(edges weight 從大排到小)來找出 maximum spanning tree。

2. directed graph

在 directed graph 的部分，我很直覺地想要使用前面的 counting sort 和 kruskal 演算法(已經寫好了，不想再寫更多新的…)。所以我先把 directed graph 當成 undirected graph 來做，一樣先找出 undirected graph 裡面的 maximum spanning tree。題目只需要我們找出 weakly-connected component，所以在 undirected 上找 MST 一定符合這個條件。接下來，我將剩下沒有用到的 edges 依照 weight 從大排到小，並且一條一條加入 graph 裡面測試，使用 DFS 判斷整個 graph 會不會有 cycle。如果有就不加，沒有則加進去。我還寫了一個條件判斷，如果剩下來的 edge weight 是負數，則我就直接不加進去。因為我們想要找 maximum spanning tree，所以加入這個負的 edges 一定只會讓 MST 的 total cost weight 下降(剩下的 edges set 的 total weight 上升)，故顯而易見地我們可以將他排除再外。

三、觀察與發現

一開始看到 cycle breaking on directed graph 是一個 NP-hard 的問題，想說一定要用一個很複雜的演算法去解他。後來想一想，其實只需要沿用之前 undirected graph 的結果，再結合 greedy algorithm 的想法，應該也能找出不錯的答案。只是很明顯地這個問題沒有 greedy choice 和 optimal substructure 的性質，所以找出來的答案應該不會是最佳解。