0819823陳子祈 演算法導論 HW8

1. 測試演算法正確性

建立一個Graph如下圖



一張含有 桌 的圖片

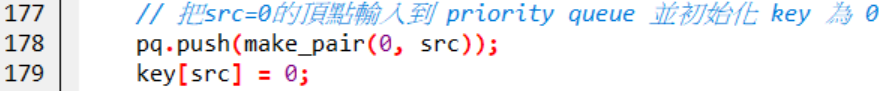
自動產生的描述

利用課堂上的教的MST-KRUSKAL(G,w)和MST-PRIM(G,w,r)找出Minimum spanning tree，並且print出來截圖，兩者皆需顯示出MST的edge還有weight如下圖：



Note: 當我使用Prim algorithm時，我選定從頂點0開始延伸，並令其key為0。





我實際印出的MST的edge還有weight如下:

一張含有 文字 的圖片

自動產生的描述

1. 討論

**Q1**: 如果有dense graph(edge多)跟sparse graph(edge少)兩種graph，他們分別適合哪種演算法，為甚麼？

**A1**: dense graph比較適合使用Prim algorithm，因為edge多，使用Prim algorithm每一步只要考慮現在選的頂點周圍的edge有沒有在MST上面和weight是不是最小。但是如果使用 Kruskal algorithm，除了一開始sort edge的時間就會比較長，當要檢查兩個subgraph可不可以合併(會不會形成cycle)時需要找兩個subgraph的源頭是不是一樣，這個所花的時間也會比較長。那sparse graph就兩者演算法執行時間差不多，所以兩者都可以用。比較數學形式的講法如下:

觀察 prim algorithm和 kruskal algorithm的時間複雜度。

讓我們定edge數為m 和頂點數為 n 。

Prim algorithm的時間複雜度為 O(m+nlog(n))，而 kruskal algorithm的時間複雜度為 O(mlog(n))。

(1) 在dense graph的情況下，例如當 m=ω(nlog(n)) 時，prim algorithm在 O(m) 中執行，比 kruskal algorithm的 O(mlog(n)) 更快。

(2) 對於sparse graph，假設 m=O(n) ，我們讓 prim algorithm在 O(nlog(n)) 中執行，而 kruskal algorithm在 O(nlog(n)) 中執行，兩者執行時間差不多。

**Q2**: 如果Graph是由兩個小Graph的集合(這兩個小Graph彼此之間沒有任何edge連結)，則哪個方法會失敗，沒辦法讓每個node被span到？為甚麼？

**A2**: Prim algorithm會失敗，Kruskal algorithm會成功。因為Prim algorithm是從一個頂點出發一直延伸下去，一定不會越過中間沒有edge連接的subgraph，所以會有頂點沒有被span到。不過Kruskal algorithm沒有管你兩個subgraph有沒有連起來，這個演算法是按照所有edge的key (或說weight) value的順序來決定下一步要挑什麼edge，所以不同的subgraph會各自生出自己的的MST，而且所有頂點都會被span到。我們來做實驗試試看:

一張含有 文字, 手錶, 時鐘 的圖片

自動產生的描述

我們把上圖這三個edge去掉，觀察執行結果:

一張含有 文字 的圖片

自動產生的描述

可以發現到還是可以正常使用Kruskal algorithm，而且會形成兩個MST:

1 – 0 – 7 – 6 – 8 – 2以及3 – 4 – 5。

但是如果使用Prim algorithm，因為我是從頂點0開始，所以左半邊的subgraph可以找到 0 – 1 – 2 – 8 – 6 – 7 的MST，但是右半邊subgraph就找不到MST了。

1. 心得

這一次作業比較困難，需要花比較多時間搞懂演算法該如何實作，這次作業讓我學到typedef自定義pair的型態，這樣要存放label與key value變的簡單很多，我以前在上資料結構的時候，還不知道那樣的方法，我只會用一般int型態的vector存adjacent list、建立path，最後弄得很複雜，也不夠有系統、有邏輯的方法去實作。另外我也學到priority queue的用法，以及如何運用在Prim algorithm。這學期就這樣結束了，時間過得好快，希望暑假還能加強自己coding的能力，把演算法都念到更熟，沒教到的部分也自己讀完，考研究所才有希望。