資料結構作業四

109703013 資科二 周彥綸

一、設計公路網路的方法:

首先我先在地圖範圍內隨機出座標,並且時時檢查是否有重複。接著我將城市分級，第一個城市(點集中 index 為 0 者)為首都,index 介於 1~0.15n(n 為總結點數)者為直轄市，index 介於 0.15n~0.5n 者(35%的機率)為縣市，最後 50%為城鎮，而決定這些節點是否連接的依據,是依照這兩點的距離是否短於最大距離(首都和直轄市的距離)乘以權重(若是沒有那麼必要相連，則合理距離會下修)，小於時表示距離適當，程式便會將兩者連起，例如:最大距離為 45，則如果直轄市與城鎮間的距離小於 45*0.5=22.5 則會將他們相連，而我以現實生活中三種不同可能的城市規劃方法，設計出不同的模型和權重分配表:

1.港埠型首都(所有城鎮以連接到首都為主要目標)

第一點 / 第二點	首都	直轄市	縣市	城鎮
首都	(不會相接)	1	0.9	0.8
直轄市	1	0.7	0.6	0.5
縣市	0.9	0.6	0.45	0.4
城鎮	0.8	0.5	0.4	0.35

2.同心圓模式(相同階級的城鎮相接機率最高，就像北京市那樣一圈一圈的分布)

第一點 / 第二點	首都	直轄市	縣市	城鎮
首都	(不會相接)	1	0.6	0.45
直轄市	1	0.9	0.5	0.4
縣市	0.6	0.5	0.8	0.35

城鎮	0.45	0.4	0.35	0.7

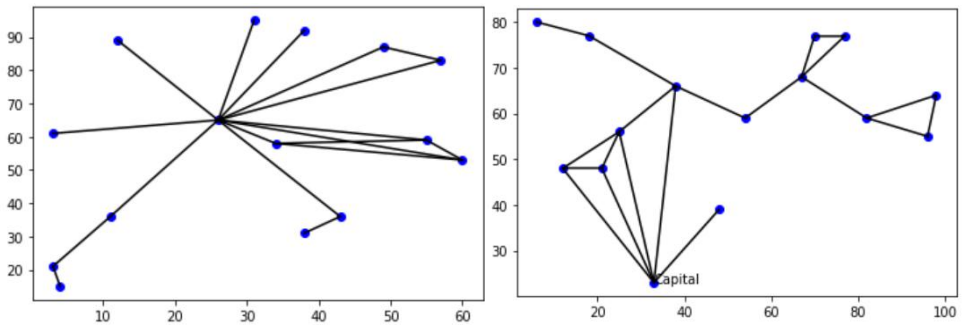
3.平衡發展(各點的連接相對平均，不過依然會受到城市等級影響)

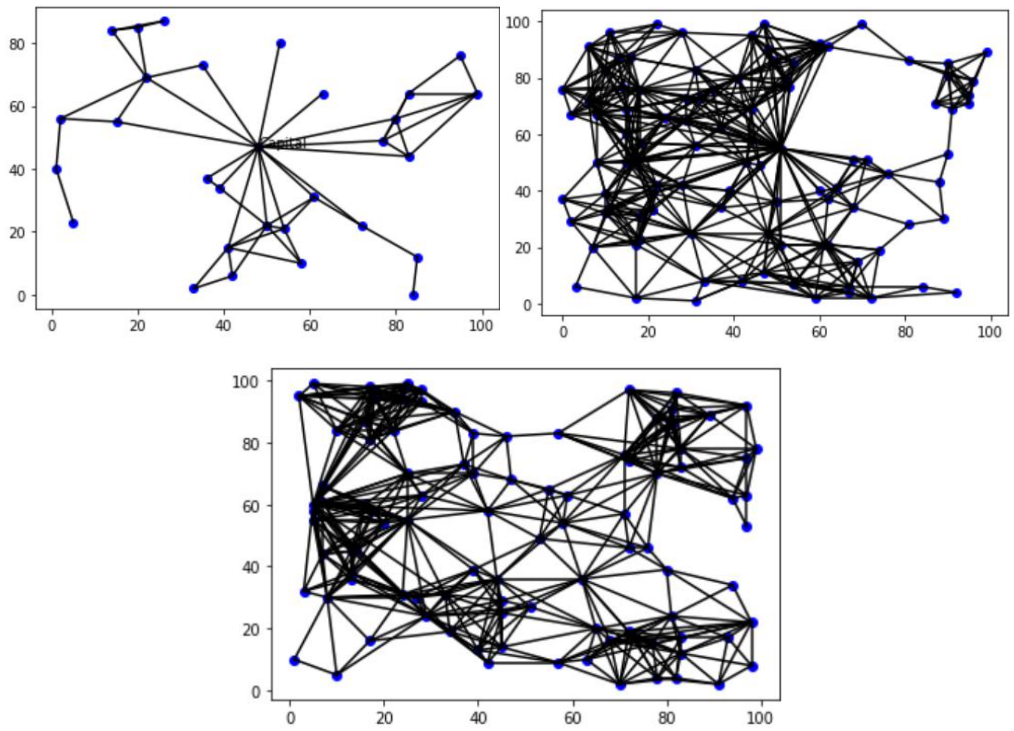
第一點 / 第二點	首都	直轄市	縣市	城鎮
首都	(不會相接)	1	0.8	0.5
直轄市	1	0.9	0.7	0.45
縣市	0.8	0.7	0.6	0.4
城鎮	0.5	0.45	0.4	0.35

最後是檢查該圖是否為連通圖，我利用之前學過的 BFS 和 set 進行檢查，把每一個點所連結的節點放入待處理 Queue 中，而每次檢查此點是否在連通集合裡面，如果沒有則加入到此 set，直到 set 個數為 n 就可以確認為連通圖，反之直到 Queue 跑完依然沒則非為聯通圖。

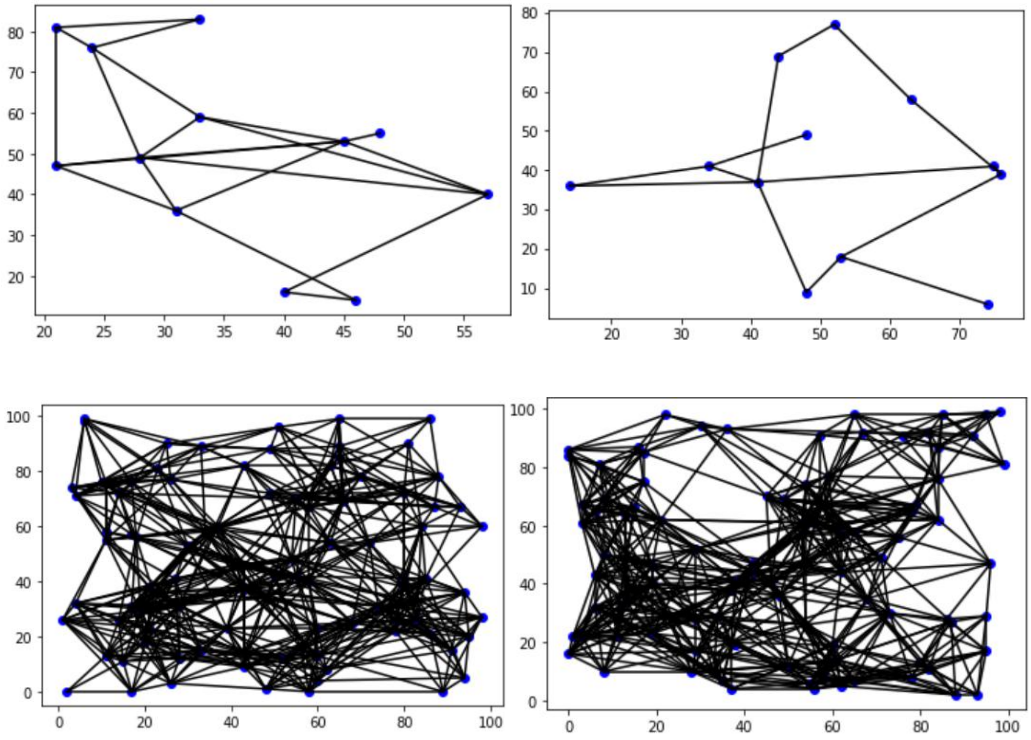
而圖形我以 python 的 matplotlib 寫成，放入資料 list 以後再指定哪兩個點需要連接，所畫出來的圖形如下:

1.港埠型:

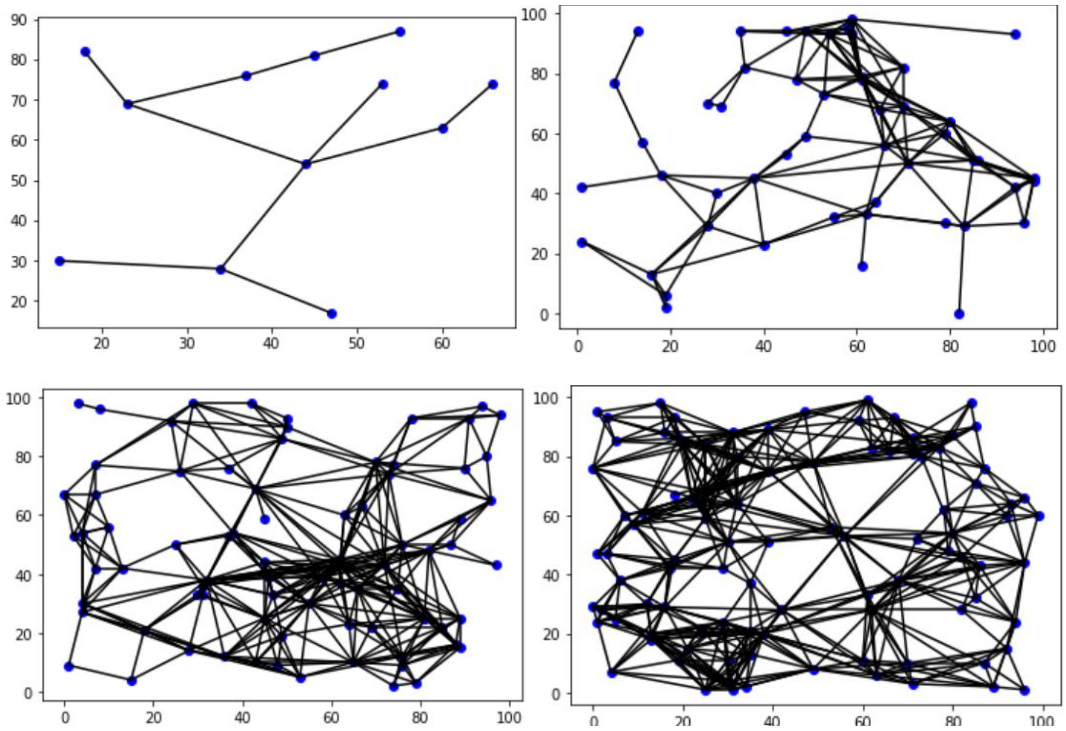




2. 同心圓型:

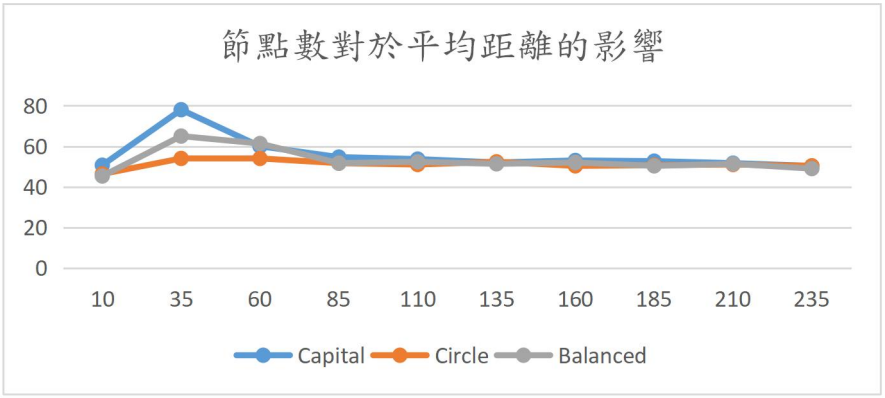


3. 平衡型:



二、節點數對於平均距離的影響

因為 n 太大時，連接兩點所需的時間太多、需要儲存的邊集太大，所以這次不以 n=2^k 進行成長，而是使用每次多加 50 個點來讓系統的負擔較小。而依照我做出來的結果再經過平均，會得出以下的折線圖(橫軸為節點數):



其中有兩點值得注意:

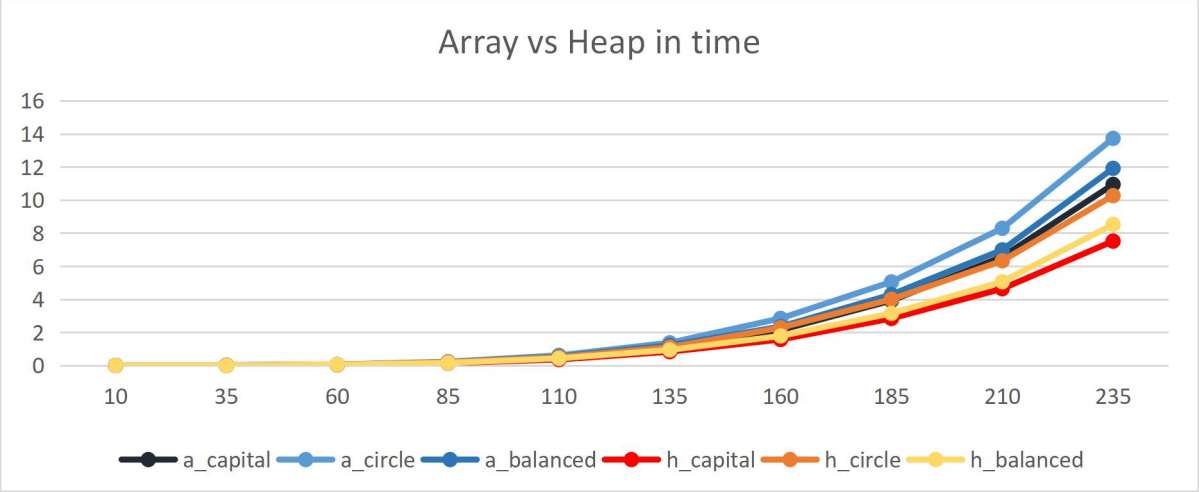
1. 在節點數較小時，實驗出的平均距離彼此差距較大，容易不穩定、有極端值

2. 平均距離隨著節點數增加先升後降，最後三者趨同，接近最大距離

我想可能是因為節點分布密度的關係，剛開始地圖未用到的空間還很多，新增加的點即使符合其中一點的合理距離，也有很大的機率和其他點距離很遠，容易形成地圖上的端點，造成平均距離過大;也因為節點數量小，所以更有可能都聚集在彼此附近，讓

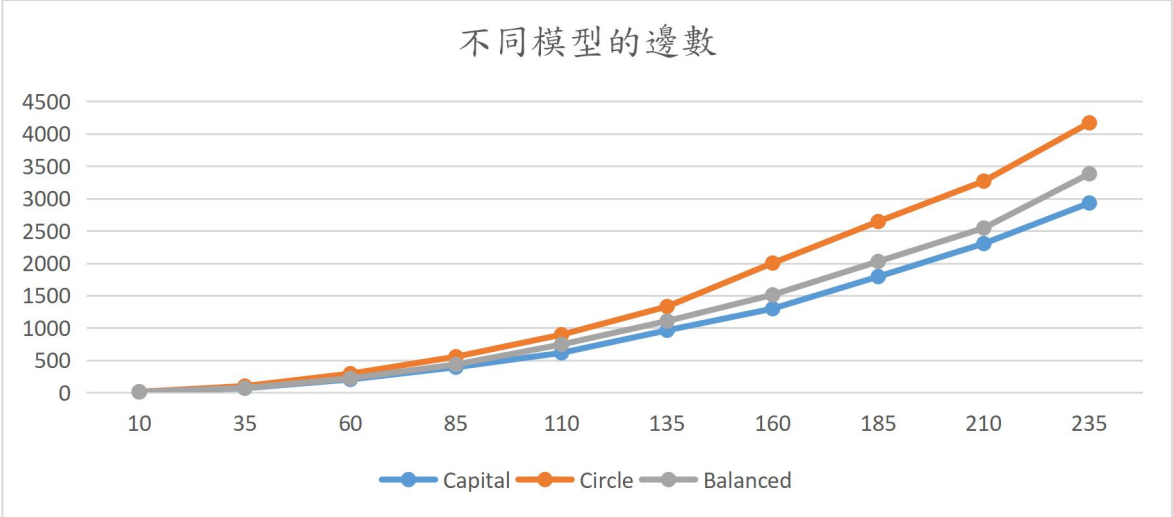
平均距離過小;而隨著節點增加，會越來越容易和其他的點相連讓端點存在的可能越來越少，而增加的點也有可能出現在兩點相連之最短距離附近，讓 A、B 兩點更能透過趨近於最短路徑來連結讓距離降低，導致後期的距離縮短。

三、使用的儲存方式對於時間的影響:



在這一次的比較中，需時 $O(n^2)$ 的 array_dijkstra 和需時 O(mlogn) 的 heap_dijkstra 不會像之前比較 $O(n^2)$ 的 sorted array 和 O(logn) 的 linked list 一樣差距那麼大，畢竟這次是在 n 不會到太大的情況下， 而 mlogn 和 n^2 相比大約為 1/2~2/3，所以在地圖上仍然可以看出兩者的增長，在 n 很小時也有可能測得 heap 所需時間大於 array 的情況, 不過整體來說，以 array 來儲存(藍色的那幾條線)所需的時間皆大於以 heap, 符合理論值。

而在三種模型中，同心圓模型所需要的時間最久，再來是平衡型，最後才是港埠型，原因是來自於邊數，如下圖所示，因為最常發生的情況是縣市連結縣市，而其機率以同心圓模型最高，所以會連結最多的邊，需要嘗試的路徑也會最多，故得出最久的時間。



四、心得

這一次自我設計的體驗真的很新奇，不像以前寫作業有明確的規範，而是要自己訂定想要研究的目標，並且運用自己的能力來達成效果，像我就對每一種模型所產生的圖形感到相當有興趣，這次作業著實考驗我們的好奇心與創造力。

但這次對我來說也是最困難重重的一次，因為我一開始在設計節點、連接方法和權重時和查詢到 dijkstra 內部設計不太一樣，導致我的兩支 dijkstra 程式都是需要自己花時間設計，也因為我不會寫 python，也花了一些時間看懂 python 的 list、 numpy array 和 matplotlib ，不過也因此收穫很多，像是讓我更加熟悉了 dijkstra 細微的部分，例如該如何設計、連結「查找指標」來馬上取得想要的資料，還有了解到 python 的實用性。

（也因為我的 code 花了一些時間寫，所以我希望能夠在作業附上，證明我真的有想過每一個步驟而和網路上的 code 不同）