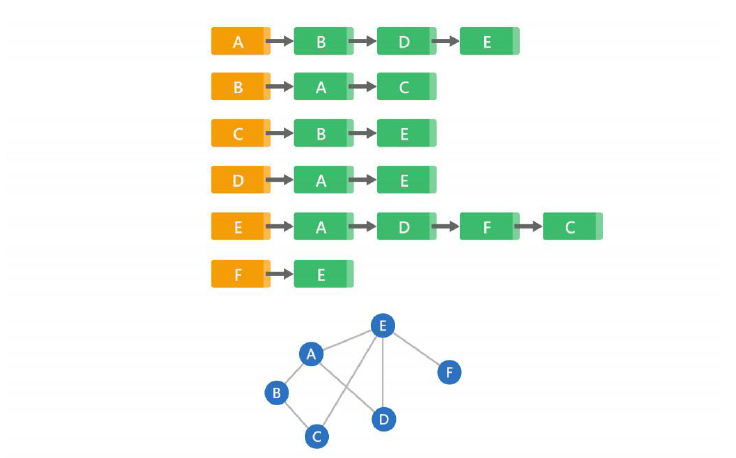
0819823陳子祈 演算法導論 HW7

1. 測試演算法正確性



按照上圖，用adjacency-list建立一個Graph

一張含有 文字 的圖片

自動產生的描述

以A為start，用BFS(G,s=A)列出該算法的visit order，還有每個vertex相對於A的minimum distance，print出來應如下：

visit order：['A', 'B', 'D', 'E', 'C', 'F']

minimum distance：[0,1,1,1,2,2]

一張含有 文字 的圖片

自動產生的描述

以A為start，用DFS-VISIT(G,s=A)列出該算法的visit order，print出來應如下：

visit order：['A', 'B', 'C', 'E', 'D', 'F']



1. 討論

Q1: 兩個visit tree的方式的複雜度是否一樣？一樣的話為甚麼？不一樣的話差在哪？

A1:

|  |  |  |
| --- | --- | --- |
|  | 時間複雜度 | 空間複雜度 |
| BFS | O(V + E) | O(V) |
| DFS | O(V + E) | O(V) |

其中 V 是頂點數，E 是圖中的邊數。

BFS及DFS兩者的時間複雜度與空間複雜度都一樣，先講時間複雜度，因為兩者終止搜尋的時機點都是當所有的頂點數都已經被訪問完，另外兩者都是搜尋到一條路的盡頭的時候，都會退回原來這條路的與其他條路的交會點(BFS是會把交會點存起來，DFS則是會return回交會點)，所以兩者的時間複雜度除了與頂點數有關，也與圖中的邊數有關。 再來講空間複查度，因為兩者的使用到的adjacent list及visited陣列是一樣的，所以兩者理論上空間複雜度是一樣的。

Q2: 當Graph的規模變很大的時候，誰消耗的Memory可能會比較多？為甚麼？

A2: 雖然兩者的空間複雜度是一樣的，可是DFS使用的是遞迴呼叫，每次遞迴呼叫一次，OS都會分配一段記憶體空間。BFS只是使用迴圈，所以DFS應該會消耗比較多。

Q3: 舉一個生活上常見的例子是可以利用BFS或者DFS來實現

A3: 把網路(Internet)看成一張圖(graph)，可以把網路的router看成一個頂點，router與router之間的聯繫看成邊，把router之間傳輸時間看成邊的權重，若想要知道從我家的電腦傳一封email到總統府的電腦，可利用BFS或DFS搜尋最快傳到的路徑，使用這個路徑傳輸。不過實際上不只有我送資料，途中還會經過很多其他人送的資料，還會有其他機制來避免碰撞，所以假設一個網路是沒有其他任何人跟我同時在使用，就可以使用BFS或DFS來搜尋最短傳輸時間的路徑。

1. 心得

這一次作業比較簡單，沒有需要測試時間複查度，只需要實作BFS與DFS，我了解這兩個搜尋演算法之後，實作也沒有很難。只剩最後一個作業，這學期就結束了，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，考研究所才有希望。