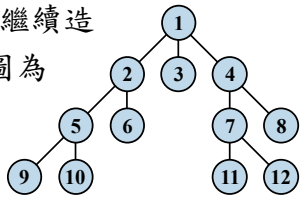
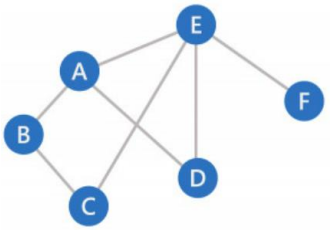
# 演算法導論 HW8

110511254 徐煜絨

## 一、BFS

廣度優先搜尋是將一個點的鄰居全部造訪過，再繼續造訪其中一個鄰居的鄰居，很適合拿來找最短路徑。右圖為當對一顆 tree 做 BFS 時，每個 vertex 被造訪的先後順序。針對下圖，假設從 A 開始，則 BFS 造訪的順序可能會是 A -> E -> B -> F -> D -> C。




## 二、程式架構

這次程式是要用 BFS 找最短路徑。因為 map 是 17*17，所以我令 map 這個 vector 是 19*19，其中 map[0][j]和 map[i][0]是為了計算能更直覺，而 map[18][j]、map[i][18]是為了在走到第 17 行或第 17 列時，不會因為 vector 的 size，導致出現 segmentation fault。map[0][j]、map[i][0] 、map[18][j]、map[i][18]的 num 全被設為 0，即視為 barrier。如果起點或終點是 2，則會對 cost 的初始值和最後的 print 進行調整；如果起點或終點是 0，則會輸出 error。

在建 BFS 的過程，我對課本上的 pseudo code 做調整，因為地圖上除了 barrier，還會有編號 2 的格子。在實作上，我捨棄掉 white, gray, black 這三個對於 vertex 的標示，並會對於每個被造訪的點做處理。當造訪到 u，且 w1（設 w1 是 u 的其中一個鄰近格子）不是 barrier，則會比較 u.cost+w1.num 是否小於 w1.cost，如果小於，則讓 w1.cost 變成 u.cost+w1.num。

在最後 print 的部分，我建立一個 vector ‘back’，從(17, 17)開始往回追蹤，如果在路徑上就加進 back，往回找路徑的方法為，當往回走到 vertex u 時，看 u 哪個鄰居的 cost 是 u.cost-u.num，如果有多種選擇，則優先順序是上下右左。當 trace 回(1, 1)，就開始 output ‘back’裡面的 vertex 的位置。