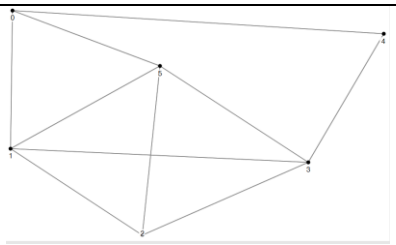
<table>
<tr>
<td>

```
input node:6
input edge:10
adjacency matrix :
0  1  0  0  1  1
1  0  1  1  0  1
0  1  0  1  0  1
0  1  1  0  1  1
1  0  0  1  0  0
1  1  1  1  0  0

adjacency list :
0-->1 -->4 -->5
1-->0 -->3 -->5 -->2
2-->3 -->1 -->5
3-->2 -->4 -->1 -->5
4-->3 -->0
5-->1 -->0 -->2 -->3
```

</td>
<td>

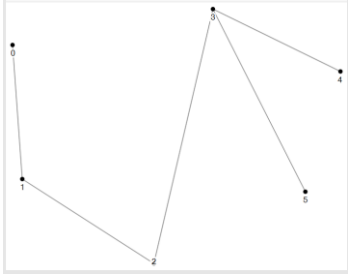


解釋:node 數為 10、edge 數 6，代表 10 個 vertices 中有 6 條 edge，且因無向圖所以 matrix 會為對稱的，而 list 則是有 0->1 就會有 1->0 也是對稱的概念。

</td>
</tr>
<tr>
<td>

```
DFS:
0->1->2->3->4->5
0  1  0  0  0  0
0  0  1  0  0  0
0  0  0  1  0  0
0  0  0  0  1  1
0  0  0  0  0  0
0  0  0  0  0  0
```

</td>
<td>

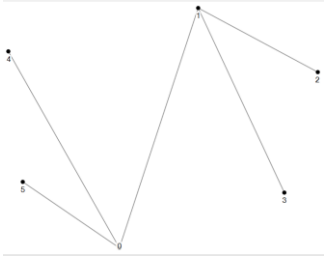


解釋:代表此 DFS TREE(matrix 表示)尋訪順序為 0->1->2->3->4->5 因為 matrix 的迴圈由 i=0 開始，所以在尋訪時會自動先走數字小的，而 List 會根據插入順序有所變動。

</td>
</tr>
<tr>
<td>

```
BFS:
0->1->4->5->2->3
0  1  0  0  1  1
0  0  1  1  0  0
0  0  0  0  0  0
0  0  0  0  0  0
0  0  0  0  0  0
0  0  0  0  0  0
```

</td>
<td>



解釋:代表此 BFS TREE(matrix 表示)尋訪順序為 0->1->4->5->2->3 因為 BFS 是將鄰近的 vertices 放進 QUEUE 裡，所以尋訪順序與 DFS 不同，由 0 附近的 3 個(1,4,5)先放入再一個個取出，取出同時再放入取出 vertex 的鄰近 vertex 到 QUEUE 裡也就是取出 1 後放 2,3。

</td>
</tr>
</table>