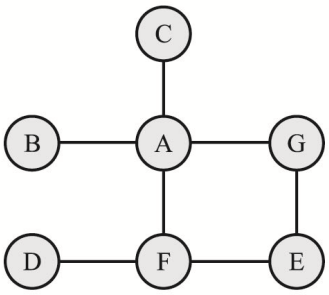
Ex. 1 將下列圖形轉換為相鄰串列。



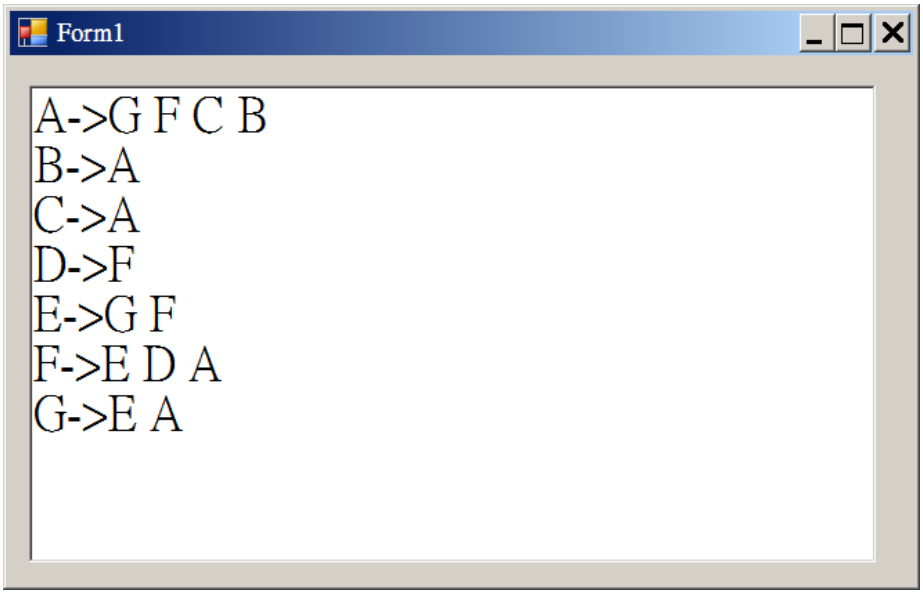
頂點 V = { A,B,C,D,E,F,G }
邊線 E = { AB, AC, AF, AG, DF, EF, EG }
相鄰串列

```
A -> G F C B
B -> A
C -> A
D -> F
E -> G F
F -> E D A
G -> E A
```

資料結構

```
class VERTEX {     //節點
   String ^name;  //目前節點名稱
   list   ^adjlist;   //相鄰串列
};

class list {
      VERTEX ^v;     //指向相鄰節點的指標
      list   ^next; //次項指標
};
```



textBox1

(程式碼)

```
ref class VERTEX {
    ref class list {
    public:
        VERTEX ^v;
        list   ^next;
        list( VERTEX ^x ){ v=x; next=nullptr; }
    };
private:
    String ^name;
    list   ^adj_list;
public:
    VERTEX( String ^s ){ name=s; adj_list=nullptr; }
    void NewAdj( VERTEX ^v ){
        list ^x=gcnew list( v );
        x->next = adj_list;
        adj_list = x;
    }
    void Link( VERTEX ^v ){
        NewAdj( v );
        v->NewAdj( this );
    }
```

```
        String^ AdjacentList(){
            String ^s=name+"->";
            list ^x=adj_list;
            while( x !=nullptr ){
                s+=x->v->name + " ";
                x=x->next;
            }
            return s;
        }
    };
    VERTEX ^A, ^B, ^C, ^D, ^E, ^F, ^G;
private: System::Void Form1_Load(System::Object^
                sender, System::EventArgs^  e) {
            A=gcnew VERTEX("A");
            B=gcnew VERTEX("B");
            C=gcnew VERTEX("C");
            D=gcnew VERTEX("D");
            E=gcnew VERTEX("E");
            F=gcnew VERTEX("F");
            G=gcnew VERTEX("G");
            A->Link( B );
            A->Link( C );
            A->Link( F );
            A->Link( G );
            D->Link( F );
            E->Link( F );
            E->Link( G );

            String^ s;
            s  = A->AdjacentList() + "\r\n";
            s += B->AdjacentList() + "\r\n";
            s += C->AdjacentList() + "\r\n";
            s += D->AdjacentList() + "\r\n";
            s += E->AdjacentList() + "\r\n";
            s += F->AdjacentList() + "\r\n";
            s += G->AdjacentList() + "\r\n";
            textBox1->Text = s;
        }
```

Ex. 2 接上題，對該圖形進行深度優先搜尋。

textBox1

參考演算法"基本圖論"深度搜尋部分
搜尋時不知有幾個節點，因此須使用一個串列(L) 來記錄已訪問過的節點。

```
void dfs( VERTEX ^v, list ^% L){
    //先嘗試訪問目前節點
    if( IsVisited( v, L ) ) return;
    Visit( v, L );

    //嘗試訪問相鄰串列裡紀錄的所有相鄰節點
    list ^x = v->adj_list;
    while( x!=nullptr ){
        if( !IsVisited( x->v, L ) ) dfs( x->v, L );
        x=x->next;
    }
}
```

(程式碼)

```
ref class VERTEX {


    ref class list {
    public:
        VERTEX ^v;
        list   ^next;
        list( VERTEX ^x ){ v=x; next=nullptr; }
    };


private:
    String ^name;
    list   ^adj_list;
public:
    VERTEX( String ^s ){ name=s; adj_list=nullptr; }
    void NewAdj( VERTEX ^v ){
        list ^x=gcnew list( v );
        x->next = adj_list;
        adj_list = x;
    }
    void Link( VERTEX ^v ){
        NewAdj( v );
        v->NewAdj( this );
    }
    String^ AdjacentList(){
        String ^s=name+"->";

        list ^x=adj_list;
        while( x !=nullptr ){
            s+=x->v->name + " ";
            x=x->next;
        }
        return s;
    }
```

```
    bool IsVisited( VERTEX ^v, list ^% L ){
        list ^x = L;
        while( x!=nullptr ){    //頂點多時效率差
            if( x->v == v ) return true;
            x=x->next;
        }
        return false;
    }
    void Visit( VERTEX ^v, list ^% L){
        list ^x = gcnew list( v );
        x->next = L;
        L = x;
    }
    void dfs( VERTEX ^v, list ^% L){
        if( IsVisited( v, L ) ) return;

        Visit( v, L );

        list ^x = v->adj_list;
        while( x!=nullptr ){
               if( !IsVisited( x->v, L ) )
                                   dfs( x->v, L );
               x=x->next;
        }
    }
    String^ DFS(){
        list ^L= nullptr;  // L 記錄已訪問的節點
        dfs( this, L );

        list ^x=L;
        String ^s="";
        while( x!=nullptr ){
            s=x->v->name + s;
            x=x->next;
        }
        return s;
    }
};
```

```
    VERTEX ^A, ^B, ^C, ^D, ^E, ^F, ^G;

private: System::Void Form1_Load(System::Object^
                    sender, System::EventArgs^  e) {
          A=gcnew VERTEX("A");
          B=gcnew VERTEX("B");
          C=gcnew VERTEX("C");
          D=gcnew VERTEX("D");
          E=gcnew VERTEX("E");
          F=gcnew VERTEX("F");
          G=gcnew VERTEX("G");

          A->Link( B );
          A->Link( C );
          A->Link( F );
          A->Link( G );
          D->Link( F );
          E->Link( F );
          E->Link( G );

          String^ s = A->DFS();

          textBox1->Text = s;
        }
```