



中原大學 資訊工程學系

VLSI 設計自動化導論  
Final Project 書面報告

資訊四乙 10727211 林彥輝

中華民國一一〇年十月

# 讀檔、建檔、資料結構

## 1. 通用資料結構

Vertex	Node
+ name : string	+ name : string
+ weight : int	+ vertex_info : vector<Vertex>
+ io : string	
+ Vertex() : constructor	+ Node() : constructor
+ show_info() : void	+ show_info() : void
+ Get_IO() : string	+ Add_vertex : void
+ Get_Name() : string	+ Get_Name() : string
+ Get_Weight() : int	+ HaveVectorInput() : bool

使用 Class Vertex 模擬邊、以 Class Node 模擬點。在 Class Vertex 中的資訊包含 name、weight、io 以及 Setter()（透過 constructor 代替）、Getter()。而在 Class Node 中的資訊除了 name 以外也包含了此 Node 中的各種 io，並且新增 HaveVectorInput() 來確認此頂點是否有相關的邊，供外部 Function 做界接。

## 2. 讀檔、執行方式

Read_Parse	Plases input a choice(0, 1, 2):
+ Identify() : bool	C 67
+ ReadFile() : vector<string>	I 73
+ Parser() : vector<Node>	R 82
+ run() : vector<Node>	C 67
	U 85
	I 73
	T 84
	32
	a 97
	v 118
	q 113
	l 108
	a 97
	r 114
	g 103
	e 101
	13
	[Enter]

Class Read\_Parse 負責讀檔及建立 DataBase，在 ReadFile() 中使用 fstream library 將 txt 檔案一行行透過 string 讀出，並包裝成 vector<string> 型別透過 Parser 進行 GetToken，過程中不斷透過 Identify() 確認 char 的單位，最終整理出一系列的 Node 型別，Node 型別中的 vector\_info 也會有 Vector 的資訊，這些種種的 Node 會被整理成 vector<Node> 型別回傳。

與此同時，發現 Linux 系統中會多抓取 ascii 碼 13 的字元，與 Windows 中模擬的環境會有出入，過程中也特別針對此 case 加做處理，以便程式在不同環境中能執行出相同的結果。

# DFS (depth first search)

## 1. 程式說明

DFS
+ visited_node : vector<string>
+ stack_node : vector<string>
+ Have_Visited() : bool
+ Have_Stack() : bool
+ FindVecorOutput() : string
+ run() : void

class DFS 中的 data member 有 visited\_node 與 stack\_node，各儲存的是已走訪過的 Node 名稱以及因 DFS 機制所儲存之尚未走訪之 Node 名稱。

程式中從 run()開始進入，以 do-while 為架構一個個呼叫節點，直到各節點都走訪過(透過 Have\_Stack()查看 stack\_node 確認是否仍有尚未走訪之節點)，走訪過程中，會順帶更新 visited\_node，以及更新 stack\_node，更新 stack\_node 的決策方法為先確認 io 方向以及是否曾走訪過，最後透過 FindVecorOutput()界接 Node 中的 HaveVectorInput()找出 Node，若此節點已存在 visited\_node 當中，則將先消去其位置，最後再將其餘 Node 透過 ID 排序 push 進 stack\_node 中。

## 2. 執行結果 & 執行方式

```
* * * * * VLSI final project * * * * *
*           Student ID : 10727211           *
* 0. QUIT                                   *
* 1. DFS (depth first search)               *
* 2. shortest path(Dijkstra)               *
*****
Plases input a choice(0, 1, 2): 1
[DFS algorithm] (depth first search)
Order: S >> V1 >> V3 >> V5 >> V9 >> D >> V7 >> V10 >> V4 >> V2 >> V6 >> V8
End of the Simulate Successfully
```

如截圖所示，作業同作業解說影片呈互動式介面，只需將檔案設定名稱為”input.txt”並與程式放置同個目錄即可執行，並再 choice 中選擇 1。結果將透過 Order 欄呈現，走訪順序由左至右輸出至螢幕上。

# Dijkstra algorithm

## 1. 程式說明

Dijkstra
+ table : int
+ visited : bool
+ table_size : int
+ list : vector<Node>
+ FindVecorOutput() : string
+ All_Visited() : bool
+ Find_List_Index() : int
+ run() : void

class Dijkstra 中儲存 data member(儲存各節點與 S 距離)、visited(走訪記錄)、table\_size(節點數)、list(節點資訊)

程式從 run() 中進入，已 do-while 為架構不斷走訪節點，終止條件為呼叫 All\_Visited() 判斷是否連結節點皆走訪過。走放過程中會先選定 Node，初始化將設定為 S，否則將選定尚未走訪過且距離最短的 Node，並更新 visited、table，table 更新的方式為將當前的節點距離與邊上的權重做加總並與 table 中的原先值做比較取最小值。

## 2. 執行結果& 執行方式

```
* * * * * VLSI final project * * * * *
*           Student ID : 10727211           *
* 0. QUIT                                   *
* 1. DFS (depth first search)               *
* 2. shortest path(Dijkstra)                *
* * * * *
Please input a choice(0, 1, 2): 2
Vertex (S) distance = 0
Vertex (V1) distance = 3
Vertex (V2) distance = 2
Vertex (V3) distance = 4
Vertex (V4) distance = 3
Vertex (V5) distance = 6
Vertex (V6) distance = 4
Vertex (V7) distance = 5
Vertex (V8) distance = 8
Vertex (V9) distance = 8
Vertex (V10) distance = 6
Vertex (D) distance = 9
End of the Simulate Successfully
```

如截圖所示，作業同作業解說影片呈互動式介面，只需將檔案設定名稱為“input.txt”並與程式放置同個目錄即可執行，並再 choice 中選擇 1。輸出結果將直接顯示再螢幕上，資訊將包含節點名稱與距離。