資料結構 HW5

系級 : 電機系大三

姓名 : 林凡皓

學號 : B103012002

1. Explain build\_graph( ) and search\_from\_graph\_dfs( )

* build\_graph( ) :

build\_graph函數主要由多個for loop組成，最外層的兩個for loop會針對迷宮矩陣中每一個位置做迭代。每一次迭代中，會先去檢查該位置是否為通路，如果是通路，就會記錄當前位置的ID，並判斷該位置是否已經儲存過，沒有儲存過就將該位置儲存。接著去判斷該位置的鄰居是否為通路，這邊也是利用for loop去迭代上下左右四個方向。每一次迭代中都先判斷該鄰居是否為通路，如果是的話就記錄位置並判斷該位置使否有被儲存過，如果有就將當前位置與鄰居位置之間加上一條邊，這裡需要呼叫兩次add\_edge因為要加入雙向的邊。

* search\_from\_graph\_dfs( ) :

search\_from\_graph\_dfs( )主要透過Stack來實現。

首先會先抓取start vertex並判斷該vertex是否存在。將start vertex的顏色設設定成灰色，代表說該vertex正在被探索，並初始化探索時間。透過while loop來探索迷宮，當stack內還有東西時，while loop就會持續執行。While loop一開始先去查看stack頂部的資料，將all\_neighbors\_visited參數設定成True，並判斷是否要更改該vertex的顏色，接著針對上下左右進行搜索。搜索過程會先去判斷鄰居是否被搜索過，如果沒有的話，將該鄰居加入stack中，並將鄰居的previous設定成目前的vertex，all\_neighbors\_visited參數也設定成False，代表說還有鄰居未被探索。當所有鄰居皆被探索後，將目前vertex的顏色改成黑色，並將其從stack中移除。接著判斷該vertex是否為出口，如果是出口的話，將(row, column)加入到path中並將path回傳。

1. Design of my program

* search\_from\_graph\_bfs( ):

1. 實現方法 :

整體實現方式與dfs類似，主要差別在於bfs採用queue做實現。

首先會先抓取start vertex並判斷該vertex是否存在。將start vertex的顏色設設定成灰色，代表說該vertex正在被探索，並初始化探索時間。透過while loop來探索迷宮，當queue內還有東西時，while loop就會持續執行。While loop一開始將queue中最前面的vertex移出來，接著針對上下左右進行搜索，判斷每一個鄰居的顏色是否為白色，如果是白色代表說該位置還沒被搜索，則將該位置加入queue中，並將該鄰居的previous設定成當前在探索的vertex。接著去判斷當前vertex是否為出口，如果是的話就將當前位置加入path中，並沿著pervious一路將搜索出來的路徑加入path中，最後再回傳path。

1. 執行結果 :

* Maze 1 :

一張含有 Rectangle, 螢幕擷取畫面, 黃色, 鮮豔 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型, 印刷術 的圖片

自動產生的描述

* Maze 2 :

一張含有 Rectangle, 螢幕擷取畫面, 正方形, 圖表 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型, 黑色 的圖片

自動產生的描述

* Maze 3 :

一張含有 Rectangle, 螢幕擷取畫面, 黃色, 正方形 的圖片

自動產生的描述



* search\_from\_graph\_dijkstra( ):

1. 實現方法 :

Dijkstra主要透過priority queue來實現。

一開始先去抓取初始位置的ID，並將該位置的距離設定成0然後加入priority queue中。接著初始化一個set為visited，紀錄拜訪過的vertex。透過while loop來探索迷宮，當priority queue內還有東西時，while loop就會持續執行。每一次迭代都先將priority queue中最優先的vertex拿出來並將其加入visited中，接著先去判斷該vertex是否為出口，如果是則利用while loop將該位置的pervious加入path中，最後回傳path。如果不是出口，利用get\_neighbors來取得目前位置的鄰居，並將鄰居的distance令成當前distance加一(每一個路徑的權重皆為1)，接著去判斷說新的距離是否小於原先的距離，如果小於的話就更新距離並將鄰居的previous設定成當前位置。

1. 執行結果 :

* Maze 1:

一張含有 Rectangle, 螢幕擷取畫面, 黃色, 鮮豔 的圖片

自動產生的描述

一張含有 文字, 字型, 螢幕擷取畫面, 印刷術 的圖片

自動產生的描述

* Maze 2 :

一張含有 Rectangle, 黃色, 螢幕擷取畫面, 正方形 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

* Maze 3 :

一張含有 黃色, 螢幕擷取畫面, Rectangle, 正方形 的圖片

自動產生的描述



1. Analyze the difference between three algorithms

BFS與DFS的差異相當明顯，BFS會將目前位置的鄰居全部搜索完成後，才去搜索鄰居的鄰居，而DFS會從一個鄰居開始一路找到底之後，才對另外一個鄰居做搜索。

DFS無法保證說找到的路徑為最短路徑，但是他會去將所有路徑都搜索過一次；相對的BFS不一定會將所有度竟都搜索過，但是可以保證找到的路徑為最短路徑。

Dijkstra與BFS有些相似，都是可以尋找到最段路徑的演算法，但是BFS沒有辦法對有權重的路徑做搜索，而dijkstra支援對有權重路徑的圖做搜索。驗算法上的差異在於，BFS會將為搜索過的鄰居加入queue中，而dijkstra會在未搜索過的路徑找找到權重最小的路徑做搜索。

1. What I have learned

在這次作業中，我認為最大的收穫在於找出學習這些演算法的盲點，像是在讀上課講義的時候，我以為我已經很了解dijkstra演算法的原理以及運作方式，但是輪到我自己去應用這套演算法時我才發覺到其實有很多細節被我忽略了。

除此之外，雖然說走迷宮這個問題在作業三的時候就已經透過stack來實現過一次了，但是這一次使用不同的驗算法來實做相同的問題感覺很不一樣，而且也能夠更深的體會到不同演算法在同一個問題中的表現差異。