dijkstra algorithm 的模擬小專題

Github 網址(內有程式碼):

https://github.com/ChristopherChen070535/dijkstra-algorithm-

撰寫人:

國立台北大學通訊工程學系三年級陳品鈞

- **專題簡介**

  此專題只要是利用 C 語言寫出一個程式，並使用 Dijkstra 演算法計算從指定起點到其他點的最短路徑。計算完成後，將最短路徑轉換成字串並輸出，然後輸出最短路徑或者訊息表示無路徑存在。根據下圖 1 搜索算法來計算從源點在不經過重複點之情況，回到原點之最短路徑，並且將沒有標示出路徑的距離設為 1000。

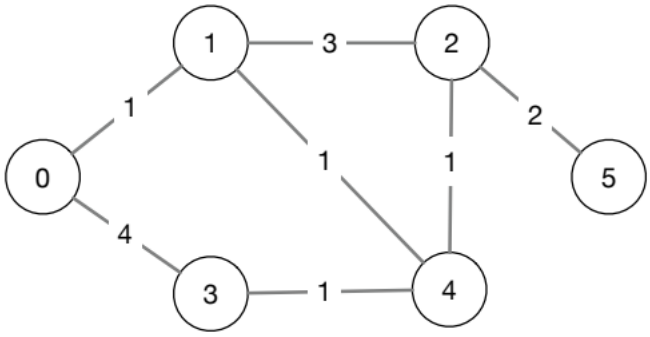

圖 1:此專題之路徑圖

- **成果**
  - 程式輸入: 此程式的輸出依賴於使用者輸入的源點（s)。根據源點和預先定義的成本矩陣（cost 如下圖 1)，計算從源點到所有其它節點的最短路徑

$$\text{cost}[10][10] = \begin{bmatrix} 0 & 1 & 10000 & 4 & 10000 & 10000 & 10000 & \cdots \\ 1 & 0 & 3 & 10000 & 1 & 10000 & 10000 & \cdots \\ 10000 & 3 & 0 & 10000 & 1 & 2 & 10000 & \cdots \\ 4 & 10000 & 10000 & 0 & 1 & 10000 & 10000 & \cdots \\ 10000 & 1 & 1 & 1 & 0 & 10000 & 10000 & \cdots \\ 10000 & 10000 & 2 & 10000 & 10000 & 0 & 10000 & \cdots \\ 10000 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}.$$

圖 1:路徑之矩陣

- 程式輸出結果:

1. 輸入想計算的原點，且存在最短路徑：當從源點到至少一個其它節點存在一條或多條路徑時，程式將輸出找到的最短路徑。這條路徑以節點序列的形式表示，節點之間用逗號分隔，如下圖 2。

```
Source: 4
1 0 3 5 4 5 10000 10000 10000 10000
1 -1 1 0 2 2 0 0 0 0

4 3 0 5 4 2 10000 10000 10000 10000
1 2 -1 4 1 2 0 0 0 0

4 5 7 0 6 9 10000 10000 10000 10000
3 0 4 -1 1 2 0 0 0 0
```

圖 2:原點設為 4 之情況

2. 輸入想計算的原點，但不存在最短路徑：如果從源點無法到達任何其他節點（即沒有可達的路徑），程式將輸出「There is no path」。這表示給定的源點是孤立的，或者與圖中其他節點之間沒有連接路徑。

```
Source: 5
There is no path
```

圖 3:沒有路徑之結果

- **程式內容(path 函式的主要想法)**:

1. 初始化：我首先初始化 cnt 變數為 0，表示與起始節點相鄰的節點的個數。接著設置 adjacent[] 陣列，用於標記與起始節點相鄰的節點，並且設定初始值為 0。 這兩個陣列是用於遍歷圖中起始節點 s 的所有相鄰節點，並更新 adjacent[] 陣列和 cnt 變數。
2. 處理相鄰節點個數：如果 cnt 等於 0 或 1，表示起始節點沒有相鄰節點或僅有一個相鄰節點，則將 p 字串設置為 "n"，表示沒有可達的路徑。

3. 處理相鄰節點的最短路徑：將起始節點 s 加入到路徑字串 p 中。
4. 遍歷與起始節點相鄰的每個節點：計算從起始節點到該相鄰節點的路徑成本 total_cost。創建臨時的 cost 陣列 temp[][]，將該相鄰節點的邊的成本設置為無窮大，以避免後續迴圈中重複訪問該節點。

- **心得與反思**:通過這個專題的實作，我深入理解了 dijkstra 演算法的運作原理，並將其應用於實際的程式開發中。這不僅提高了我們的演算法和程式設計能力，也為我們日後的工作和學習提供了寶貴的經驗。這次我主要卡住的困難點是如何將陣列運算結合入 dijkstra 演算法中，一開始我的想法是直接對於同一個陣列運算到底，但這個方法非常麻煩且容易出錯，因此我使用了短暫儲存的陣列去防止我在計算上出錯。