

1121DS TA #04

TAs：李冠穎、陳柏翰、藍振恩、朱孟淇



Outline

- 規則
- Problem A
- 補充



規則

通過 HackerRank 系統 (網頁型judge評分工具) 的評測，使用瀏覽器即可submit程式碼。

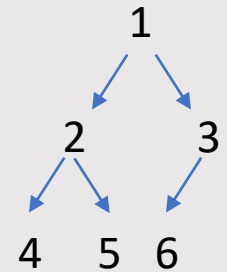
- HackerRank網址：<https://www.hackerrank.com/ds-fp>
- 請在 HackerRank 系統 Setting 設定自己的 **Your Username** 為登入 moodle的帳號(舉例: Z91234567，z_201234)，未設定成規定格式導致成績無法登錄時**後果自負**。
- 其他規則詳見 Moodle 文件。
- 11/23 練習題無需使用此系統。



Problem A

- 建立一簡單的二元樹，並實現深度優先遍歷(DFS, Depth First Search)
- 建立至少六個節點並賦予數值 1 ~ 6

```
C:\Users\user\Documents\Dev\Untitled2.exe
Tree traversal (DFS): 1 2 4 5 3 6
-----
Process exited after 0.03205 seconds with return value 0
請按任意鍵繼續 . . . . .
```





```
#include <iostream>

using namespace std;

// 定義樹節點的類別
class TreeNode {
public:
    int val; // 節點的值
    TreeNode* left; // 左子節點指標
    TreeNode* right; // 右子節點指標

    // 節點建構函式，初始化節點值和左右子節點為空指標
    TreeNode(int value) : val(value), left(NULL), right(NULL) {}
};

// 深度優先遍歷函式，遞迴方式遍歷樹並輸出節點值
void depthFirstTraversal(TreeNode* root) {
    if (root == NULL) { // 若當前節點為空，直接返回
        return;
    }

    cout << root->val << " "; // 輸出當前節點的值
    depthFirstTraversal(root->left); // 遞歸處理左子樹
    depthFirstTraversal(root->right); // 遞歸處理右子樹
}
```

```
int main() {
    // 建立節點並建立樹狀結構
    TreeNode* root = new TreeNode(1); // 根節點為1
    root->left = new TreeNode(2); // 左子節點為2
    root->right = new TreeNode(3); // 右子節點為3
    root->left->left = new TreeNode(4); // 2的左子節點為4
    root->left->right = new TreeNode(5); // 2的右子節點為5
    root->right->left = new TreeNode(6); // 3的左子節點為6

    // 進行深度優先遍歷 (DFS)
    cout << "Tree traversal (DFS): ";
    depthFirstTraversal(root); // 從根節點開始進行深度優先遍歷
    cout << endl;

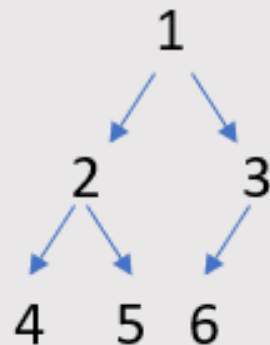
    // 釋放記憶體
    delete root->left->left; // 釋放4節點
    delete root->left->right; // 釋放5節點
    delete root->right->left; // 釋放6節點
    delete root->left; // 釋放2節點
    delete root->right; // 釋放3節點
    delete root; // 釋放1節點，也就是根節點

    return 0;
}
```



補充

- 前、中、後序走訪



```
C:\Users\user\Documents\Dev\Untitled2.exe
Pre-order traversal: 1 2 4 5 3 6
-----
Process exited after 0.03454 seconds with return value 0
請按任意鍵繼續 . . .
```

```
C:\Users\user\Documents\Dev\Untitled2.exe
In-order traversal: 4 2 5 1 6 3
-----
Process exited after 0.031 seconds with return value 0
請按任意鍵繼續 . . .
```

```
C:\Users\user\Documents\Dev\Untitled2.exe
Post-order traversal: 4 5 2 6 3 1
-----
Process exited after 0.03077 seconds with return value 0
請按任意鍵繼續 . . .
```

```
// 前序遍歷函式
void preOrderTraversal(TreeNode* root) {
    if (root == NULL) {
        return;
    }

    cout << root->val << " "; // 輸出當前節點的值
    preOrderTraversal(root->left); // 遞歸處理左子樹
    preOrderTraversal(root->right); // 遞歸處理右子樹
}
```

```
// 中序遍歷函式
void inOrderTraversal(TreeNode* root) {
    if (root == NULL) {
        return;
    }

    inOrderTraversal(root->left); // 遞歸處理左子樹
    cout << root->val << " "; // 輸出當前節點的值
    inOrderTraversal(root->right); // 遞歸處理右子樹
}
```

```
// 後序遍歷函式
void postOrderTraversal(TreeNode* root) {
    if (root == NULL) {
        return;
    }

    postOrderTraversal(root->left); // 遞歸處理左子樹
    postOrderTraversal(root->right); // 遞歸處理右子樹
    cout << root->val << " "; // 輸出當前節點的值
}
```



Thanks



1121DS TA #05

TAs：朱孟淇、藍振恩、李冠穎、陳柏翰



Outline

- 規則
- Floyd-Warshall Algorithm 介紹
- Practice-最低車流量



規則

通過 HackerRank 系統 (網頁型judge評分工具) 的評測，使用瀏覽器即可submit程式碼。

- HackerRank網址：<https://www.hackerrank.com/ds-fp>
- 請在 HackerRank 系統 Setting 設定自己的 **Your Username** 為登入 moodle的帳號(舉例: Z91234567，z_201234)，未設定成規定格式導致成績無法登錄時**後果自負**。
- 其他規則詳見 Moodle 文件。



Floyd-Warshall Algorithm 介紹

- Floyd-Warshall 演算法可以求出全部節點配對的最短路徑，是一個全配對最短路徑(all-pair shortest path)演算法。
- 此演算法可以處理有負邊的圖，但是不能用以檢查有負迴圈的圖。也就是說，當圖有負邊但是沒有負循環時，Floyd-Warshall 演算法仍然可以求出正確的最短路徑。
- 【原理】採用動態規劃策略(DP)解決問題

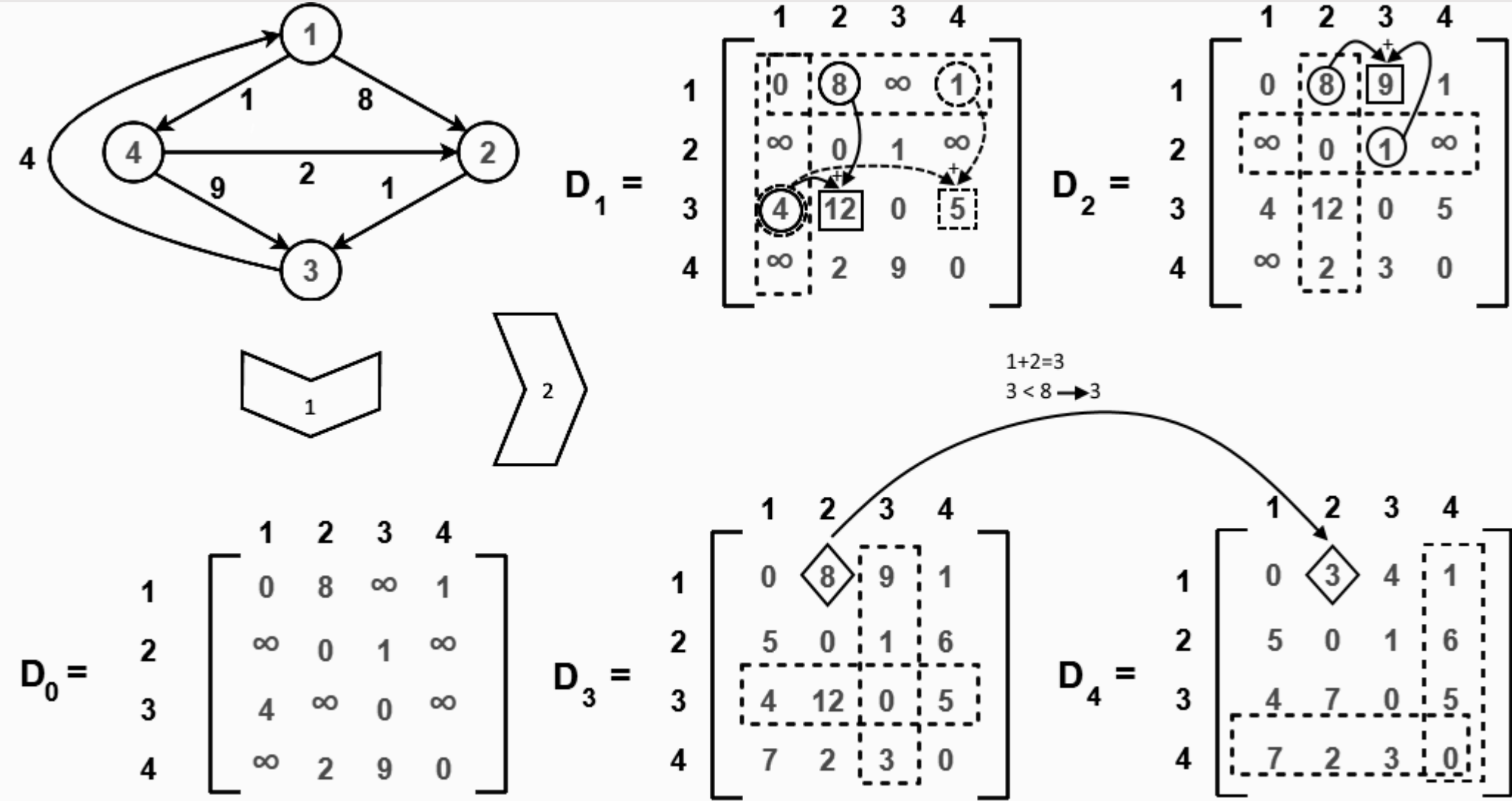


Floyd-Warshall Algorithm 介紹

- 【實作】
- 節點數量為 $(1 \dots k)$
 - 以最短路徑為舉例
 - $D[i][j]$ 表示 i 到 j 的最短距離
 - 除了 $D[i][i] = 0$ 其他設定為無限大
 - 加入其他邊
 - 如果最短路徑會經過 k 點，則 $D[i][k] + D[k][j] < D[i][j]$ ，替換答案
- 【複雜度】 (N 為頂點)
 - 時間複雜度： $O(N^3)$ ，因需檢查所有點到所有點且中間點為其他點的路徑
 - 空間複雜度： $O(N^2)$ ，利用一個 $N \times N$ (N 為節點總數) 的二維陣列來記錄每一節點配對間的最短路徑成本或距離。

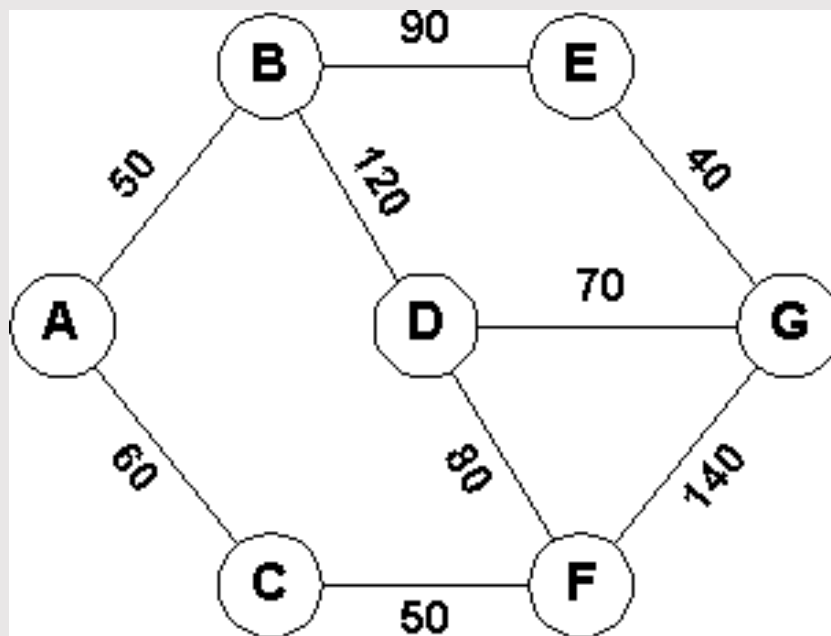


Floyd-Warshall Algorithm介紹



Practice-最低車流量

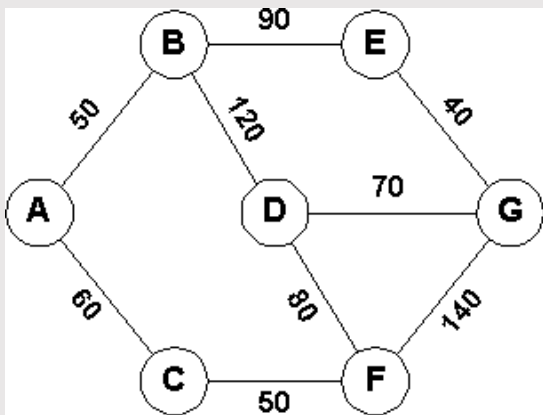
- 現在有一件工程是「車流」的疏通，車流只要超過130就會堵塞，而最適當的車流量大約是60到65，較繁忙的路徑大概是70到80。
- 以下的百貨公司旁代表街道，點代表十字路口。在邊上的整數代表這條街平均的車流大小。



Practice-最低車流量

- 從A到G，你可以走這樣的路徑：ACFG，而你必須忍受的最高車流量就是140。而ACFDG是最舒服的路徑了，因為你面臨的車流量最大不會超過80。
- 請你找出從某個點到某個點你所必須忍受的最低車流量。
- 範例輸入說明:第一行有一數字，代表問題數，每行會有2個點，分別為起點和終點
- 例如第一行3為詢問3條路徑，1 7代表找出從A開始走到G的必須忍受的最低車流量，對應的輸出為80 (路徑為ACFDG)

註:若欲找尋頂點沒有邊連接ex.若AB和AC不存在，則欲找起點或終點為A的最低車流量為no path



範例輸入	範例輸出
3	80
1 7	60
2 6	60
6 2	



Practice-最低車流量

```
#include<iostream>
#include<string>
using namespace std;
const int maxn = 100 + 100;
const int INF = 0x3f3f3f3f; //設定為無限大
int c, s, q; //宣告c為頂點數量 s為邊數 q為需要求出解答的路徑數量
int c1, c2, d;
int dist[maxn][maxn]; //宣告一個二維矩陣做路徑儲存
void Floyd()
{
    for(int k = 1; k <= c; k++)
    {
        for(int i = 1; i <= c; i++)
            for(int j = 1; j <= c; j++)
                dist[i][j] = min(dist[i][j], max(dist[i][k], dist[k][j])); //車流量大小比較
    }
}
```



Practice-最低車流量

```
int main()
{
    int cnt = 0;
    c = 7; //頂點7個
    s = 9; //邊9條
    cin >> q;
    for(int i = 1; i <= c; i++)
        for(int j = 1; j <= c; j++)
            dist[i][j] = INF; //先將依照頂點數量所建構的二維矩陣做初始值無限大設定
    dist[1][2] = dist[2][1] = 50; //AB邊車流量為50
    dist[1][3] = dist[3][1] = 60; //AC邊車流量為60
    dist[2][4] = dist[4][2] = 120;
    dist[2][5] = dist[5][2] = 90;
    dist[3][6] = dist[6][3] = 50;
    dist[4][6] = dist[6][4] = 80;
    dist[4][7] = dist[7][4] = 70;
    dist[5][7] = dist[7][5] = 40;
    dist[6][7] = dist[7][6] = 140;
    Floyd(); //設定好頂點和邊的設定後進行計算
    for(int i = 0; i < q; i++)
    {
        cin >> c1 >> c2; //輸入起點和終點
        if(dist[c1][c2] == INF) //若此路徑為無限大(即無邊連接)
            cout << "no path" << endl; //代表沒有路可走
        else //若有車流量
            cout << dist[c1][c2] << endl; //輸出所求的車流量
    }
}
```



Thanks



Problem C

Chang Kung Kingdom

Time limit: 3 seconds

Memory limit: 512 MB

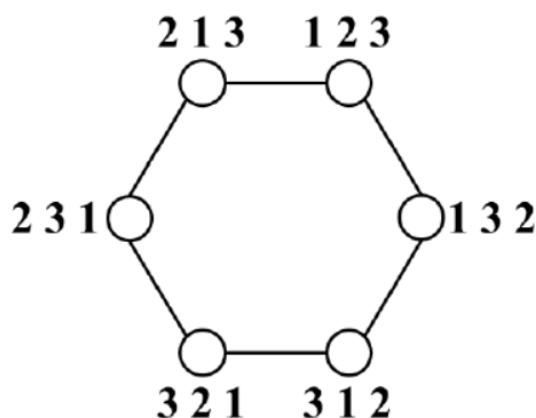
題目內容

腸功王國共有 $3!$ 座城市。每座城市由 3 個數字組成，它們是由 1 2 3 的排列組合。腸功王國的城堡位於 1 2 3 的城市中。 $a_1 a_2 a_3$ 和 $b_1 b_2 b_3$ 分別是城市 A 和城市 B 的編碼，在王國的 A 和 B 之間修建了一條距離為 1 的道路，且僅當存在一個 i ($1 \leq i < 3$)，且滿足以下兩個條件時，該道路才會被修建。

1. $a_i = b_{i+1}$ and $a_{i+1} = b_i$;
2. $a_j = b_j$ for $j \in \{1, 2, 3\} \setminus \{i, i+1\}$

有一天，國王邀請所有市長到城堡開會。請幫市長們計算他們到城堡的路程。請注意，城堡所在城市的編碼為 1 2 3。

舉下圖為例：王國共有 6 座城市，即 n 值為 3。每座城市都以 1 2 3 的排列編碼。



輸入說明

第一行包含一個整數 m , $1 \leq m \leq 5$ ，表示接下來會有幾組測試。下一行由 $\{1, 2, 3\}$ 中排列組合組成，表示城市的編碼，兩個數字之間有一個空格隔開。

輸出說明

對於每組測試，輸出一個整數，表示給定城市與城堡之間的距離。每筆輸出之間有一個換行隔開。

範例輸入 #1

5

1 3 2

2 1 3

3 1 2

2 3 1

3 2 1

範例輸出 #1

1

1

2

2

3