AlgoSeacow 

Graph I

Summer A Class 10

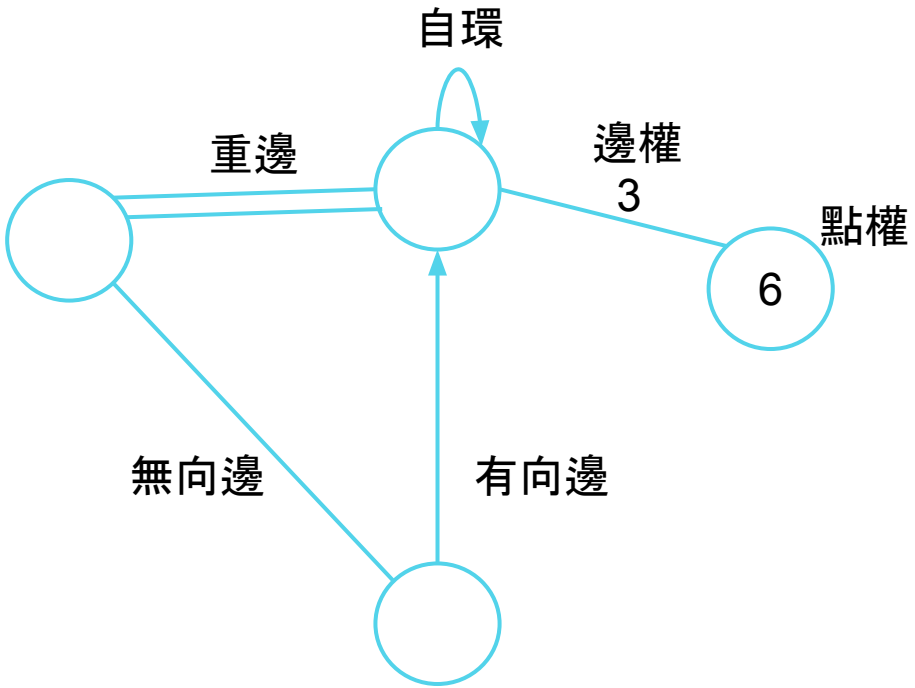
關於這堂課

- 先備知識
  - 資料結構 I
  - Tree I

- 學習重點
  - Graph 基本名詞及儲存方法
  - Graph DFS 和 Graph BFS
  - Topological Sort on Directed Acyclic Graph (DAG)
  - Disjoint Set

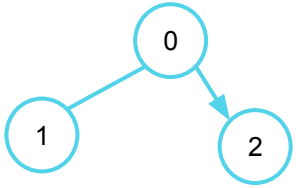
Graph 表示法
(Graph Representation)

Graph 基本名詞



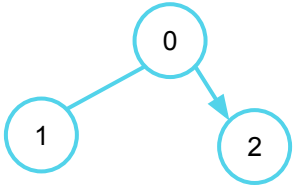
Graph Representation

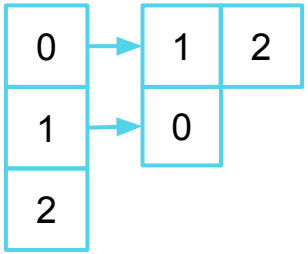
- Adjacency Matrix



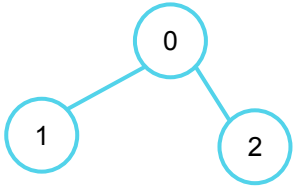
	0	1	2
0	0	1	1
1	1	0	0
2	0	0	0

- Adjacency List





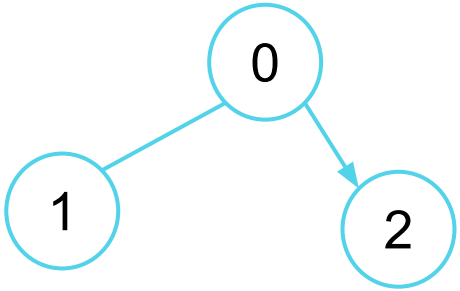
- Edge List



(0, 1)
(0, 2)

Adjacency Matrix

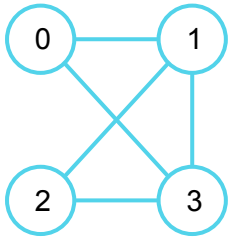
- 空間複雜度 O(V^2)
- 查詢或更新某條邊 O(1)
- 走訪所有邊 O(V^2)
- 實作資料結構
  - `int G[MAXN][MAXN];`



	0	1	2
0	0	1	1
1	1	0	0
2	0	0	0

Adjacency Matrix - Code

```
int n, m;
int G[MAXN][MAXN] = {};

void init() {
    cin >> n >> m;
    for (int i = 0; i < m; i++) {
        int u, v;
        cin >> u >> v;
        G[u][v] = 1;
        G[v][u] = 1;   // 無向邊多加這行
    }
}
```

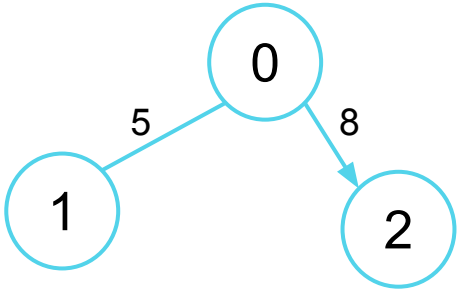


Input

```
4 5
0 1
0 3
1 2
1 3
2 3
```

Adjacency Matrix 帶權

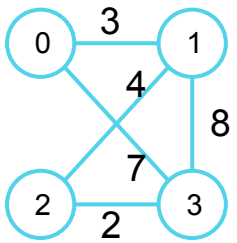
- 空間複雜度 O(V^2)
- 查詢或更新某條邊 O(1)
- 走訪所有邊 O(V^2)
- 實作資料結構
  - `int G[MAXN][MAXN];`



	0	1	2
0	0	5	8
1	5	0	0
2	0	0	0

Adjacency Matrix 帶權 - Code

```
int n, m;
int G[MAXN][MAXN] = {};

void init() {
    cin >> n >> m;
    for (int i = 0; i < m; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        G[u][v] = w;
        G[v][u] = w;   // 無向邊多加這行
    }
}
```

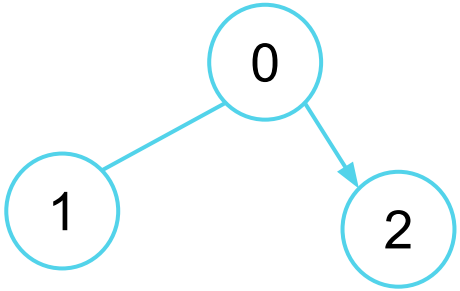


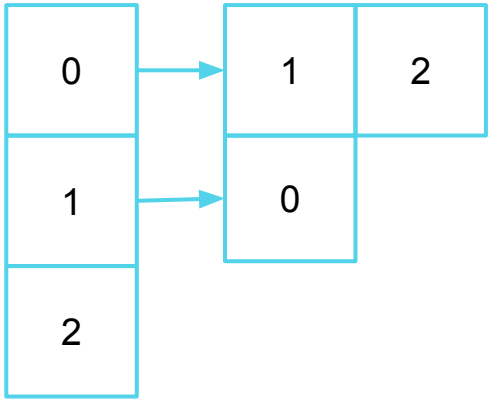
Input

```
4 5
0 1 3
0 3 7
1 2 4
1 3 8
2 3 2
```

Adjacency List

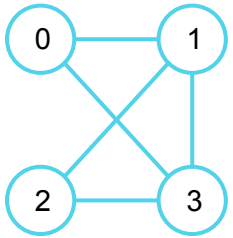
- 空間複雜度 O(E)
- 查詢或更新某條邊 O(V)
- 走訪所有邊 O(E)
- 實作資料結構
  - `vector<int> G[MAXN];`





Adjacency List - Code

```
int n, m;
vector<int> G[MAXN];

void init() {
    cin >> n >> m;
    for (int i = 0; i < m; i++) {
        int u, v;
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);   // 無向邊多加這行
    }
}
```

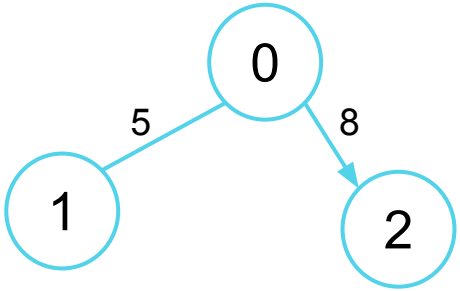


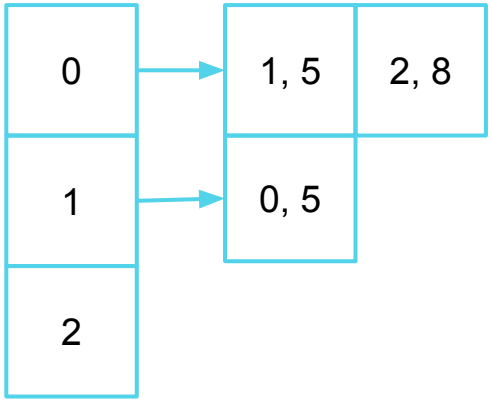
Input

```
4 5
0 1
0 3
1 2
1 3
2 3
```

Adjacency List 帶權

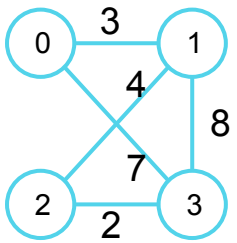
- 空間複雜度 O(E)
- 查詢或更新某條邊 O(V)
- 走訪所有邊 O(E)
- 實作資料結構
  - `vector<Edge> G[MAXN];`
  - `Edge: pair<int, int>` or `struct Edge`





Adjacency List - Code

```
struct Edge {
    int v, w;
};

int n, m;
vector<Edge> G[MAXN];

void init() {
    cin >> n >> m;
    for (int i = 0; i < m; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        G[u].push_back({v, w});
        G[v].push_back({u, w});   // 無向邊多加這行
    }
}
```

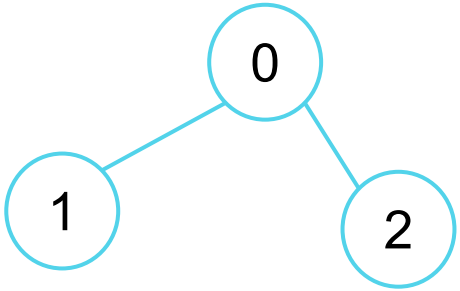


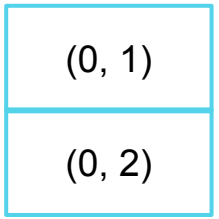
Input

```
4 5
0 1 3
0 3 7
1 2 4
1 3 8
2 3 2
```

Edge List

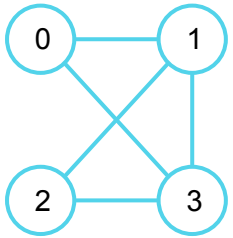
- 空間複雜度 O(E)
- 查詢或更新某條邊 O(E)
- 走訪所有邊 O(E)
- 實作資料結構
  - `vector<Edge>`
  - `Edge: pair<int, int>` or `struct Edge`





Edge List - Code

```
int n, m;
vector<pair<int, int>> edges;

void init() {
    cin >> n >> m;
    for (int i = 0; i < m; i++) {
        int u, v;
        cin >> u >> v;
        edges.push_back({u, v});
    }
}
```



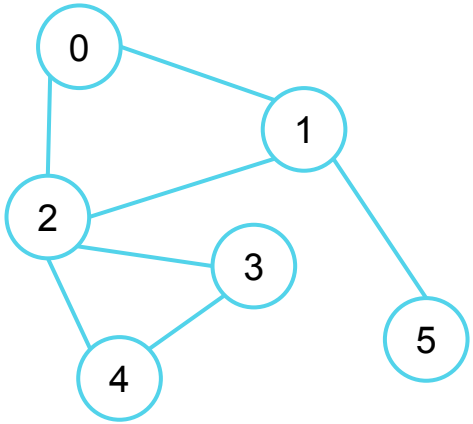
Input

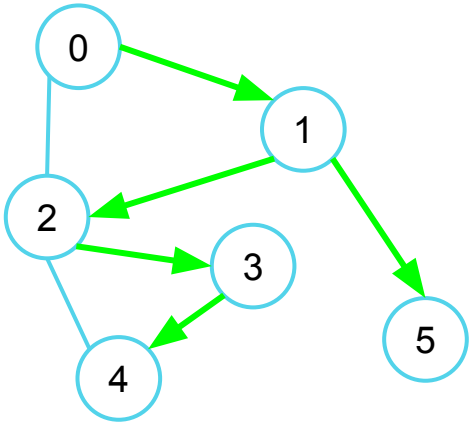
```
4 5
0 1
0 3
1 2
1 3
2 3
```

Graph DFS

Graph Traversal

- Graph DFS



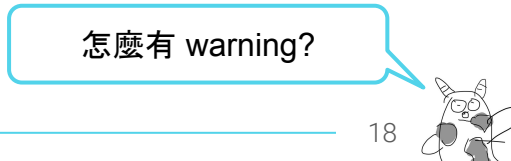


Graph DFS - Structure

- 寫法和 Tree DFS 類似, 但 node v 沒被走過才需要 dfs

```
bool vis[MAXN] = {};

void dfs(int u) {
    vis[u] = true;
    for (int i = 0; i < G[u].size(); i++) {
        int v = G[u][i];
        if (vis[v] == true) continue;
        dfs(v);
    }
}
```

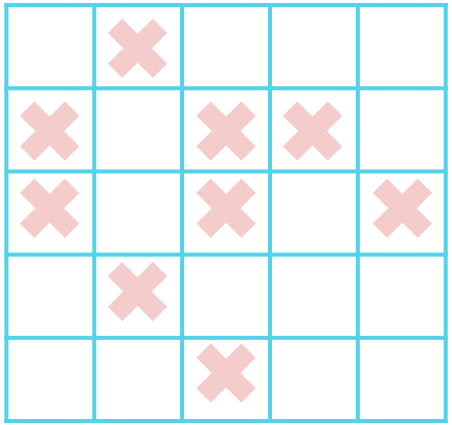
- 每條邊最多只會被走一次, 時間複雜度 O(n + m)



CSES - Counting Rooms

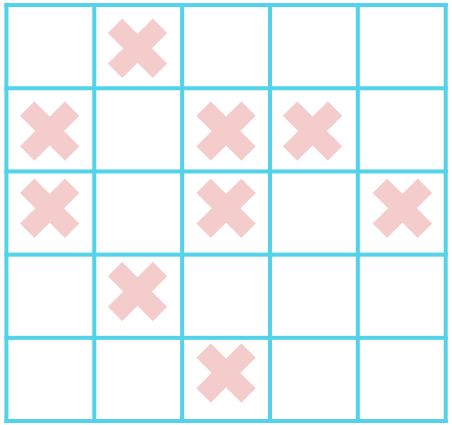
- 題目敘述
  - 給一個 nxm 的 grid, ' . ' 代表地板, ' # ' 代表不可通過的牆壁
  - 問有幾個不聯通的房間

- 測資範圍
    - 1 ≤ n, m ≤ 1000



CSES - Counting Rooms

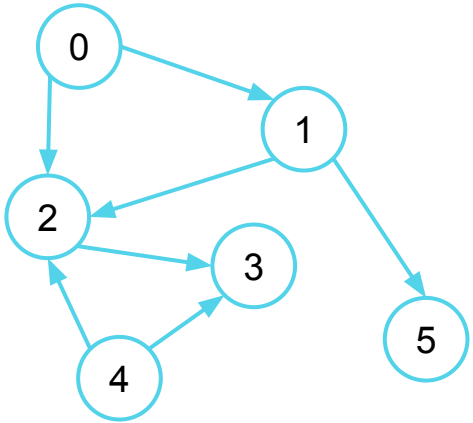
- 作法
  - 依序檢查每個格子
  - 如果發現一個格子尚未走過(vis == 0)，就把它以及跟它同一個連通塊的東西都設定成走過，並把 ans++



LeetCode 841. Keys and Rooms

- 題目敘述
  - 給一張有向圖跟一個起點，問能不能從起點走完這張圖

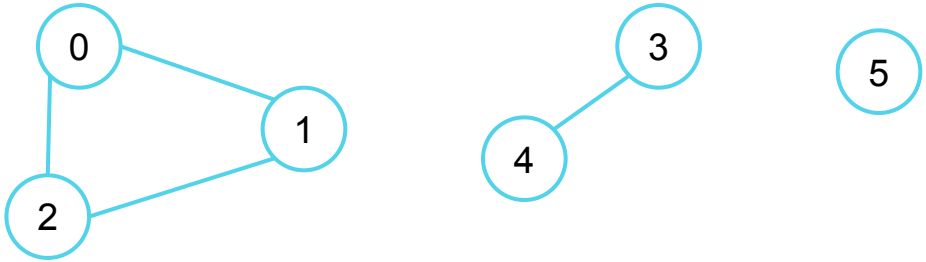
- 測資範圍
  - 節點數 n ≤ 1000，邊數 m ≤ 3000



CSES - Building Roads

- 題目敘述
  - 有 n 個城市與 m 條雙向道路，每條道路連接兩個不同的城市
  - 現在要蓋最少條新的道路讓n 個城市兩兩都有一條路徑連通
  - 問最少要蓋幾條，以及哪些城市之間要蓋新的路

- 測資範圍
  - 節點數 n ≤ 100000，邊數 m ≤ 200000



題單 - DFS

- Grid DFS
  - zerojudge c129: 00572 - Oil Deposits : grid graph 連通塊個數
  - CSES - Counting Rooms
  - leetcode 695. Max Area of Island
  - leetcode 130. Surrounded Regions
  - TIOJ 1336 (八連通 Grid)
- Graph DFS
  - LeetCode 841. Keys and Rooms
  - CSES - Building Roads
  - CSES - Building Teams (黑白圖色判斷二分圖)
  - uva 280. Vertex (找出起點無法到達的點)

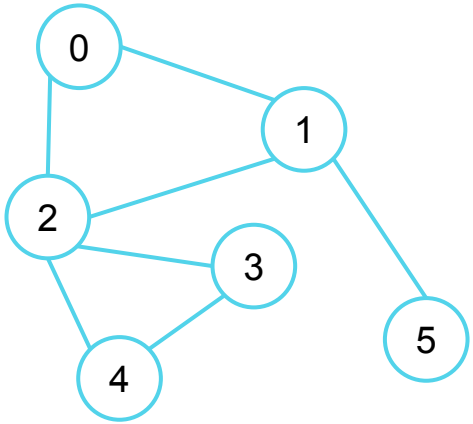
題單 - DFS

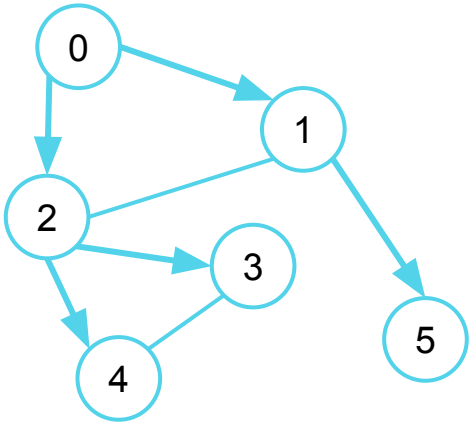
- Hard
  - CSES - Round Trip
  - CSES - Round Trip II
  - leetcode 959. Regions Cut By Slashes
  - leetcode 1391. Check if There is a Valid Path in a Grid

Graph BFS

Graph Traversal

- Graph BFS





Graph BFS

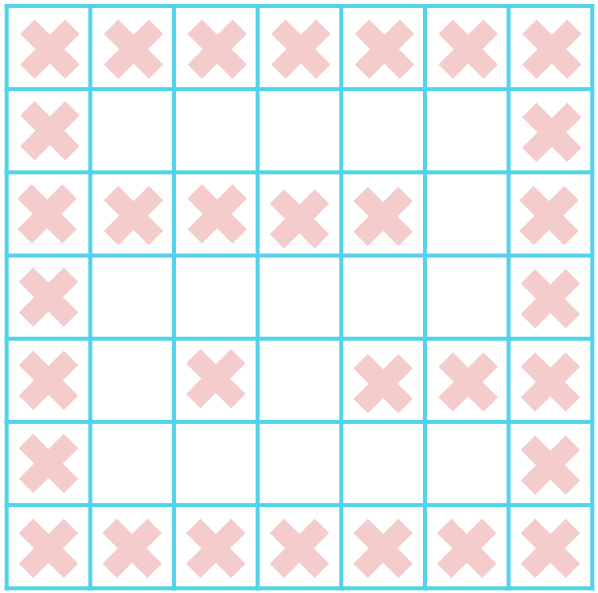
- node v 沒被走過才需要丟進 queue
- 每條邊最多只會被走訪一次, 時間複雜度 O(n + m)

```
bool vis[MAXN] = {};

void bfs(int s) {
    queue<int> q;
    q.push(s);
    vis[s] = true;
    while (q.size()) {
        int u = q.front();
        q.pop();
        for (int i = 0; i < G[u].size(); i++) {
            int v = G[u][i];
            if (vis[v]) continue;
            q.push(v);
            vis[v] = true;
        }
    }
}
```


ZeroJudge a982: 迷宮問題#1

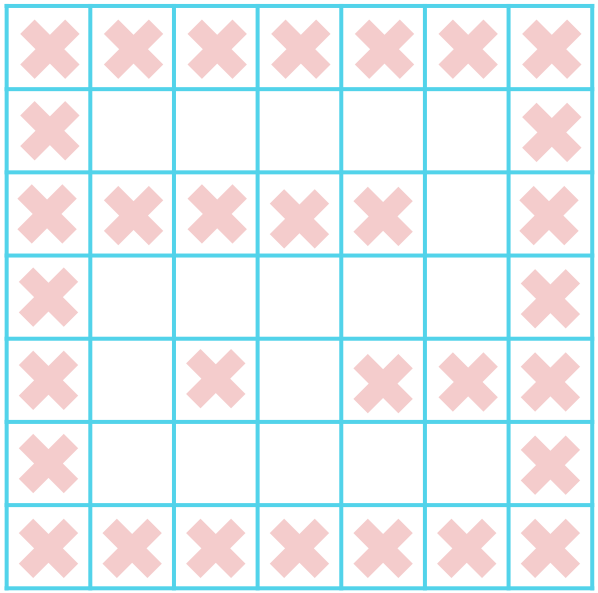
- 題目敘述
  - 給你一個 n * n 格的迷宮，迷宮中以 # 代表障礙物，以 . 代表路
  - 從 (2, 2) 出發，目的地是 (n - 1,n - 1)
  - 求包括起點和終點，最少路徑的長度

- 測資範圍
  - n ≤ 100



ZeroJudge a982: 迷宮問題#1

- 作法
  - 以 (source_x, source_y) 表示一個節點
  - (x, y) 連到 (x, y-1), (x, y+1), (x-1, y), (x+1, y)
  - BFS 順便紀錄距離
    - u 走到 v
    - dis[v] = dis[u] + 1



LeetCode 542. 01 Matrix

- 題目敘述
  - 給一個 n x m 的 01 矩陣，對於每個 cell 輸出其到最近 0 的距離

- 測資範圍
  - n x m ≤ 10000，至少有一個 0

0						
				0	0	0
		0				
					0	
				0		
0						

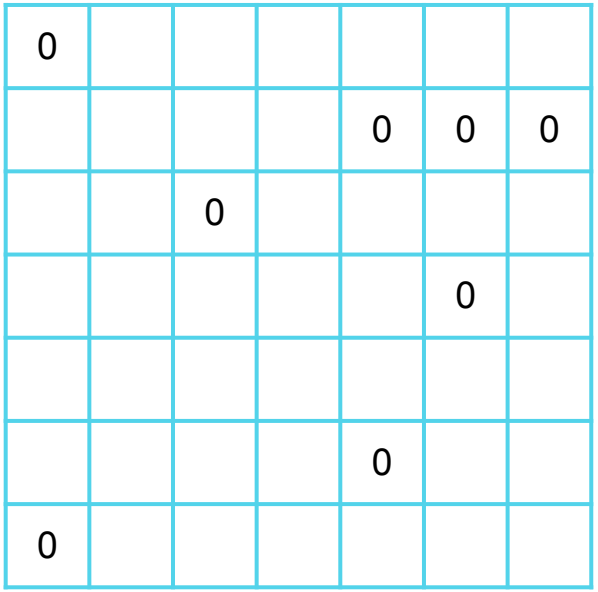
LeetCode 542. 01 Matrix

- 作法一
  - 每個 0 都當起點做一次 BFS
  - 時間複雜度 O($n^2 * m^2$)

0						
				0	0	0
		0				
					0	
				0		
0						

LeetCode 542. 01 Matrix

- 作法二
  - 使用 DP
  - 左上角過來的最近的0 的距離
  - dp(i, j) =
    - if (a[i][j] == 0) : 0
    - else : 1 + min{ dp(i-1, j), dp(i, j-1) }
  - 四個方向各跑一次
  - 時間複雜度 O(n * m)



LeetCode 542. 01 Matrix

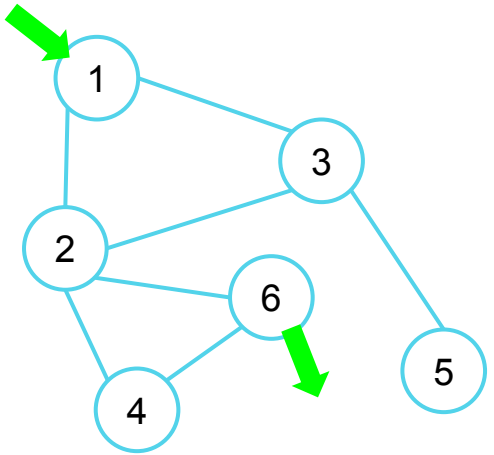
- 作法三
  - 多源點 BFS
  - 一開始把所有的 0 的位置放入 queue
  - 時間複雜度 O(n * m)

0						
				0	0	0
		0				
					0	
				0		
0						

CSES - Message Route

- 題目敘述
  - 給一張無向圖(無邊權)，指定起點終點
  - 求一條起點到終點的最短路徑

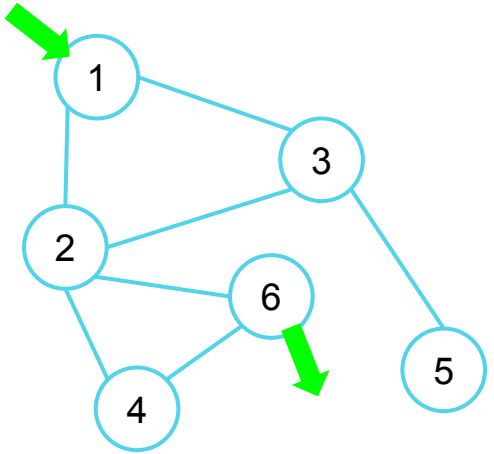
- 測資範圍
  - 節點數 n ≤ 100000，邊數 m ≤ 200000



CSES - Message Route

- 作法
  - 先求出 BFS 距離
  - 從終點往回一步一步找到是從哪裡來的

- BFS tree
  - 把每個點從哪個點過來畫出來，會是一棵樹



題單 - Graph BFS

- Grid BFS
  - zerojudge a982 (算長度)
  - zerojudge b059 (算長度)
  - CSES - Labyrinth (輸出一條路徑)
  - zerojudge a634 (輸出字典序最小)
  - TCIRC d093 (方格棋盤的最少轉彎數路線)
  - codeforces 329B. Biridian Forest
  - zeroJudge d406: 倒水時間
- Graph BFS
  - APCS 1081026 p3 闖關路線
  - CSES - Message Route (輸出路徑)

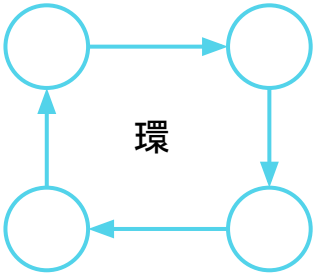
題單 - Graph BFS

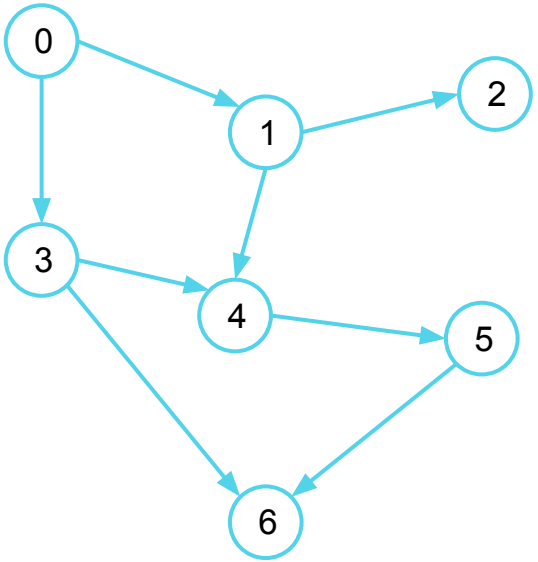
- 多源點 BFS
  - LeetCode 542. 01 Matrix
  - CSES - Monsters

拓墣排序
(Topological Sort)

Directed Acyclic Graph (DAG)

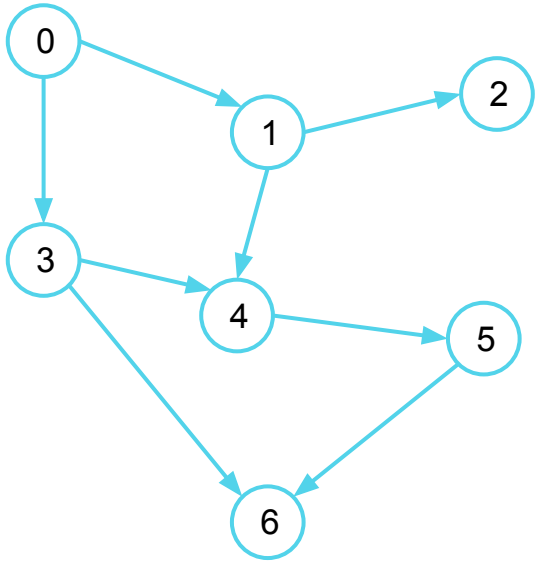
- 有向無環圖
  - 不存在環的有向圖





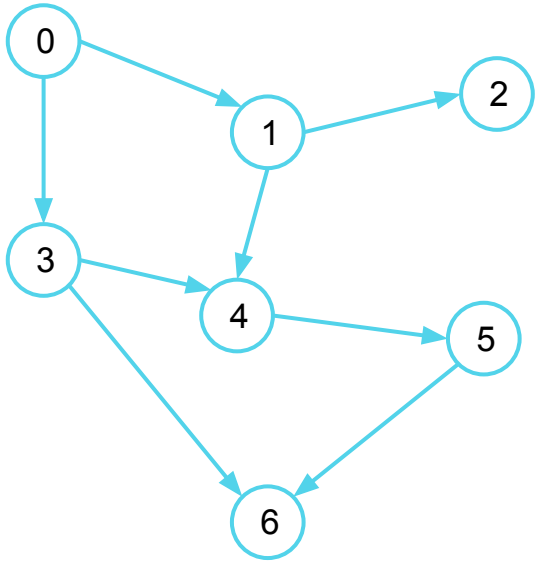
Topological Sort

- 在 DAG 上找一個序列, 使得任一條有向邊 (u, v), u 在 v 前面。
  - ex. 0 1 3 4 2 5 6



Topological Sort - Solution

- 維護每個點被邊指到的數量
  - in degree
- 依序拔掉 in degree 為 0 的點
  - 過程中要維護其他點的in degree,
    當 in degree 變成 0 就丟進 queue
- 直到做完所有的點, 拔掉的順序就是
  一個拓墣排序



	0	1	2	3	4	5	6
indeg	0	1	1	1	2	1	2

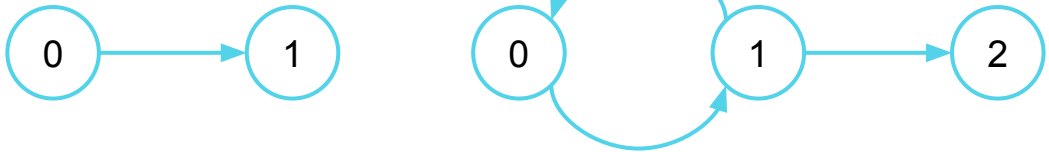
Topological Sort - Code

```
vector<int> topological_sort() {
    // 計算每個點的 in-degree
    calculate_indeg();

    // init queue
    queue<int> que;
    for (int i = 0; i < n; i++) {
        if (din[i] == 0) que.push(i);
    }

    // 不斷移除 in-degree 是 0 的點
    vector<int> topo_ans;
    while (que.size()) {
        int u = que.front();
        que.pop();
        topo_ans.push_back(u);
        for (int v : G[u]) {
            din[v]--;
            // in-degree 變成 0, 加入 queue
            if (din[v] == 0) que.push(v);
        }
    }
    return topo_ans;
}
```

```
int din[MAXN];

void calculate_indeg() {
    for (int u = 0; u < n; u++) {
        for (int i = 0; i < G[u].size(); i++) {
            int v = G[u][i];
            din[v]++;
        }
    }
}
```

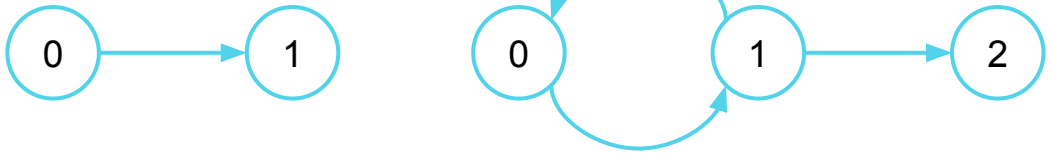

CSES - Course Schedule

- 題目敘述
  - 給一張有向圖，輸出任意拓撲排序
  - 無拓撲排序則輸出 IMPOSSIBLE
- 測資範圍
  - 節點數 n ≤ 100000，邊數 m ≤ 200000



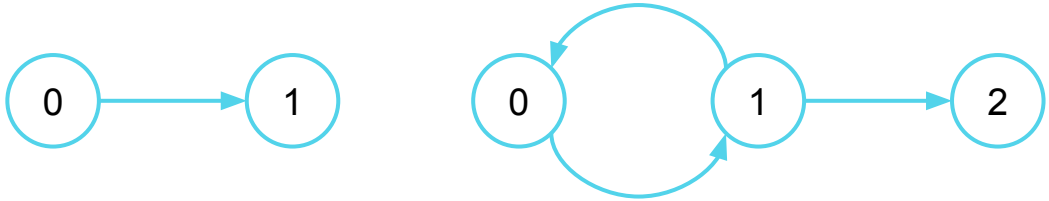
Zerojudge b583: 一個環

- 題目敘述
  - 給一個有向圖，判斷是否存在有向環
- 測資範圍
  - 節點數 n ≤ 1000，邊數 m ≤ 100000



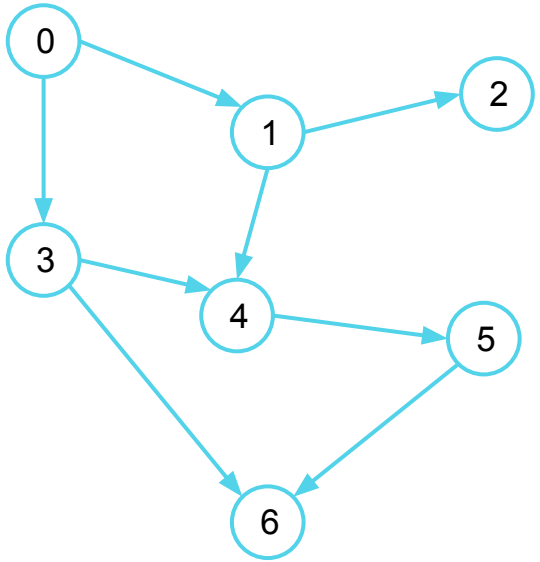
Zerojudge b583: 一個環

- 作法
  - 執行用 queue 找 topological order 的演算法
  - 若存在一個環，在環上的點不會被放到queue 裡面
  - 紀錄進入過 queue 的點數量



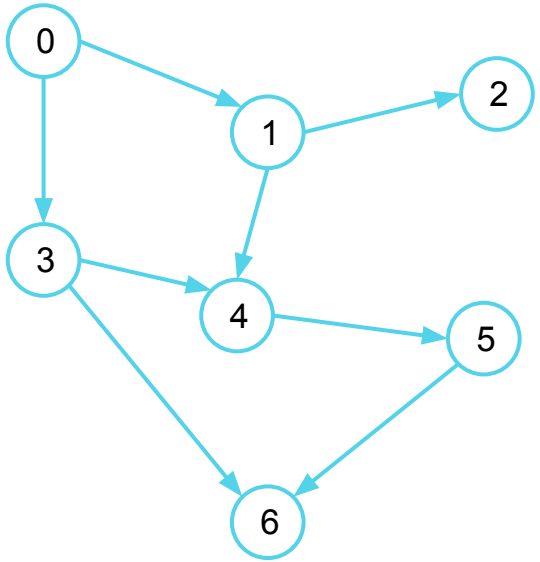
CSES - Longest Flight Route

- 題目敘述
  - 找 DAG 最長路
- 測資範圍
  - 節點數 n ≤ 100000，邊數 m ≤ 200000



CSES - Longest Flight Route

- 作法 : DAG 上面 DP
  - dp(v) : 以 v 結尾的最長路徑長度
  - dp(v) = max{ dp(u) + 1 | u 有邊走到 v }



題單 - Topological Sort

- 找到 Topological Ordering
  - uva 10305 : 輸出任意一組 topological ordering
  - leetcode 210. Course Schedule II
  - zerojudge a552 (最小字典序 topological order)
- 判斷是否存在有向環
  - zerojudge b583
  - CSES - Course Schedule

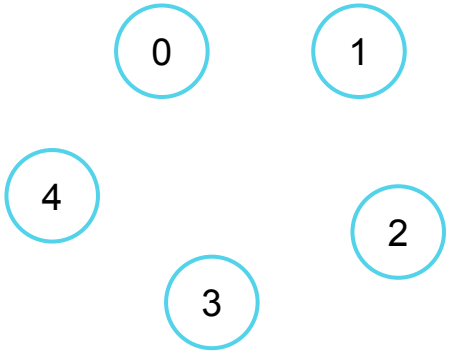
題單 - Topological Sort

- DAG DP
  - CSES - Longest Flight Route (DAG 最長路徑)
  - CSES - Game Routes (DAG 路徑數量)
- Hard
  - codeforces 510C. Fox And Names
  - leetcode 1591. Strange Printer II

並査集
(Disjoint Set)

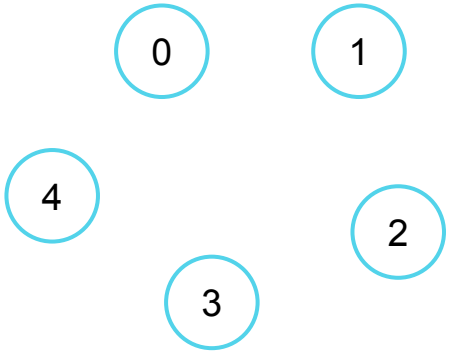
動態連通性查詢

- 題目敘述
  - 一開始有 n 個點編號 0 ~ (n - 1)，接下來依序有 q 個指令
  - 指令有兩奘
    - 1 u v : 連接 u, v 兩個節點
    - 2 u v : 檢查 u, v 兩點是否可以直接或是間接到達
- 測資範圍
  - 節點數 $n \le 10^6$，查詢數 $q \le 10^6$



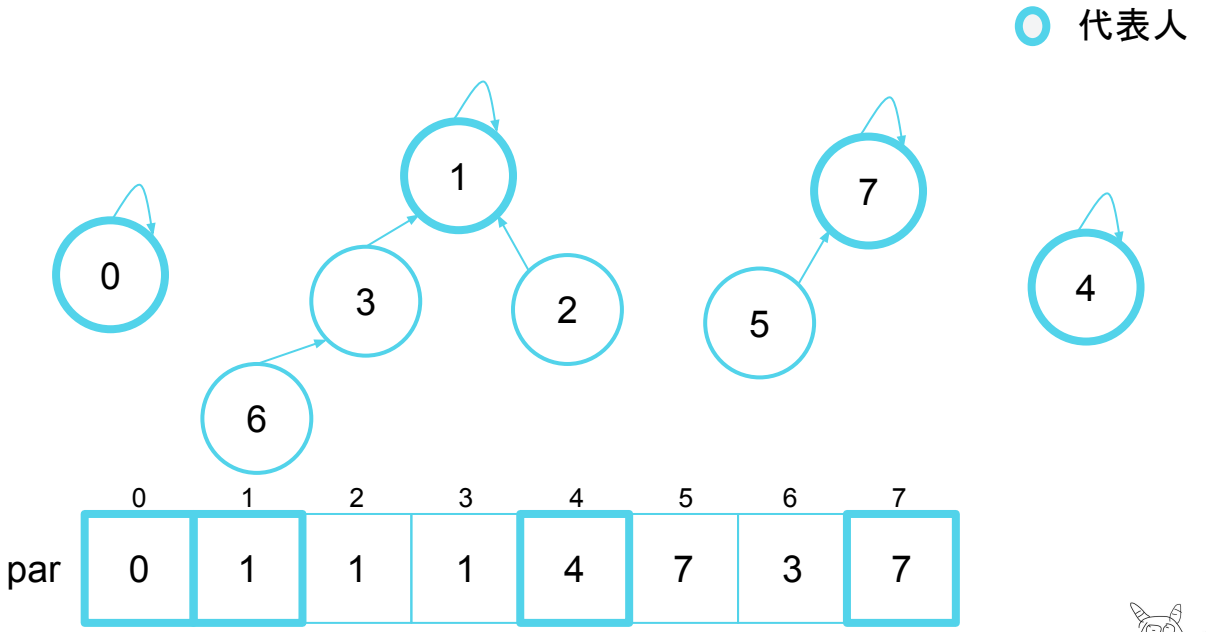
動態連通性查詢

- 暴力作法
  - 每次都重新跑 DFS / BFS
  - 時間複雜度 O(q * (n+m))



Disjoint Set 表示法

- 每個組別有一個代表人 (老大)
- 要檢查兩個人是否在同一個組別，只要檢查代表人是否相同



Disjoint Set 初始化

- 一開始每個點都自己一組

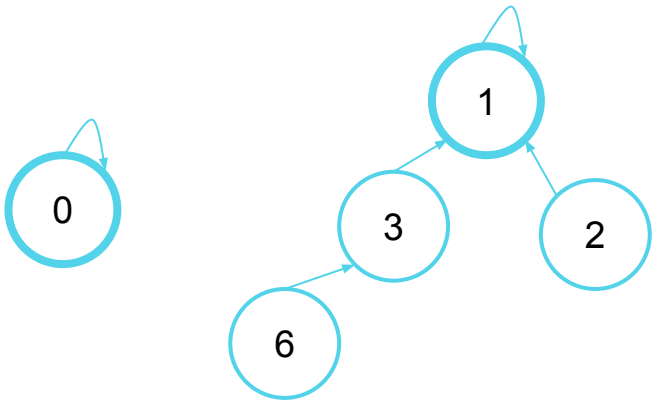
```
int par[MAXN];

void dsu_init(int n) {
    for (int i = 0; i < n; i++) {
        par[i] = i;
    }
}
```


Disjoint Set Operations

- `find(x)`
  - 找到 `x` 所處在集合的代表人

- `merge(u, v)`
  - 將 `u`, `v` 合起來，若本來就在同一個集合則不動作

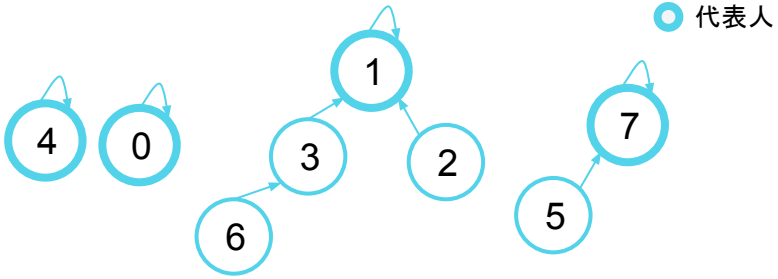
Disjoint Set find - Code



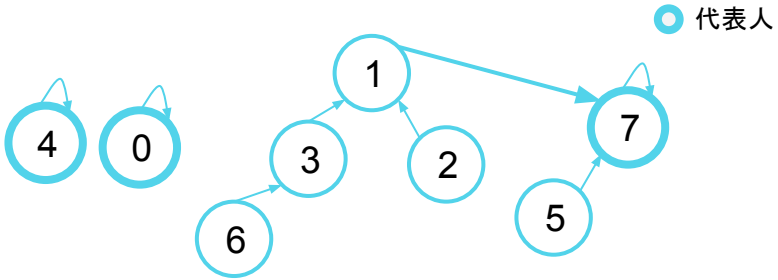
代表人

```
int find(int x) {
    if (par[x] == x) {
        return x;
    } else {
        return find(par[x]);
    }
}
```

Disjoint Set merge - Code




```
merge(3, 5) -> find(3) = 1
               find(5) = 7
```

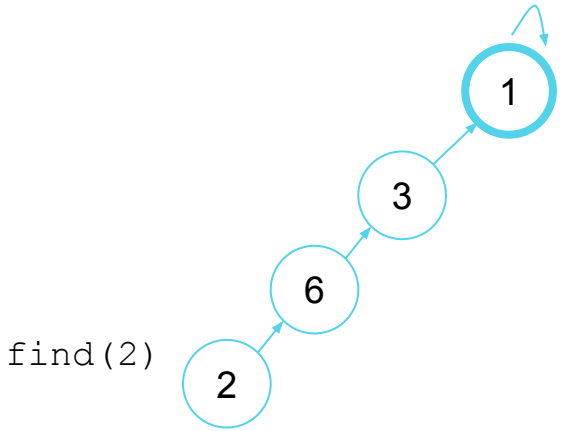



```
void merge(int u, int v) {
  int x = find(u);
  int y = find(v);
  if (x != y)
     par[x] = y;
}
```

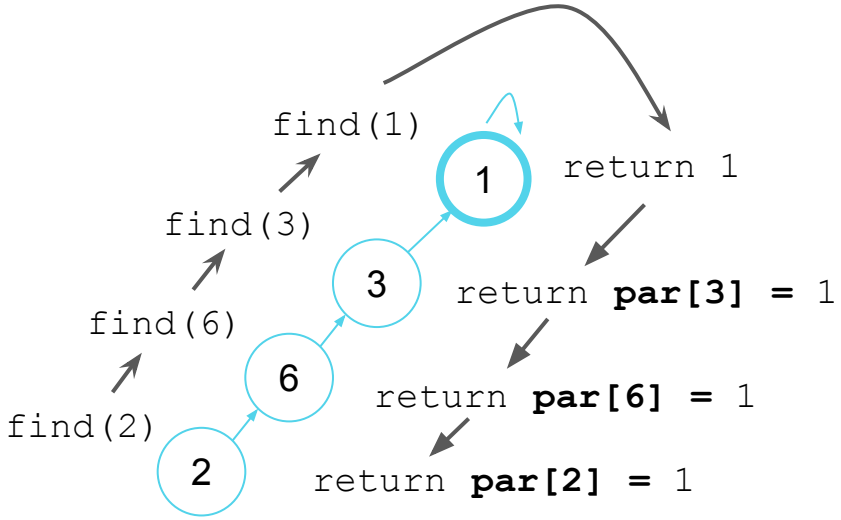

Disjoint Set Complexity

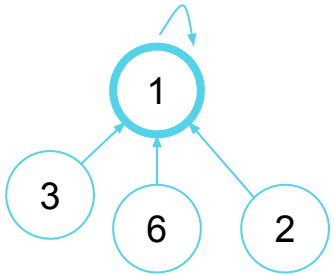
- `find`
  - 時間複雜度 `O(h)`
    h 表示樹的高度

- `merge`
  - 由 `find` 組成
  - 因此時間複雜度 `O(h)`
    h 表示樹的高度



Disjoint Set find 優化 - 路徑壓縮





Disjoint Set find 路徑壓縮 - Code

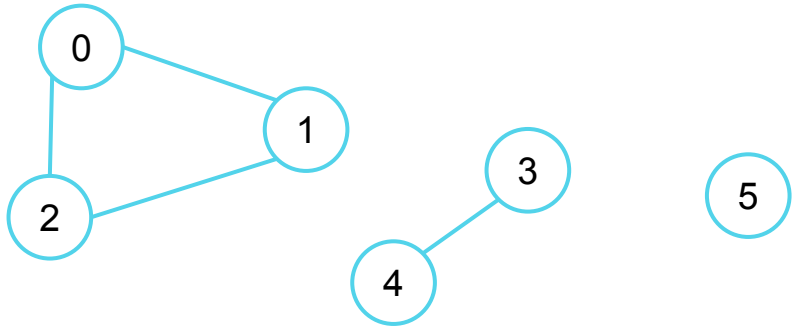
- find 和 merge 的時間複雜度都變成 O(log* n)
- log* x 表示 x 一直取 log 幾次後會小於 0，多數情況下可以想成小於 5 的數字

```
int find(int x) {
    if (par[x] == x) {
        return x;
    } else {
        return par[x] = find(par[x]);
    }
}
```


ZeroJudge d813: 10583 - Ubiquitous Religions

- 題目敘述
  - 給一個無向圖的，問總共有幾個連通塊
    (使用 disjoint set)

- 測資範圍
  - 節點數 n ≤ 50000



Disjoint Set 順便維護 size

- 若合併時候按照 size 小的在下面，find 和 merge 的時間複雜度都變成 O(α(n))，α(n) 為反阿克曼函數
- 這個技巧也稱作啟發式合併

```
int sz[MAXN]; // 要初始化成 1

void merge(int u, int v) {
    int x = find(u);
    int y = find(v);
    if (sz[x] > sz[y]) swap(x, y);
    if (x != y) {
        par[x] = y;
        sz[y] += sz[x];
        // sz[x] = 0;
    }
}
```


題單 - Disjoint Set

- zerojudge a445
- zerojudge d813 (計算連通塊數量)
- zerojudge d831 (維護 size)
- leetcode 399. Evaluate Division
- CSES - Road Construction