

資料結構

temmie

1. 前言、名詞解釋

2. 圖的儲存

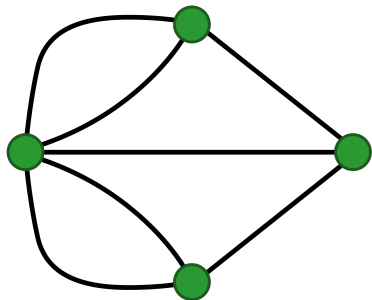
3. DFS

4. BFS

5. 回溯解

前言、名詞解釋

七橋問題



- 有辦法經過所有的橋嗎？
- 上面的「圖」是怎麼構成的呢？

名詞解釋

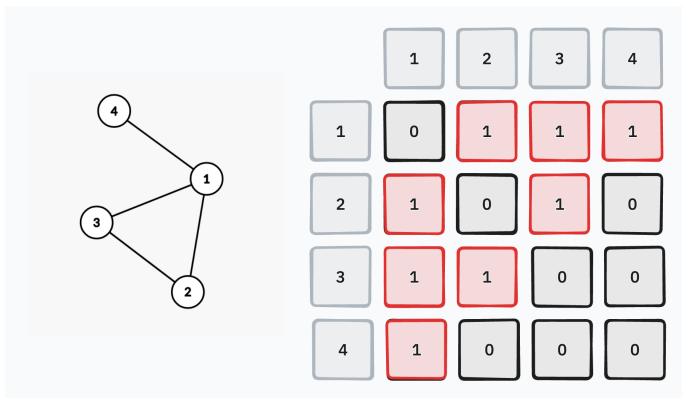
- 節點 (Vertex, V)
- 邊 (Edge, E) : 邊會將兩個節點相連
- 圖 (Graph, G) : 由很多邊和節點組成
- 有向圖 (Directed Graph) : 邊只能讓 u 節點走向 v 節點
- 無向圖 (Undirected Graph) : 邊可以讓 u 節點和 v 節點互通

名詞解釋

- 入度 (In-degree, $\deg^-(u)$) : 在有向圖中指向該節點的數量
- 出度 (Out-degree, $\deg^+(u)$) : 在有向圖中節點向外走的數量
- 度數 (Degree, $\deg(u)$) : 入度 + 出度
- 路徑 (Path) : 從 u 節點走向 v 節點所經歷的邊
- 環 (Cycle) : 從 u 節點走向 u 節點所經歷的邊

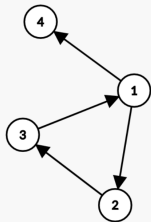
圖的儲存

相鄰矩陣



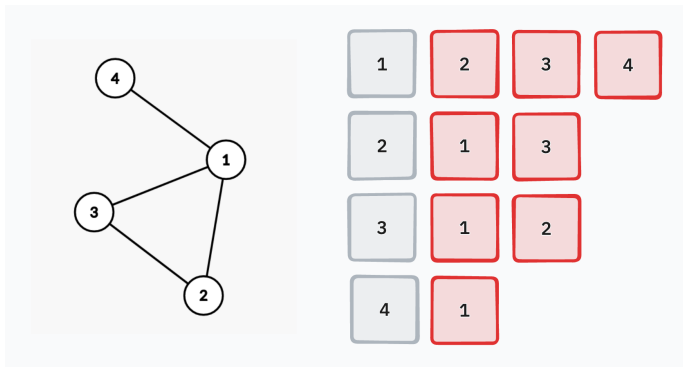
- 尋找 u 節點的所有相鄰節點： $O(V)$
- 遍歷整張圖： $O(V^2)$
- 空間複雜度： $O(V^2)$

相鄰矩陣



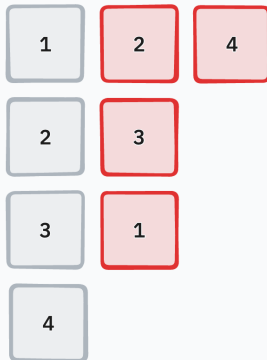
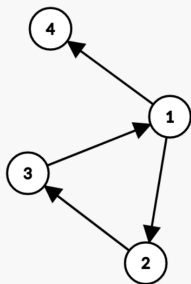
	1	2	3	4
1	0	1	0	1
2	0	0	1	0
3	1	1	0	0
4	0	0	0	0

相臨串列



- 尋找 u 節點的所有相鄰節點： $O(deg^+)$
- 遍歷整張圖： $O(E)$
- 空間複雜度： $O(E)$

相臨串列



實作：相鄰矩陣

```
1 int n, m; // 有 n 個節點，m 條邊
2 int u, v;
3 int G[500][500]; // 相鄰矩陣
4
5 for (int i=0 ; i<m ; i++){
6     cin >> u >> v;
7     G[u][v]=1;
8     // G[v][u]=1; // 如果是無向圖的話就加上此行
9 }
10
```

實作：相鄰串列

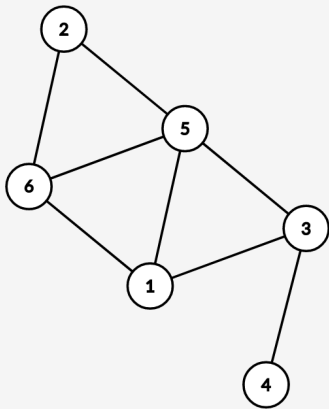
```
1 int n, m; // 有 n 個節點，m 條邊
2 int u, v;
3 vector<vector<int>> G(500); // 相鄰串列
4
5 for (int i=0 ; i<m ; i++){
6     cin >> u >> v;
7     G[u].push_back(v);
8     // G[v].push_back(u); // 如果是無向圖的話就加上此行
9 }
```

DFS

- 深度優先搜尋（Depth-First-Search，DFS），一種遍歷圖的方法
- 顧名思義：我們會將**深度作為最先搜尋的對象**
- 步驟如下
 - ① 選擇起點當作**目前的節點**
 - ② 紀錄目前的節點走過
 - ③ 前往其中一個沒走過的相鄰節點當作目前的節點
 - ④ 如果全部的相鄰節點都走完就將目前的節點設為父節點
 - ⑤ 如果所有節點都走過就結束

DFS

- 讓我們以這張圖做個例子



實作：DFS

```
1 vector<vector<int>> G(500); // 相鄰串列
2 bitset<500> vis; // 紀錄是否走過第 i 點
3
4 void dfs(int now){
5     vis[now]=1; // 2. 紀錄目前的節點走過
6     for (auto x : G[now]){
7         if (vis[x]==0){ // 3. 前往沒走過的相鄰節點
8             dfs(x);
9         }
10    }
11    return; // 4. 全部的相鄰節點都走完就將目前節點設為父節點
12 }
13
14 dfs(1); // 1. 選擇起點當作目前節點
```

例題

Building Roads

題目連結

有 n 個城市 m 條道路，請問需要新增幾條道路才能讓所有城市連通。
請輸出數量以及新增的道路。

例題

Building Roads

題目連結

有 n 個城市 m 條道路，請問需要新增幾條道路才能讓所有城市連通。
請輸出數量以及新增的道路。

- Tip1：已經有連通的城市還需要新增道路嗎？

例題

Building Roads

題目連結

有 n 個城市 m 條道路，請問需要新增幾條道路才能讓所有城市連通。
請輸出數量以及新增的道路。

- Tip1：已經有連通的城市還需要新增道路嗎？
- Tip2：兩個不連通的程式要新增幾條道路才能連通？

例題

```
1 vector<int> ans;
2
3 for (int i=1 ; i<=n ; i++){
4     // 一個城市還沒被遍歷，就記錄他並且走過所有跟他連通的城市
5     if (vis[i]){
6         ans.push_back(i);
7         dfs(i);
8     }
9 }
10
11 cout << ans.size()-1 << "\n";
12 for (int i=0 ; i<ans.size()-1 ; i++){
13     cout << ans[i] << " " << ans[i+1] << "\n";
14 }
```

例題

Counting Rooms

題目連結

給你一個大小為 $n \times m$ 的房間，@代表為地板，#代表為牆壁，在一個地板上可以上下左右走向另一個地板，並視為在同一個房間。
請問房間的數量？

例題

Counting Rooms

題目連結

給你一個大小為 $n \times m$ 的房間，@代表為地板，#代表為牆壁，在一個地板上可以上下左右走向另一個地板，並視為在同一個房間。
請問房間的數量？

- Tip1：請先想想看，這要怎麼轉換成一張圖呢？

例題

Counting Rooms

題目連結

給你一個大小為 $n \times m$ 的房間，@代表為地板，#代表為牆壁，在一個地板上可以上下左右走向另一個地板，並視為在同一個房間。

請問房間的數量？

- Tip1：請先想想看，這要怎麼轉換成一張圖呢？
- Tip2：這是一個二維的平面，請問該如何表示目前的節點？

例題

```
1 bool vis[1005][1005];
2
3 bool out(int x, int y){ // 檢查下一個座標是否出界
4     return x<0 || x>n || y<0 || y>m;
5 }
6
7 void dfs(int x, int y){
8     vis[x][y]=1;
9     if (out(x+1, y)==0 && v[x+1][y]=='.') dfs(x+1, y);
10    if (out(x-1, y)==0 && v[x-1][y]=='.') dfs(x-1, y);
11    if (out(x, y+1)==0 && v[x][y+1]=='.') dfs(x, y+1);
12    if (out(x, y-1)==0 && v[x][y-1]=='.') dfs(x, y-1);
13    return;
14 }
```

例題

```
1 bool vis[1005][1005];
2 const int mx[4]={1, 0, -1, 0};
3 const int my[4]={0, 1, 0, -1};
4
5 bool out(int x, int y){ // 檢查下一個座標是否出界
6     return x<0 || x>n || y<0 || y>m;
7 }
8
9 void dfs(int x, int y){
10     vis[x][y]=1;
11     for (int i=0 ; i<4 ; i++){
12         if (out(x+mx[i], y+my[i])==0 &&
13             v[x+mx[i]][y+my[i]]=='.') dfs(x+mx[i], y+my[i]);
14     }
15     return;
16 }
```

BFS

- 深度優先搜尋（Breadth-First-Search，BFS），一種遍歷圖的方法
- 同樣地，會以廣度為最優先的順序。
- 步驟如下
 - ① 選擇起點當作目前的節點，**紀錄走過**後將處理順序設為 1
 - ② 紀錄目前的節點走過
 - ③ 將所有相鄰節點依序紀錄處理順序後，**並直接紀錄走過**
 - ④ 尋找下一個處理順序的節點
 - ⑤ 如果所有節點都走過就結束，否則重複第二步驟

比較

- DFS 跟 BFS 都可以遍歷整張圖
- DFS 相對更好實作，也比較方便一些動態規劃的題目
- BFS 比較難實作一些，不過擁有了**最短路**的性質

回溯解

回溯解

- 在一些問題中，題目將會要求構造出一種解法。
- 這個問題我們可以透過**逆推**達成。
- 也就是紀錄**這個結果是由誰推過來的**

例題

CLabyrinth

題目連結

給你一個大小為 $n \times m$ 的房間，@代表為地板，#代表為牆壁，在一個地板上可以上下左右走向另一個地板，並且給予一個點 **A** 代表起點，**B** 代表終點。

請判斷是否有辦法從起點走向終點，如果可以，則輸出 **"Yes"**，並且輸出**最短路徑長度**以及用 **LRUD** 代表**左右上下**構造一任意一組解。

例題

CLabyrinth

題目連結

給你一個大小為 $n \times m$ 的房間，@代表為地板，#代表為牆壁，在一個地板上可以上下左右走向另一個地板，並且給予一個點 **A** 代表起點，**B** 代表終點。

請判斷是否有辦法從起點走向終點，如果可以，則輸出 **"Yes"**，並且輸出**最短路徑長度**以及用 **LRUD** 代表**左右上下**構造一任意一組解。

- Tip1：怎麼知道是否有辦法從起點走到終點？

例題

CLabyrinth

題目連結

給你一個大小為 $n \times m$ 的房間，@代表為地板，#代表為牆壁，在一個地板上可以上下左右走向另一個地板，並且給予一個點 A 代表起點，B 代表終點。

請判斷是否有辦法從起點走向終點，如果可以，則輸出 "Yes"，並且輸出**最短路徑長度**以及用 LRUD 代表**左右上下**構造一任意一組解。

- Tip1：怎麼知道是否有辦法從起點走到終點？
- Tip2：如果是最短路徑，那們要用 DFS 還是 BFS ？

例題

CLabyrinth

題目連結

給你一個大小為 $n \times m$ 的房間，@代表為地板，#代表為牆壁，在一個地板上可以上下左右走向另一個地板，並且給予一個點 A 代表起點，B 代表終點。

請判斷是否有辦法從起點走向終點，如果可以，則輸出 "Yes"，並且輸出**最短路徑長度**以及用 LRUD 代表**左右上下**構造一任意一組解。

- Tip1：怎麼知道是否有辦法從起點走到終點？
- Tip2：如果是最短路徑，那們要用 DFS 還是 BFS ？
- Tip3：如果知道 A 是從 B 轉移的話，要怎麼紀錄資訊進行反推？

例題

```
1 int parent[1005][1005];  
2  
3 void dfs(int x, int y){  
4     vis[x][y]=1;  
5     for (int i=0 ; i<4 ; i++){  
6         if (out(x+mx[i], y+my[i])==0 &&  
7             v[x+mx[i]][y+my[i]]=='.'){  
8  
9             dfs(x+mx[i], y+my[i]);  
10        }  
11    }  
12    return;  
13 }
```