演算法與程式解題實務

Mong-Jen Kao (高孟駿)

Monday 18:30 – 21:20

Graph 圖

- Graph 泛指一群**點 (Vertex / Node)** 以及它們之間的**邊 (Edge)** 所形成的關係圖.
  - Ex. 道路交通網; 交友關係圖; 捷運地圖; 工作分配圖等等

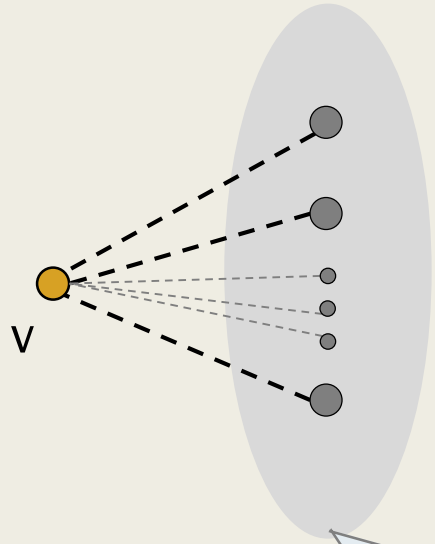
使用 adjacency list 來儲存圖

往後我們希望大家盡量練習用這個方式來儲存圖。

- 儲存圖的方式主要有兩種,

  其中一種是將每個點的鄰居列表(Adjacency List) 儲存下來

```
struct nodes
{
    int adj_list[MaxN], *end_of_list;
} nodes[ MaxN ];
```



v 的鄰居列表
(adjacency list)

- 每個點的 Adjacency List 裡,

  儲存的是所有與它相鄰的點的編號列表

DFS 深度優先探索

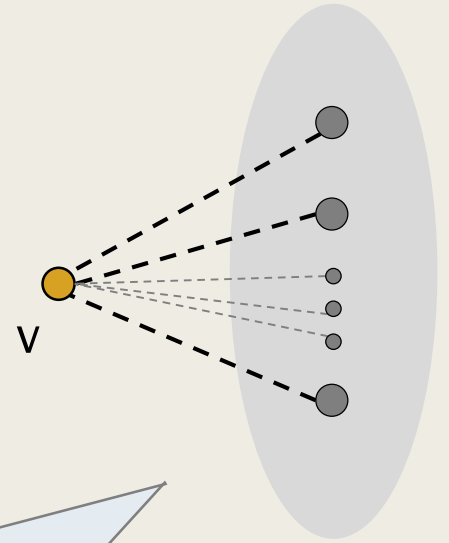
這是 DFS 的基礎模板。

- DFS 是用遞迴的方式來探索圖,

  其精神與本週的迷宮探索方法完全一樣

  - 走到一個點的時候,
    我們考慮它所有的鄰居 (對應到所有可以走的方向)
  - 若是還沒有走過, 就遞迴走進去.

```
void dfs( int v ) {
    set visited[v] to be true;
    for each u ∈ adj(v),
        if visited[u] is false, then
            dfs(u);
}
```



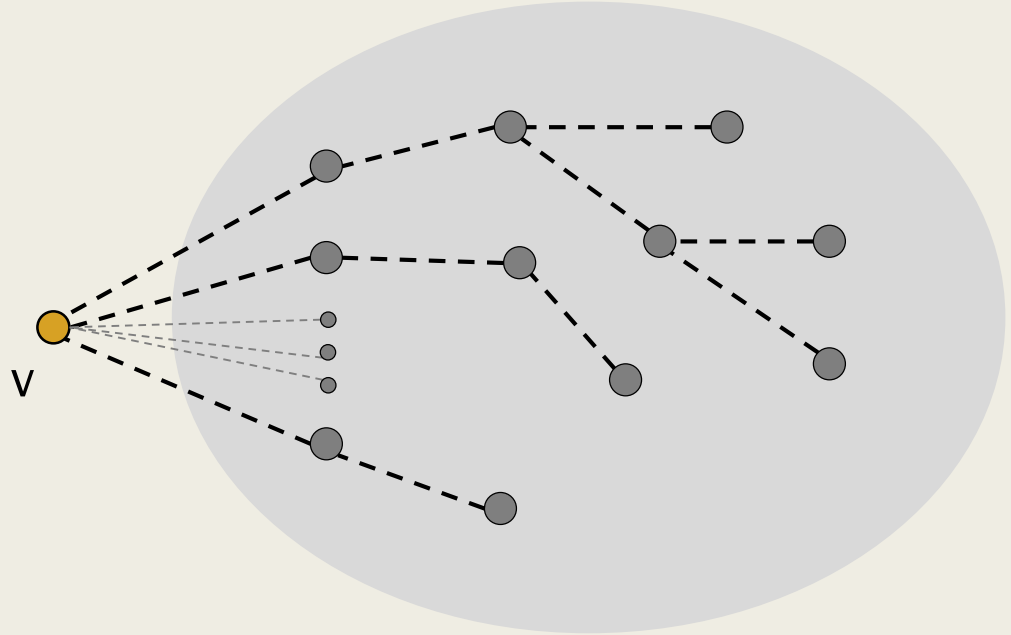
考慮 v 的所有鄰居,
若還沒走過, 就遞迴走進去

DFS 深度優先探索

一開始呼叫 dfs 前, 必須先將
所有點的 visited 初始化為 false。

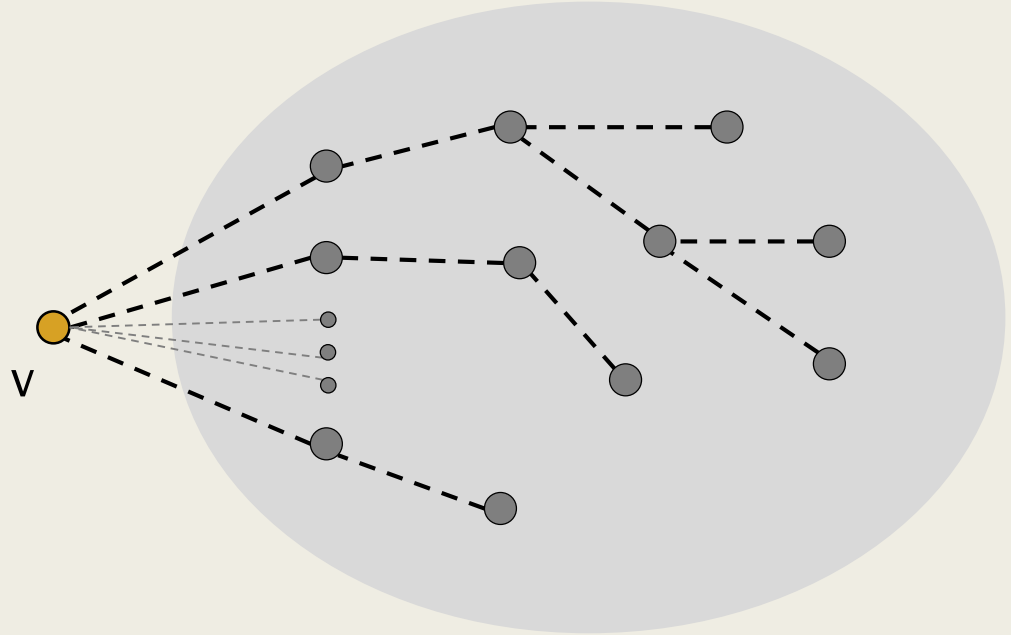
- DFS 可以保証探索到所有走得到的點
  - 在 dfs(v) 執行完畢後,

    所有由 v 可以到達的點, 都會被標記為可到達 (visited 為 true)



DFS 深度優先探索

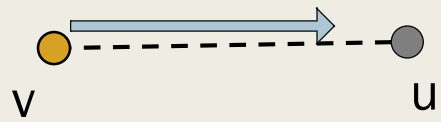
- DFS 的探索過程, 其實已經對應到從起點至各個點的路徑
  - 我們可以用一個欄位 (例如 fromm) 來紀錄各個點的來源點
  - 依照所紀錄的來源點, 即對應到完整的路徑



紀錄 DFS 的過程

- 藉由紀錄每個點的來源點 (是由誰走到它的),
  我們可以完整紀錄 DFS 的過程

```
void dfs( int v ) {
     set visited[v] to be true;
     for each u ∈ adj(v),
        if visited[u] is false, then
           set fromm[u] to be v;
           dfs(u);
}
```



u 的來源點為 v

印出路徑

- 有了來源點資訊後, 要印出路徑很容易 (!)
  - 舉例來說, 若 v 為 DFS 的起點

    要印出 v -> w 的路徑,

    我們只要先印出 v 至 w 的來源點的路徑, 再印出 w 即可

```
void print_path( int w, int v ) {
    if w != v, then
        print_path( fromm[w], v );
    print w;
}
```