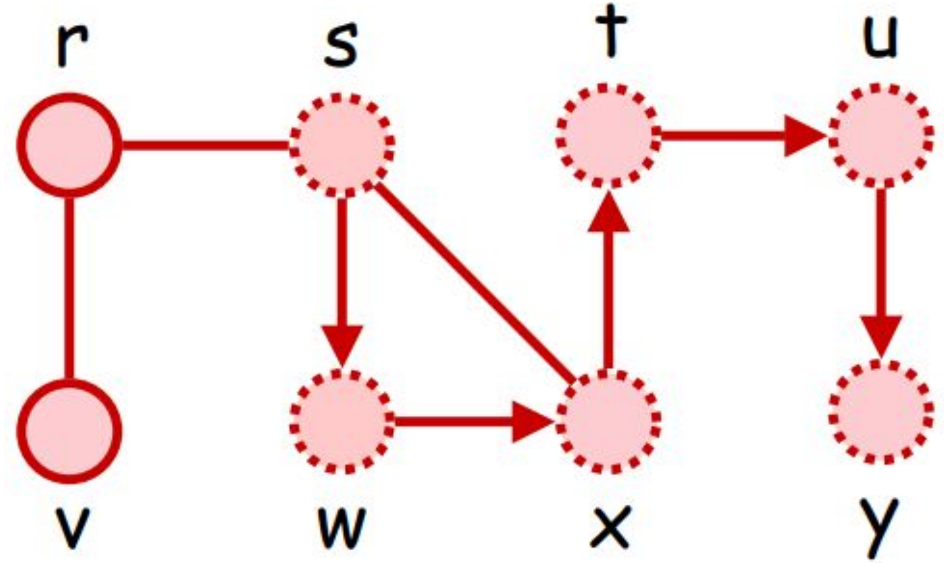
Graph Traversal

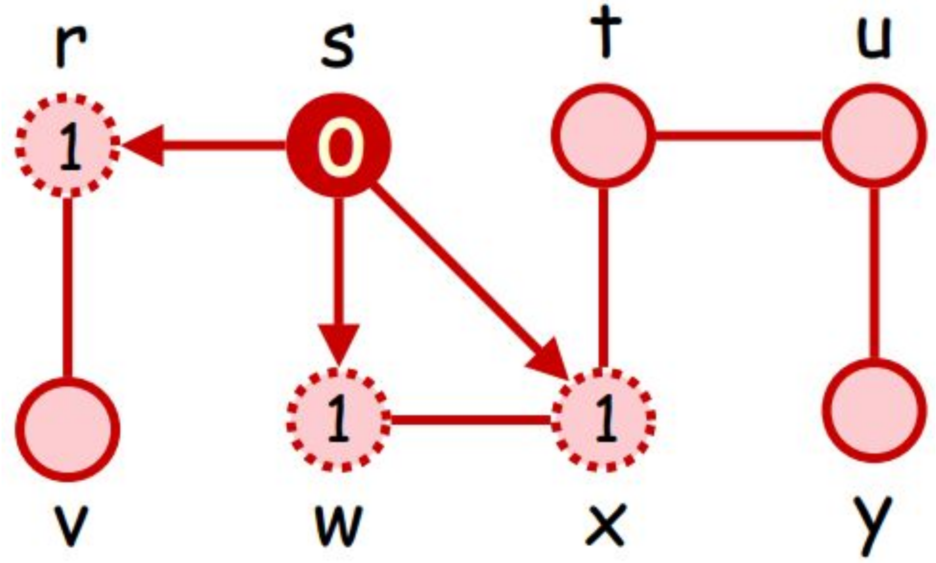
Graph Traversal: DFS, BFS

- Depth First Search (DFS), 深度優先搜尋
  - 將某條岔路探索到底後, 再探索其他岔路



Graph Traversal: DFS, BFS

- Breadth First Search (BFS), 廣度優先搜尋
  - 同時探索所有岔路，並記錄每條岔路探索到哪



DFS, BFS - Implementation

1. 節點目前狀態：未造訪、已造訪
2. 造訪新的節點：只能造訪「未造訪」的節點

DFS - Implementation

```
vector< vector<int> > adjacency_list(n); // graph
vector<bool> visited(n, false); // state of vertices
```

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```

利用遞迴，將某條岔路探索到底後，再探索其他岔路

DFS - Implementation

```
vector< vector<int> > adjacency_list(n); // graph
vector<bool> visited(n, false); // state of vertices
```

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```

更新節點狀態

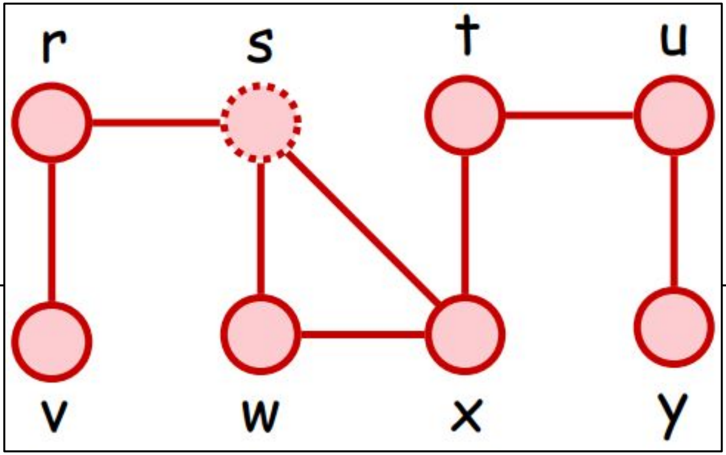
根據節點狀態決定是否造訪

● finished

◌ discovered

DFS - Example

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s

function call: dfs(s)

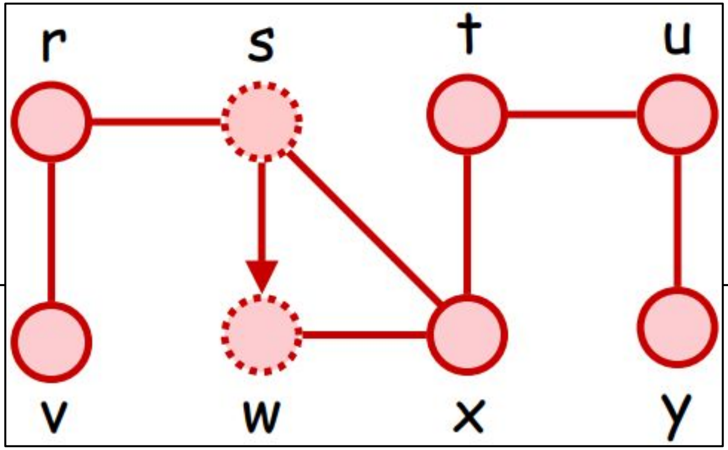
In adj[s], find vis[w] = false, call dfs(w)

● finished

◌ discovered

DFS - Example

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w

function call: dfs(s) => dfs(w)

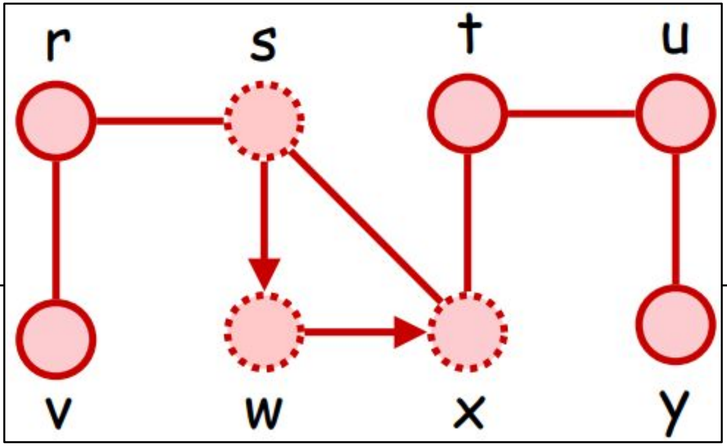
In adj[w], find vis[x] = false, call dfs(x)

● finished

◌ discovered

DFS - Example

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x

function call: dfs(s) => dfs(w) => dfs(x)

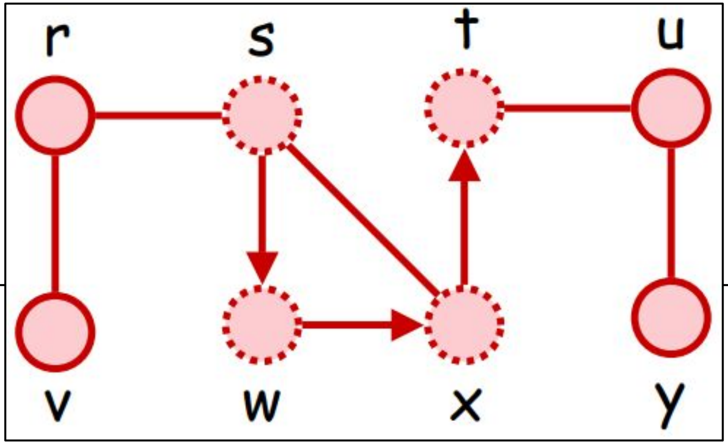
In adj[x], find vis[t] = false, call dfs(t)

● finished

◌ discovered

DFS - Example

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t

function call: dfs(s) => dfs(w) => dfs(x) => dfs(t)

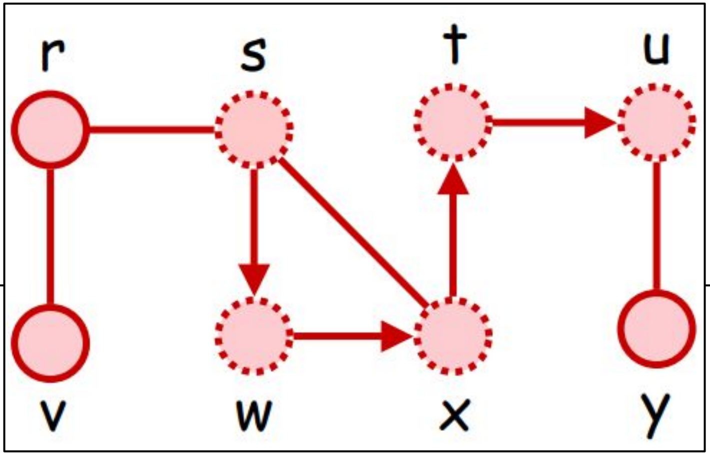
In adj[t], find vis[u] = false, call dfs(u)

DFS - Example

● finished

◌ discovered

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t , u

function call: dfs(s) => dfs(w) => dfs(x) => dfs(t) => dfs(u)

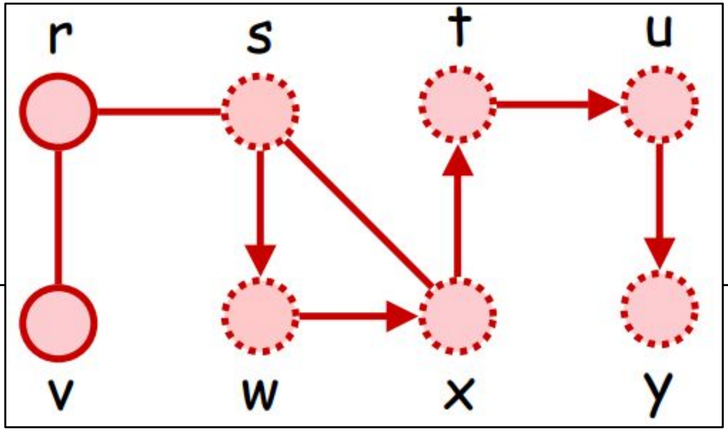
In adj[u], find vis[y] = false, call dfs(y)

DFS - Example

● finished

◌ discovered

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t , u , y

function call: dfs(s) => dfs(w) => dfs(x) => dfs(t) => dfs(u) => dfs(y)

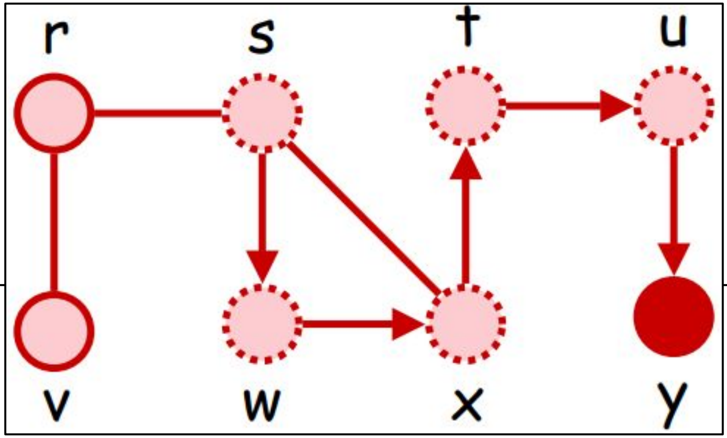
In adj[y], find all neighbors are visited, dfs(y) terminate

DFS - Example

● finished

◌ discovered

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t , u , y

function call: dfs(s) => dfs(w) => dfs(x) => dfs(t) => dfs(u)

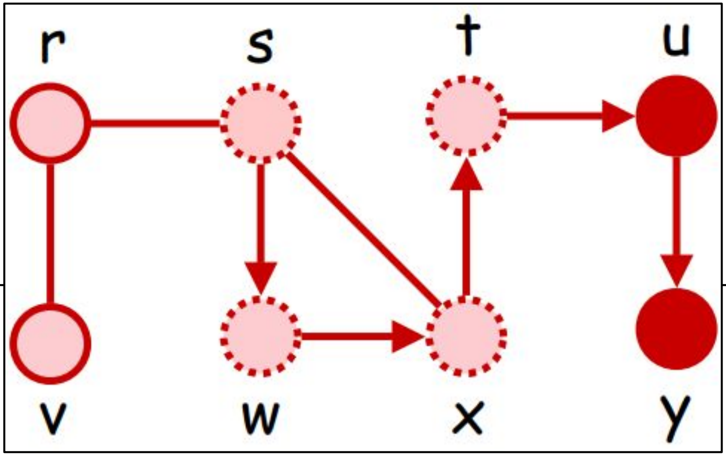
In adj[u], find all neighbors are visited, dfs(u) terminate

● finished

◌ discovered

DFS - Example

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t , u , y

function call: dfs(s) => dfs(w) => dfs(x) => dfs(t)

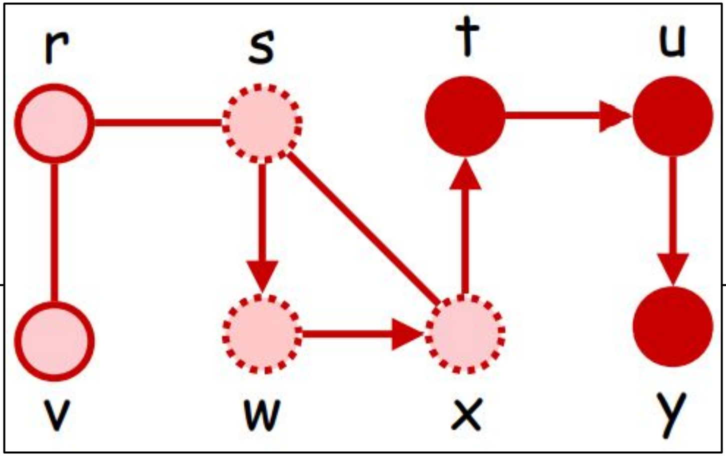
In adj[t], find all neighbors are visited, dfs(t) terminate

DFS - Example

● finished

◌ discovered

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t , u , y

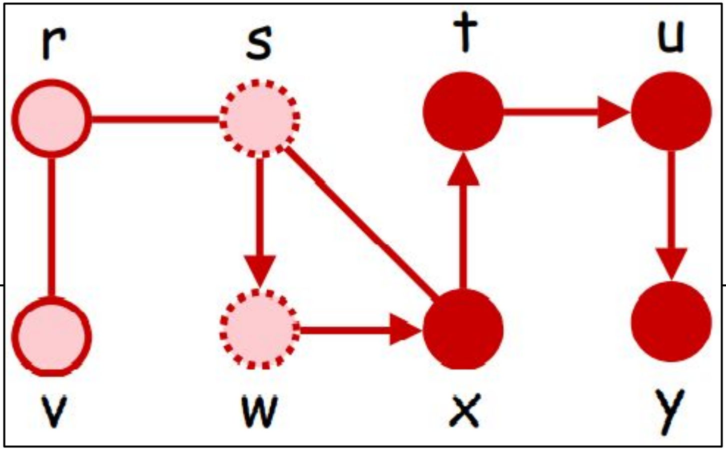
function call: dfs(s) => dfs(w) => dfs(x)

In adj[x], find all neighbors are visited, dfs(x) terminate

DFS - Example

● finished
◌ discovered

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t , u , y

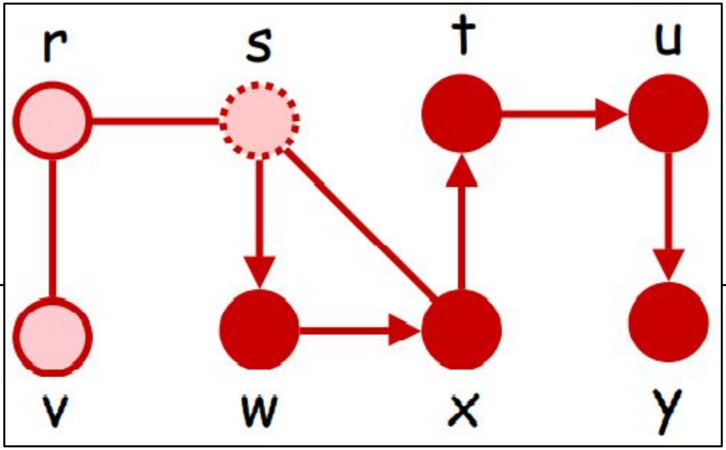
function call: dfs(s) => dfs(w)

In adj[w], find all neighbors are visited, dfs(w) terminate

DFS - Example

●	finished
◌	discovered

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t , u , y

function call: dfs(s)

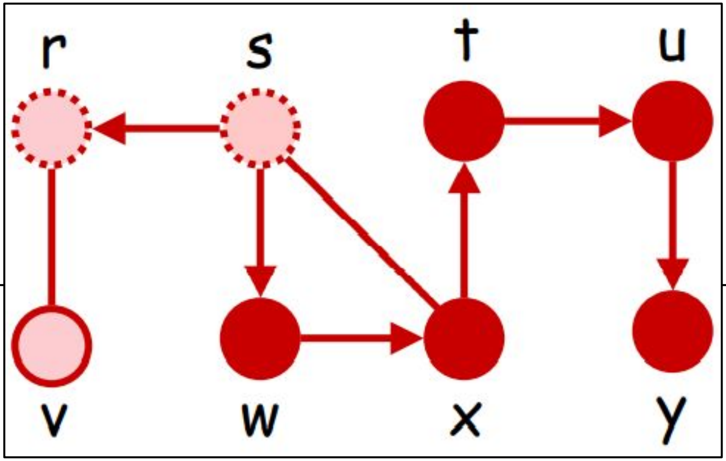
In adj[s], find vis[r] = false, call dfs(r)

● finished

◌ discovered

DFS - Example

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t , u , y , r

function call: dfs(s) => dfs(r)

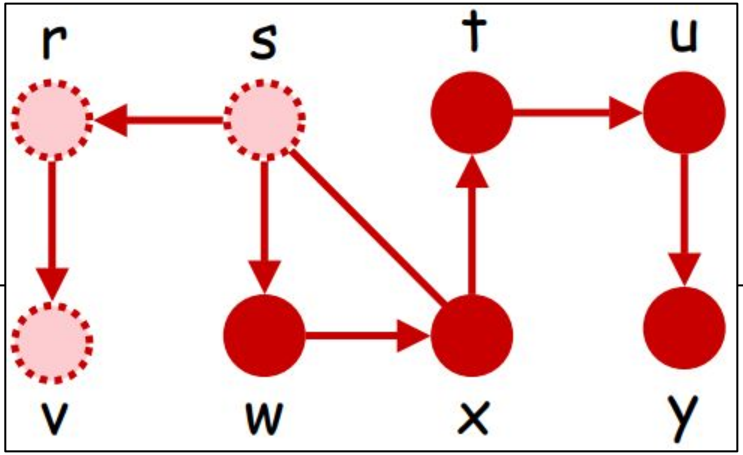
In adj[r], find vis[v] = false, call dfs(v)

● finished

◌ discovered

DFS - Example

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t , u , y , r , v

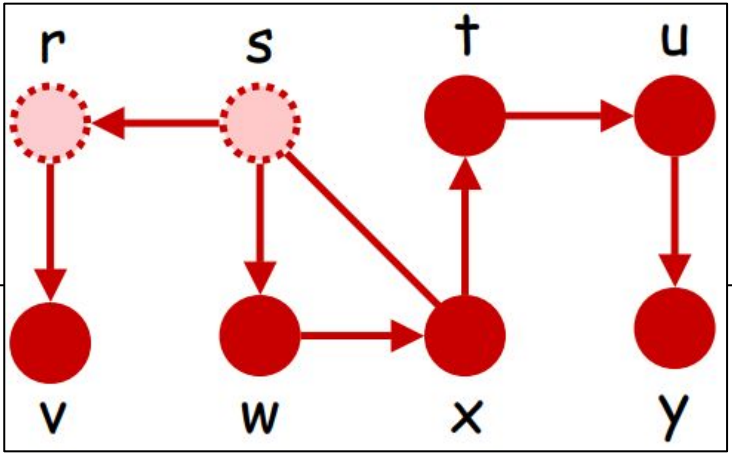
function call: dfs(s) => dfs(r) => dfs(v)

In adj[v], find all neighbors are visited, dfs(v) terminate

DFS - Example

●	finished
◌	discovered

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t , u , y , r , v

function call: dfs(s) => dfs(r)

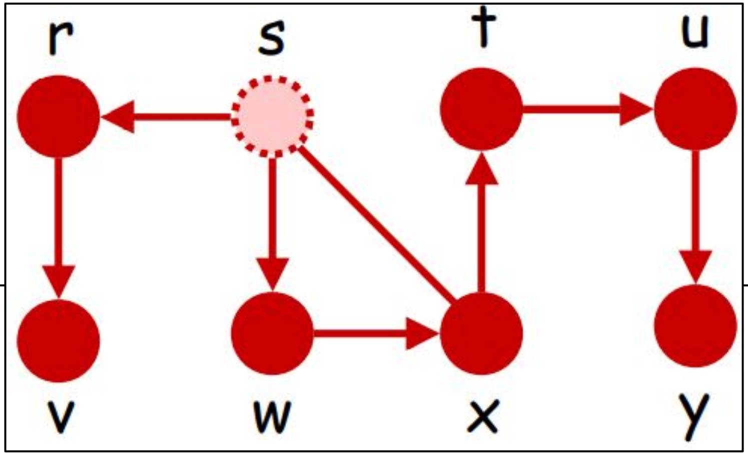
In adj[r], find all neighbors are visited, dfs(r) terminate

● finished

◌ discovered

DFS - Example

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t , u , y , r , v

function call: dfs(s)

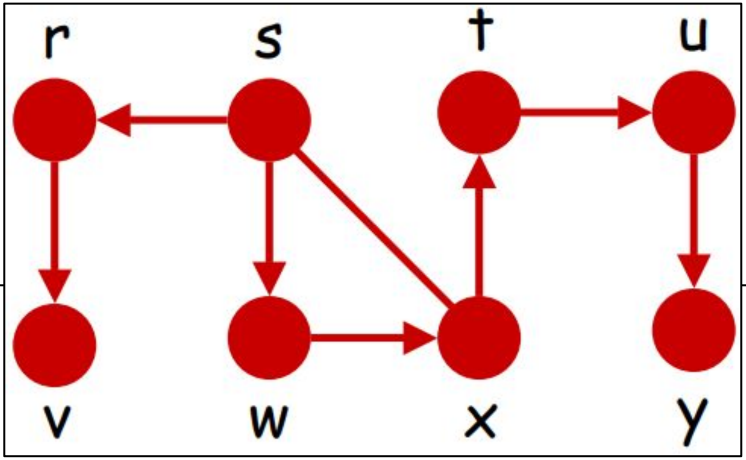
In adj[s], find all neighbors are visited, dfs(s) terminate

● finished

◌ discovered

DFS - Example

```
void dfs(int u, const vector< vector<int> > &adj, vector<bool> &vis)
{
    vis[u] = true;
    for (auto v : adj[u])
        if (!vis[v]) dfs(v, adj, vis);
}
```



visit order: s , w , x , t , u , y , r , v

function call: (empty)

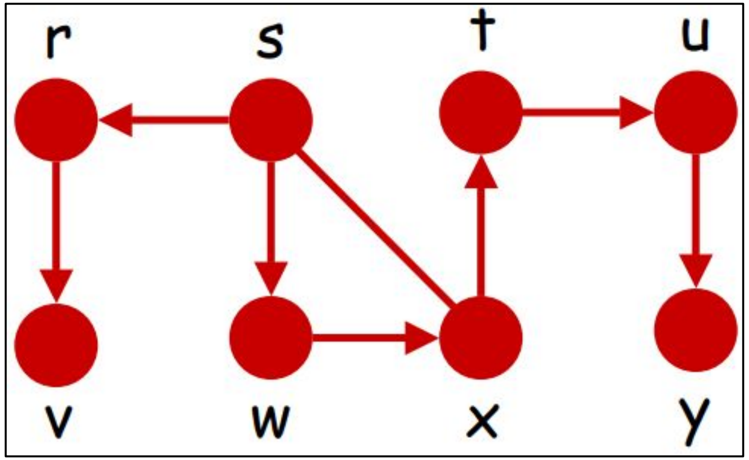
All function call terminate, DFS end

DFS - Example

● finished

◌ discovered

將某條岔路探索到底後，再探索其他岔路



visit order: s , w , x , t , u , y , r , v

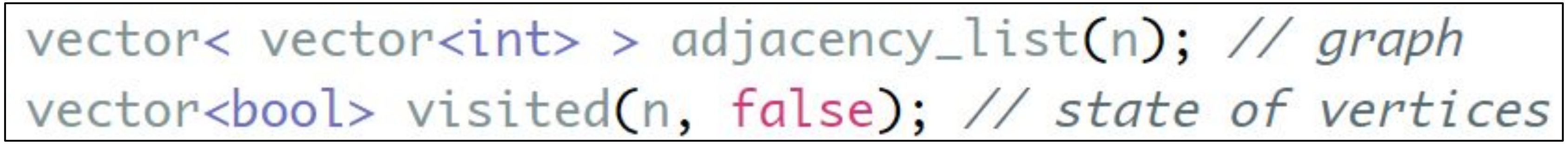
第一條岔路：s , w , x , t , u , y

第二條岔路：s , r , v

DFS - Time complexity

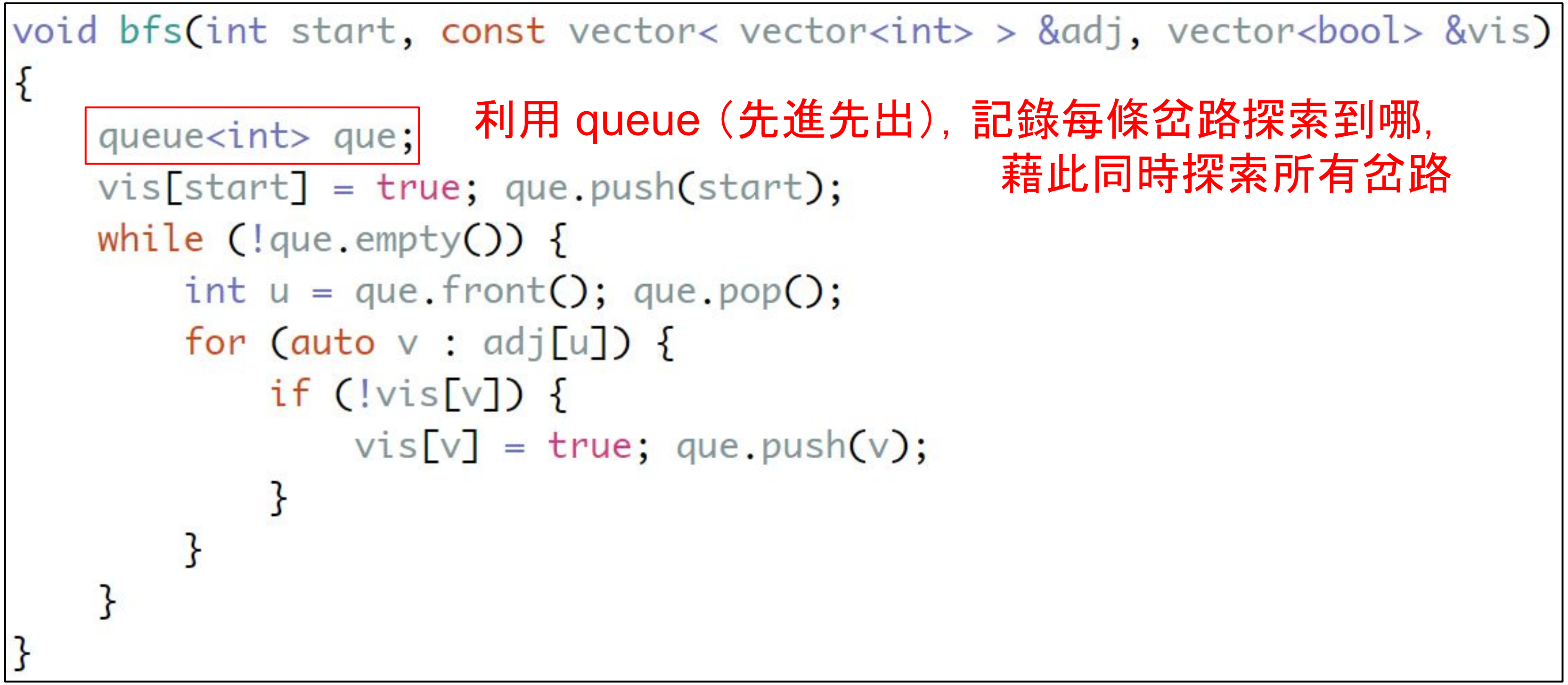
- Adjacency list
  - each vertex in the adjacency lists is examined at most once
  - O(|V| + |E|)
- Adjacency matrix
  - determine all neighbors of specific vertex takes O(|V|)
  - at most |V| vertices are visited
  - O(|V| ^ 2)

BFS - Implementation



```
vector< vector<int> > adjacency_list(n); // graph
vector<bool> visited(n, false); // state of vertices
```

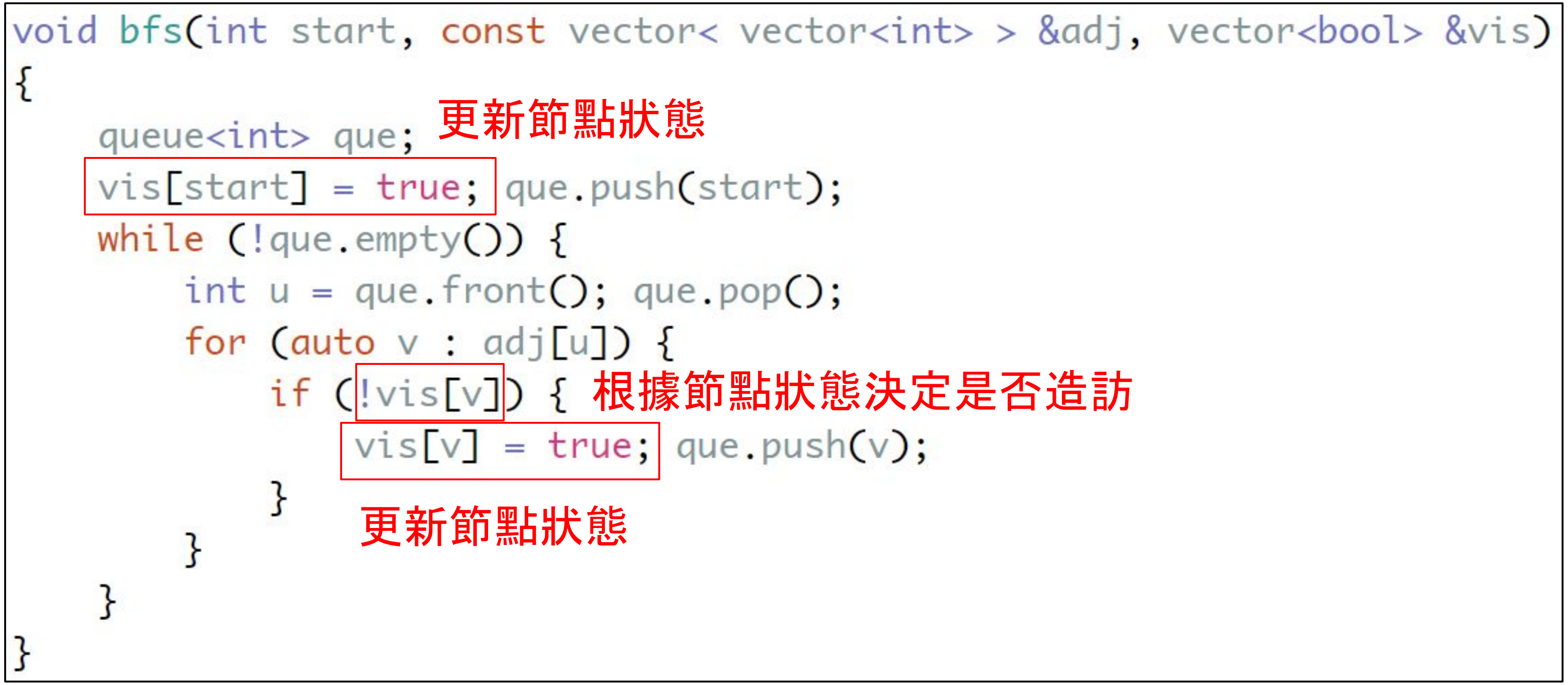
BFS - Implementation



```cpp
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
            }
        }
    }
}
```

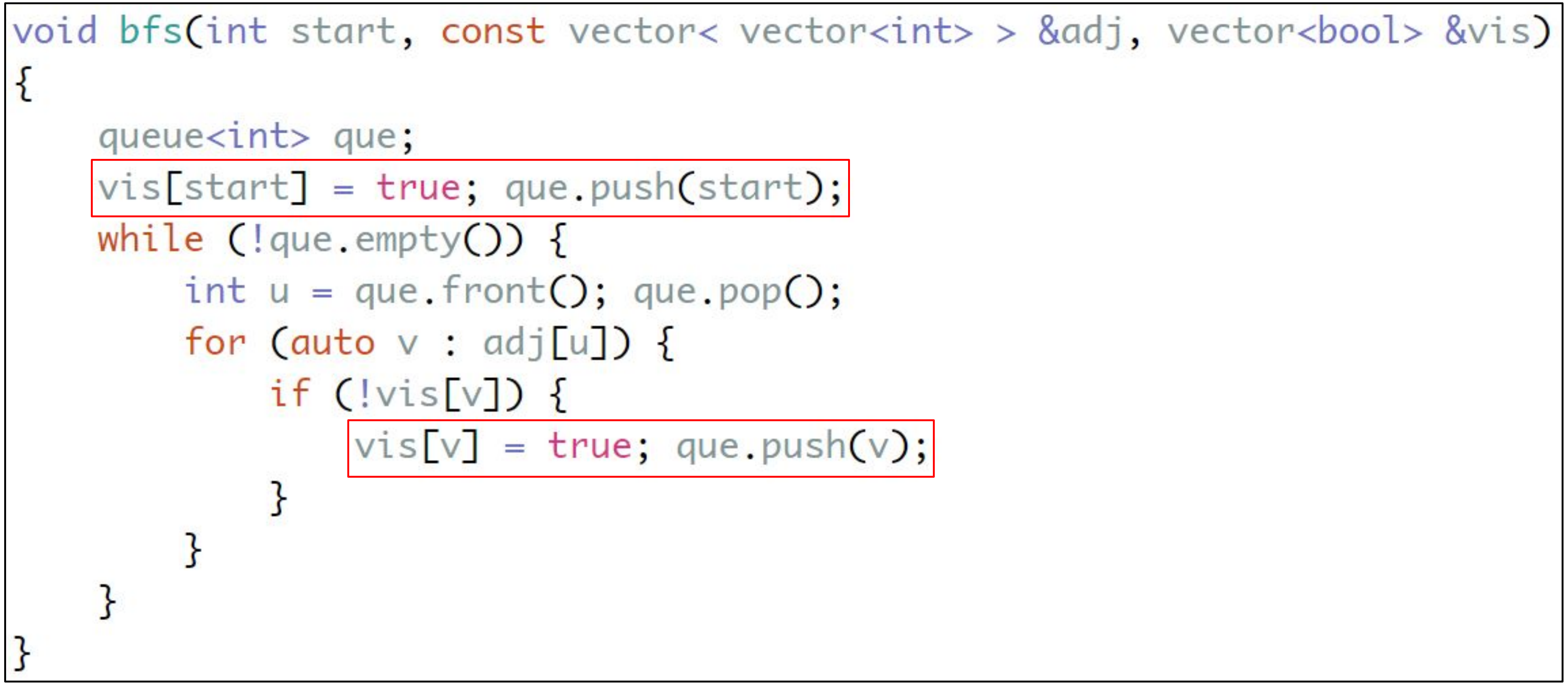
利用 queue（先進先出）, 記錄每條岔路探索到哪,
藉此同時探索所有岔路

BFS - Implementation



```
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
            }
        }
    }
}
```

BFS - Implementation



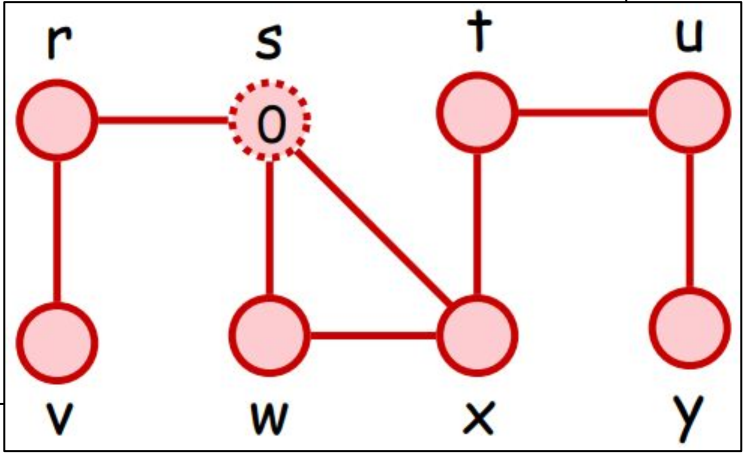
```cpp
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
            }
        }
    }
}
```

● finished

◌ discovered

BFS - Example

```
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
```



visit order:

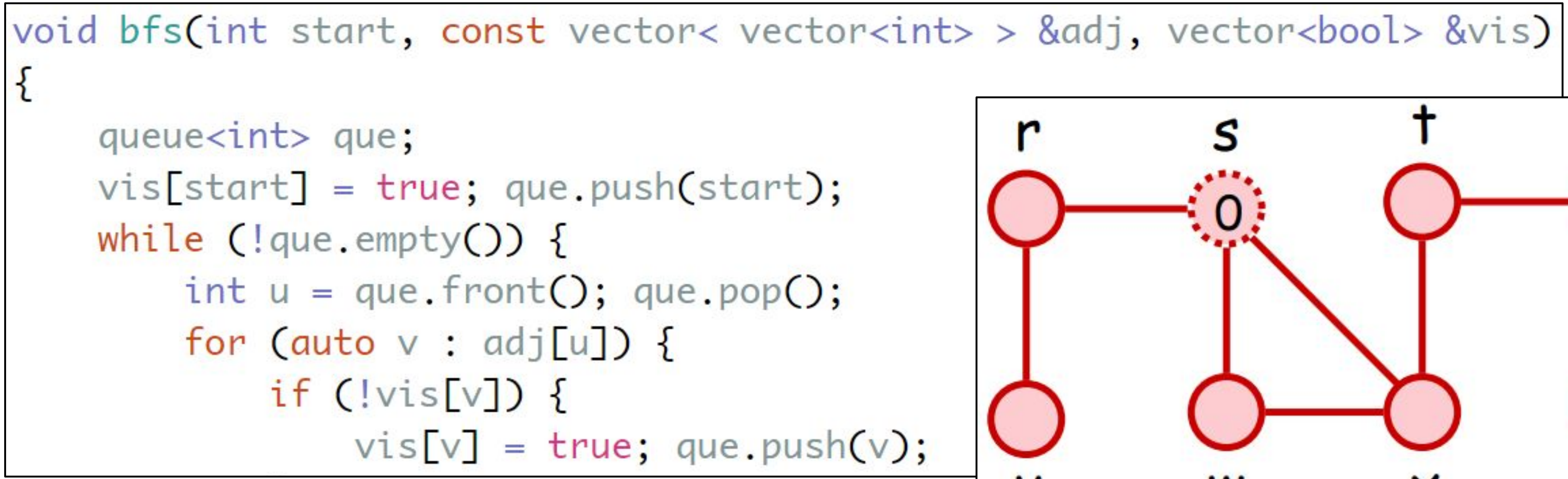
queue: (empty)

Initialize, push s into queue

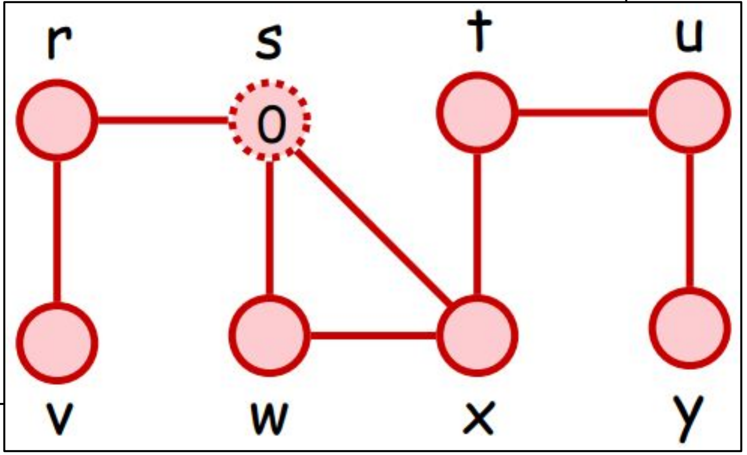
BFS - Example

● finished

◌ discovered



```
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
```



visit order: s

queue: s

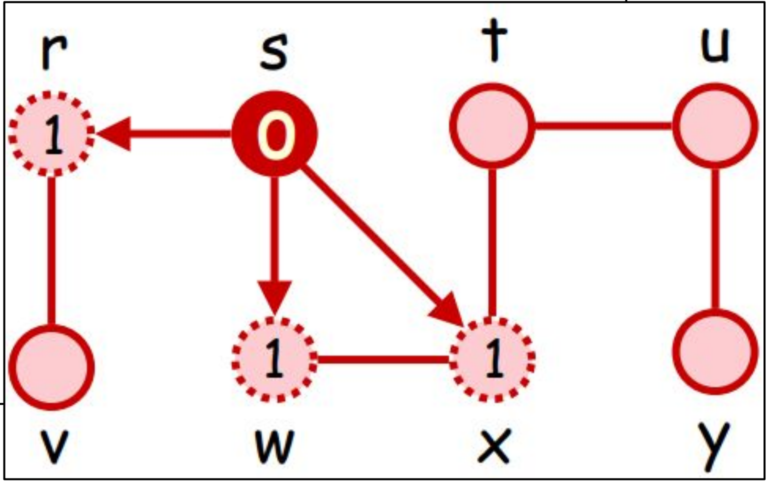
pop queue, in adj[s], find vis[w] = vis[r] = vis[x] = false, push them into queue

BFS - Example

● finished

◌ discovered

```
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
```



visit order: s, w, r, x

queue: w, r, x

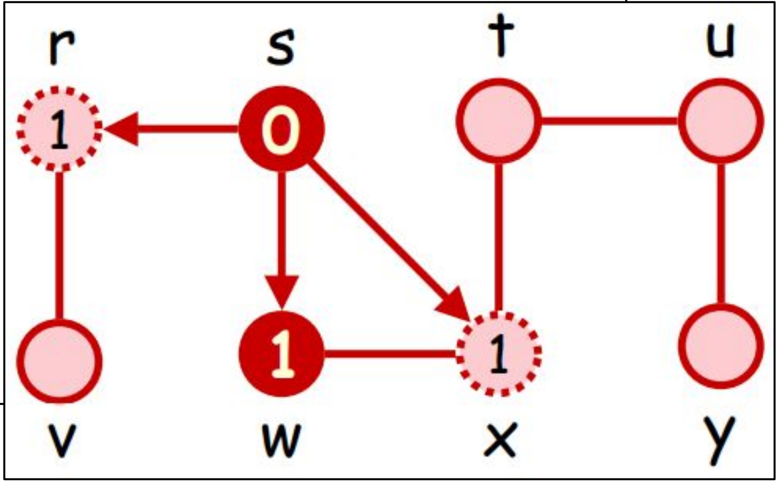
pop queue, in adj[w], find all neighbors are visited, do nothing

BFS - Example

● finished

◌ discovered

```
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
```



visit order: s, w, r, x

queue: r, x

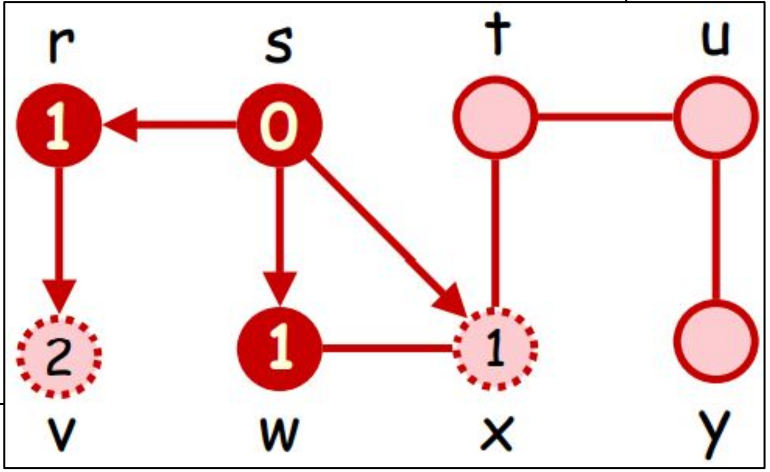
pop queue, in adj[r], find vis[v] = false, push it into queue

BFS - Example

● finished

◌ discovered

```
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
```

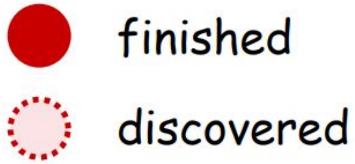


visit order: s, w, r, x, v

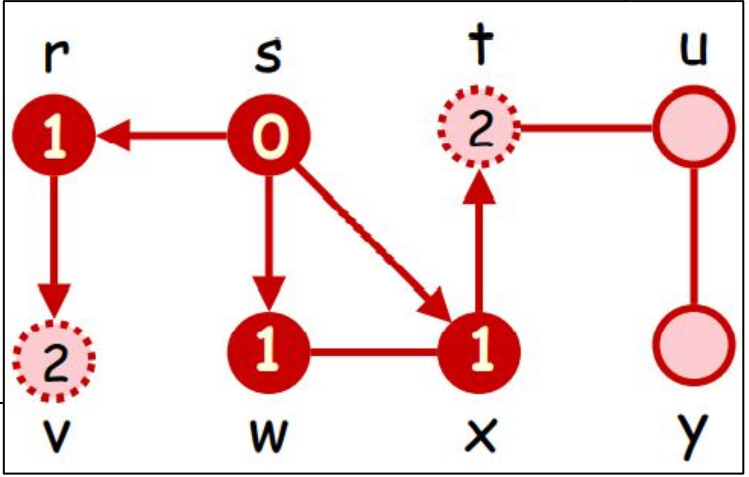
queue: x, v

pop queue, in adj[x], find vis[t] = false, push it into queue

BFS - Example



```
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
```



visit order: s, w, r, x, v, t

queue: v, t

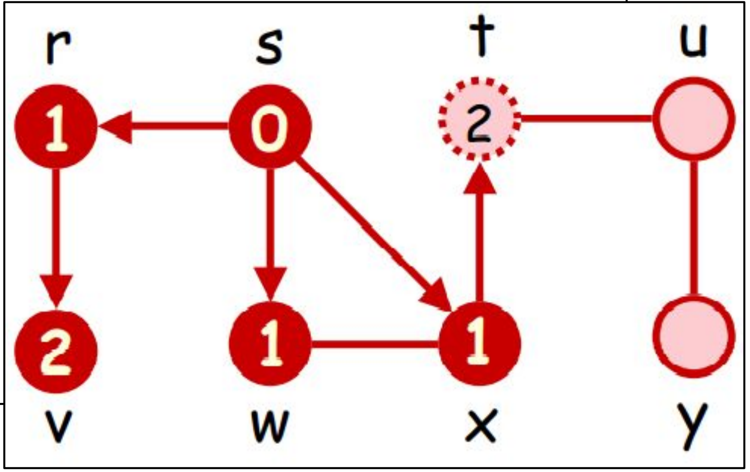
pop queue, in adj[v], find all neighbors are visited, do nothing

● finished

◌ discovered

BFS - Example

```
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
```



visit order: s, w, r, x, v, t

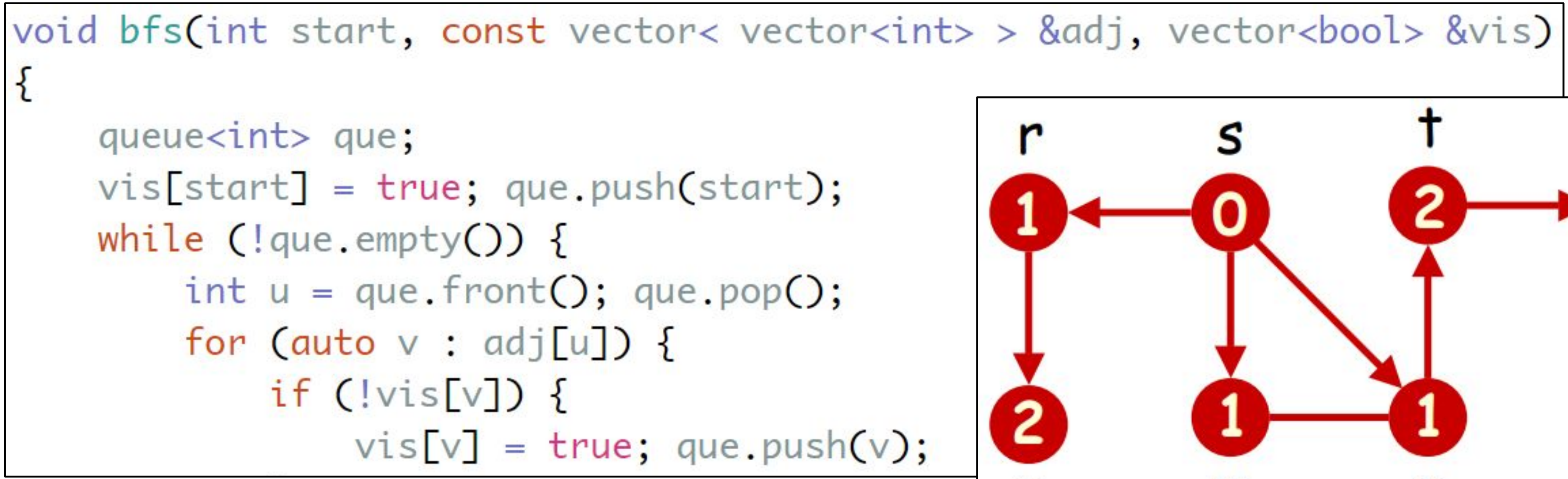
queue: t

pop queue, in adj[t], find vis[u] = false, push it into queue

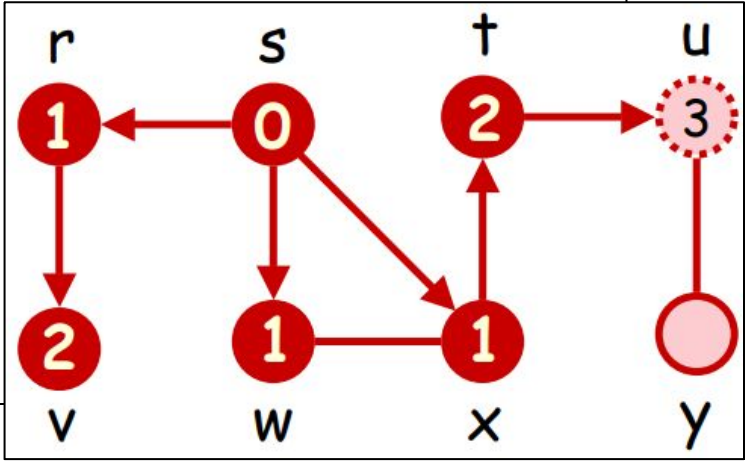
BFS - Example

● finished

◌ discovered



```
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
```



visit order: s, w, r, x, v, t, u

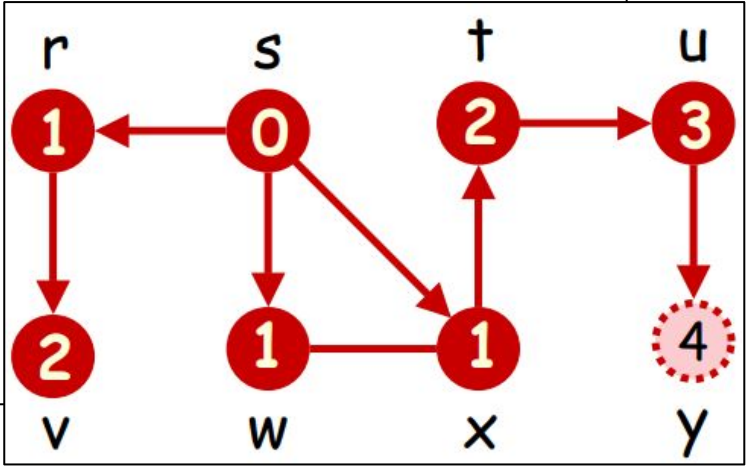
queue: u

pop queue, in adj[u], find vis[y] = false, push it into queue

BFS - Example

● finished
◌ discovered

```
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
```



visit order: s, w, r, x, v, t, u, y

queue: y

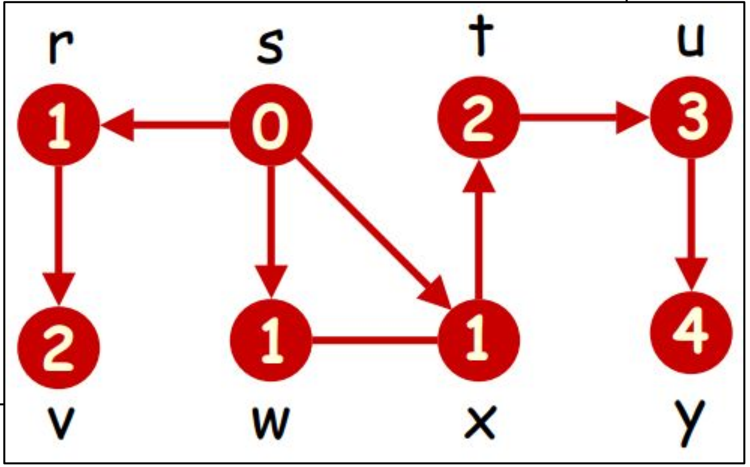
pop queue, in adj[y], find all neighbors are visited, do nothing

● finished

◌ discovered

BFS - Example

```
void bfs(int start, const vector< vector<int> > &adj, vector<bool> &vis)
{
    queue<int> que;
    vis[start] = true; que.push(start);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto v : adj[u]) {
            if (!vis[v]) {
                vis[v] = true; que.push(v);
```



visit order: s, w, r, x, v, t, u, y

queue: (empty)

The queue is empty, BFS end

● finished

◌ discovered

BFS - Example

同時探索所有岔路，並記錄每條岔路探索到哪

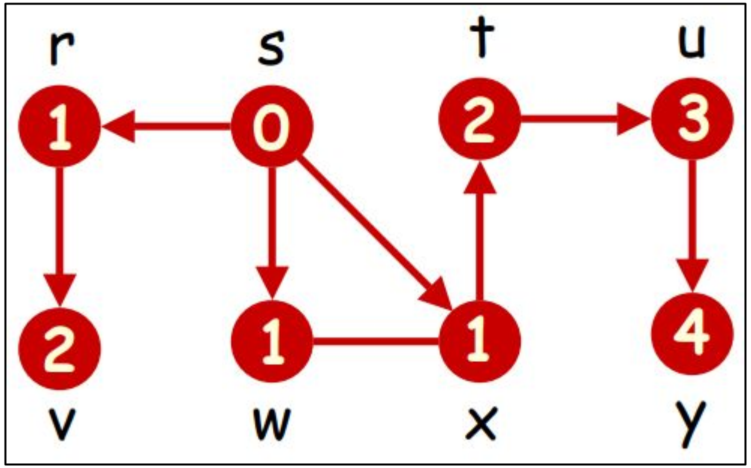
visit order: s , w , r , x , v , t , u , y

第一輪探索：w , r , x

第二輪探索：v , t

第三輪探索：u

第四輪探索：y

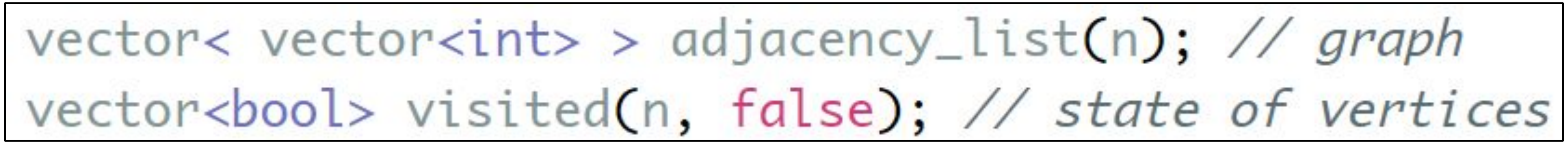


BFS - Time complexity

- Same as DFS

想想看

如果保證 graph 是一顆 tree，能否不記錄「所有」vertices 的被造訪狀態進行 DFS / BFS？



```
vector< vector<int> > adjacency_list(n); // graph
vector<bool> visited(n, false); // state of vertices
```

in class sample codes

DFS

https://ide.usaco.guide/O3RFwGTL41cRqB-PIbf

BFS

https://ide.usaco.guide/O3RH2mZBOXOv5gSGrwu

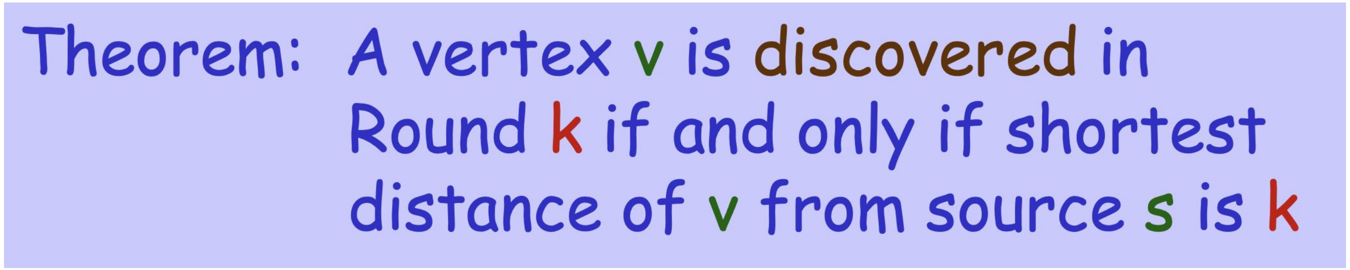
Lab 14. How far is the closest cookie

題目敘述

給定一張 n 點 m 邊的無向圖、起點 t 和一個點集 S，求從 t 到點集 S 的最小距離。

Hint

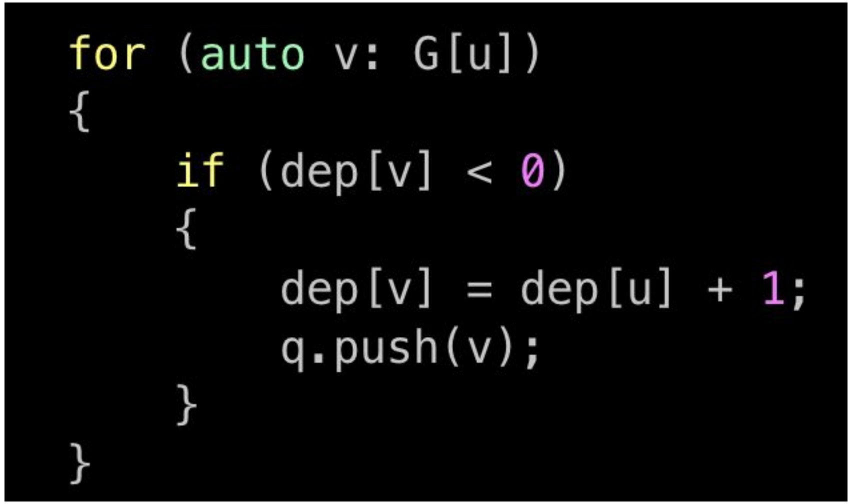
- 從起點 BFS
- 若在 Round k 時，遍歷到一個節點 u 屬於集合 S，
  則 k 是起點到該點集的最小距離



Theorem: A vertex v is discovered in Round k if and only if shortest distance of v from source s is k

實作

- 使用 queue 時如何紀錄 round?
- $round[v] = round[u] + 1$
- 若尚未造訪過 $round[v] = -1$
  - 可以代替 $vis[]$



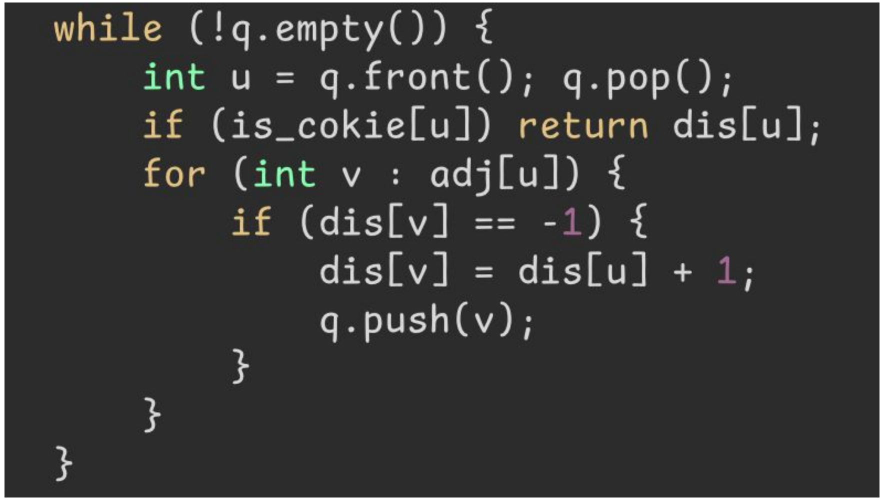
```
for (auto v: G[u])
{
    if (dep[v] < 0)
    {
        dep[v] = dep[u] + 1;
        q.push(v);
    }
}
```

實作

- 看從哪一輪開始可以找到餅乾

- code 單起點

- code 多起點



```
while (!q.empty()) {
    int u = q.front(); q.pop();
    if (is_cokie[u]) return dis[u];
    for (int v : adj[u]) {
        if (dis[v] == -1) {
            dis[v] = dis[u] + 1;
            q.push(v);
        }
    }
}
```

Lab 14. Graph Connected Component

題目敘述

給定一張 n 點 m 邊的無向圖、求連通快的數量。

Hint

- DFS 一定可以走完一個連通塊
- 遍歷點, 若點沒有被參訪過則 DFS, 且答案加一

實作

- Code

End