

4.1 Breadth-First Search (BFS)

Define: $G=(V,E)$: undirected or directed

V : vertex set E : edge set

$\forall u,v \in V$ " $s \in V$: source vertex (起點)

(2). $u.\pi$: predecessor (u的前一個點, 由誰發現u的)
(or parent of u)

$s.\pi = Nil$

(3). $u.color = \begin{cases} \text{white: undiscovered} \\ \text{gray: } u \text{ 為 discovered 且 } \exists u \text{ 之 neighbor 為 undiscovered} \\ \text{black: others} \end{cases}$

(4). $u.d$: BFS 計算由 s 到 u 之距離

(5). $\delta(s,v)$: 由 s 至 v 所需經之最少邊數

(6). $adj[u] = \{v | (u,v) \in E\}$

Algorithm:

BFS(G, s)

for each vertex $u \in G.V - \{s\}$ // initialization $\Rightarrow O(|V|)$

$u.color = \text{white}$
 $u.d = \infty$
 $u.\pi = Nil$

$s.color = \text{gray}$

$s.d = 0$

$s.\pi = Nil$

Let Q be a new queue // 宣告 Queue 有放必須拜訪的 vertex

Enqueue(Q, s)

while $Q \neq \emptyset \Rightarrow O(|V|)$

$u = \text{Dequeue}(Q)$

for each $v \in G.adj[u] \Rightarrow O(|V|)$

if $v.color == \text{white}$

$v.color = \text{gray}$

$v.d = u.d + 1$

$v.\pi = u$

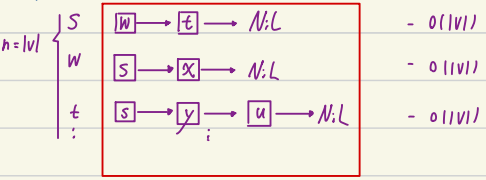
Enqueue(Q, v)

$u.color = \text{black}$

Time complexity: 從上面 Algo 知: $O(|V|^2)$ 但不夠準確 (使用 for loop 分行方式, 只有在 k_n 才有可能)

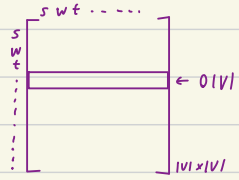
BFS 時間會和 儲存 graph 的 DS 相關

① adjacency list: (一般利用此種方式儲存) (用 linked list 方式儲每個點的 neighbor)



$O(2|E|) \Rightarrow$ 為 undirected graph
 \therefore 為: $O(|V| + |E|)$ 每個邊算兩次

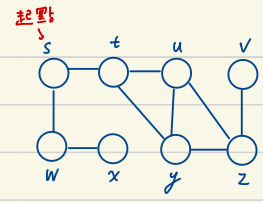
② adjacency matrix



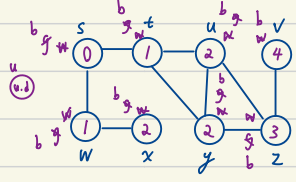
\therefore look up $adj[u]$ costs $O(|V|)$
 $\therefore O(|V|^2)$

Example: Initial:

FIFO Queue: 空



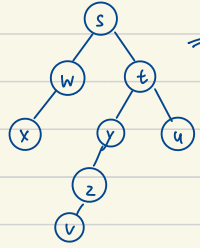
* Node 中數字為 $u.d$



① 記錄了 distance ($u.d$)
② 若 BFS 走完後有白色, 不為 connected 而白色的點就是 s 出發無法到達的點。

FIFO Queue: s, w, x, y, u, v

step: I. 從 s 開始, 先計算 $u.d$, 放到 queue 中, 再改 $u.color$
II. 找附近 color 不為白的, 放到 queue 中
III. 依 queue FIFO 取出, 重複 I ~ III.



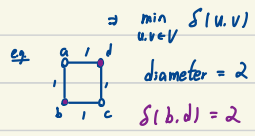
\Rightarrow breadth-first tree: 由 BFS 過程生長得到樹

Note: ① BFS 只會走訪 s 可到達的 vertex
② BFS 可用作判斷 G 是否 connected

Note: ① $\forall v \in V, v.d = \delta(s, v)$ (由 s 到 v 的最短距離, 或經過最少邊數)

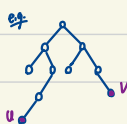
i.e. 當 $G=(V, E)$ 為 unweighted 時, 則 BFS 可用作求 single-source shortest path

② diameter: G 中距離最遠 = 點的 distance (shortest path 長度)



Algorithm: brute force: 若為 single source 求 shortest pair 等價於 BFS 把每個點當作 source 作一遍
 $\Rightarrow O(|V| + |E|) \cdot O(|V|) = O(|V|^2)$

③ Tree 中找 diameter: $O(|V|)$ (無 cycle)



① 先任意挑一黑點做 BFS
② 建 BFS tree 可得該 tree 的 leaf
③ 再找 leaf 中離 source 最遠的點, 設為 u
④ 再用 v 做一 BFS 則可得離最遠點 v 則 $\delta(u, v)$ 為所求 *

\therefore Tree 中 $|E| = |V| - 1$
 $\therefore O(|E|) = O(|V|)$
 $\Rightarrow O(|V|^2)$

資源: $\frac{u.d}{u.f}$ - 一旦還有 white, 就 continue

depth-first forest:

4.2 Depth-first Search

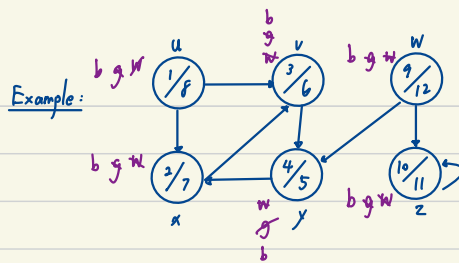
Define: Time Stamp

$G = (V, E)$ 為一有(無)向圖

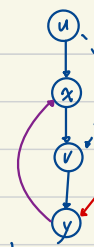
$\forall u \in V$, 定義: $u.d$ 為 discovery time

$u.f$ 為 finishing time

示意: $\xrightarrow{\text{white}} \xrightarrow{\text{gray}} \xrightarrow{\text{black}}$ time



和 BFS 不同, 採用深度優先 (一直走, 走到不能走)



在 depth first forest 中
有的邊為: tree edge

tree edge: 發現點為 white
forward edge: 發現點為 black
back edge: 發現點為 gray
cross edge: 發現點為 black (無法和 forward edge 區別)

Theorem: 若 G : undirected graph

$\Rightarrow G$ 中僅含 tree edge 和 back edge

\therefore forward edge 變成 back edge

cross edge 變成 tree edge

Note: $\forall (u, v) \in E$

①. $v.color = white \Rightarrow (u, v)$: tree edge

②. $v.color = gray \Rightarrow (u, v)$: back edge

③. $v.color = black$ 且 $u.d < v.d \Rightarrow (u, v)$: forward edge $\Rightarrow u.d < v.d < v.f < u.f$

④. $v.color = black$ 且 $v.d < u.d \Rightarrow (u, v)$: cross edge $\Rightarrow v.d < v.f < u.d < u.f$

Algorithm: DFS(G)

for each vertex $u \in G.V - \{s\}$ // initialization $\Rightarrow O(|V|)$

$u.color = white$

$u.d = \infty$

$u.\pi = NIL$

$t = 0$ // time stamp

for each $u \in G.V$

if $u.color == white$

DFS-visit(G, u)

DFS-visit(G, u)

$t = t + 1$

$u.d = t$

$u.color = gray$

for each $v \in G.adj[u]$

if $v.color == white$

$v.\pi = u$

DFS-visit(G, v)

$u.color = black$

$t = t + 1$

$u.f = t$

Note: DFS 的應用

①. 判斷 G 是否為 connected, 並找出所有 connected component

②. 判斷 G 是否為 acyclic

③. 在 directed acyclic graph 上找 - topological sort (DAG)

④. 在 directed graph (digraph) 上找出所有 strongly connected component

⑤. 在 undirected graph 找 biconnected component 和 articulation point

Note: ①. 给定 - digraph $G = (V, E)$. 若 L 為 V 之 sorting 滿足:

$\forall u, v \in V$, 若 $(u, v) \in E$

則 L 中 u 出現在 v 之前

則 L : topological sort (TS)

用在工作排程問題上: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$

$\Rightarrow u \rightarrow v$
 $u.f > v.f$

TS: 1 2 3 4 5 6 (不唯一)
2 3 1 4 5 6

等再做一次 DFS 後, 對所有點的 finishing time 排序
可得 - topological sort

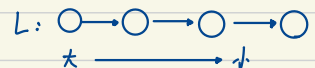
②. $G = (V, E)$: digraph $\Rightarrow G$ 之 TS 存在 $\Leftrightarrow G$ 為 acyclic (有 cycle 會 circular waiting)

Algorithm: Topological-Sort(G) // G 為 DAG

$L = \emptyset$: Linked List

DFS(G), 每當 $v.color = black$, 將其插入 L 前左端

return L



Time complexity:

\therefore 每個點最多走過 constant time $\Rightarrow O(|V|)$

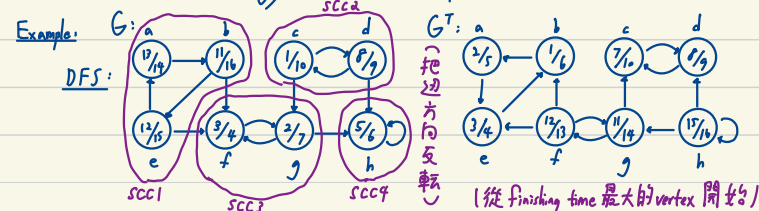
每個邊最多走過 constant time $\Rightarrow O(|E|)$

$\therefore O(|V| + |E|)$

Define: $G = (V, E)$: digraph. 若 $C \subseteq V$ 為 maximal 滿足 $\forall u, v \in C$

\exists - path 由 u 走到 v 且 \exists - path 由 v 走到 u

則稱 C 為 strongly connected component of G .



Depth-First Forest

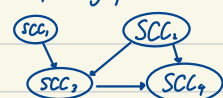
$\Rightarrow G^T$ 的 depth-first forest

中的每一棵 tree 為 G

的 strongly-connected component

AAAAA
Idea:

①. 建 component graph, 會為 DAG



②. 將該 component graph 的邊反向

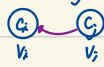
做 DFS 可得各個 SCC

In component graph:

G : $C_i \rightarrow C_j$ \Rightarrow 具有 $f(C_i) > f(C_j)$ 關係
其中: $f(u) = \max \{u.f \mid u \in V\}$

$\boxed{b} \downarrow \boxed{a} \downarrow \boxed{e}$
 $\boxed{c} \downarrow \boxed{d}$
 $\boxed{g} \downarrow \boxed{f}$
 $\boxed{h} \Rightarrow \text{components}$

$\langle PF \rangle: \because \forall v_j \in C_j, v_i \in C_i, v_j$ 皆為 v_i 後代
 又後代的 finishing time 必較小

\Rightarrow 把邊取反向, 又找 finishing time 最大點開始作 DFS
 必無法走出 \Rightarrow  $\because f(v_i) > f(v_j)$
 \therefore 必取到 C_i 中的點

Algo: SCC(G) // G: digraph

1. 執行 DFS(G) 以求 $v.f, \forall v \in V$
2. 求 $G^T = (V, E^T)$, 其中: $E^T = \{(u, v) \mid (v, u) \in E\}$
3. 執行 DFS(G^T). 過程中, 以 step 2 中之 $v.f$ 大到小選點
4. return DFS(G^T) 中每個 tree

4.3 Single-source shortest path

Define: $G = (V, E)$: digraph 且 weighted (non-weighted 可用 BFS 求)

$w: E \rightarrow \mathbb{R}$: weight function

$w(P)$: path P 上所有边 weight 和

①. $\forall u, v \in V, \delta(u, v) = \min \{w(P) \mid P \text{ 为 } u \text{ 到 } v \text{ 的 path}\}$

②. single-source: 求单一起点 s 到各点 v 之 $\delta(s, v)$

③. all-pairs: 求 G 中任两黑点 u, v 之 $\delta(u, v)$

Note: $G = (V, E)$: digraph 且 weighted

s : source vertex

①. T : rooted tree with root s

若 T 中的每一條由 s 到 v 之 path

皆為 G 上 s 至 v 的 shortest path

則 T : shortest path tree

(single source 上必存在 shortest path tree)