

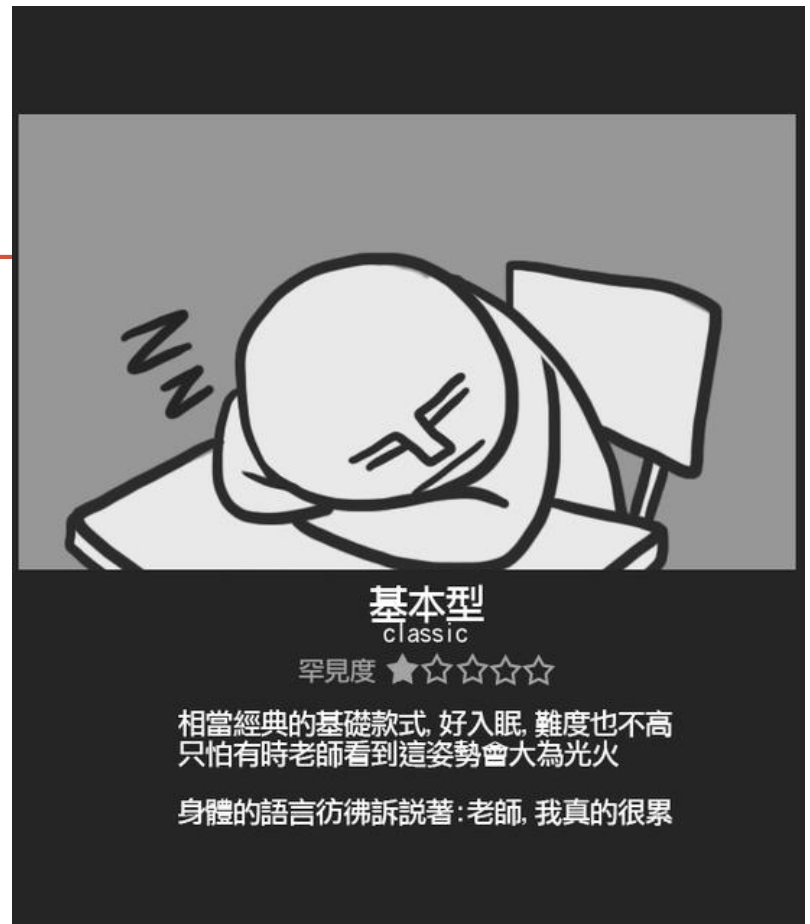
連載: 學生上課睡覺姿勢大全

<http://www.wretch.cc/blog/chi771027/26489957>

# GRAPH 1

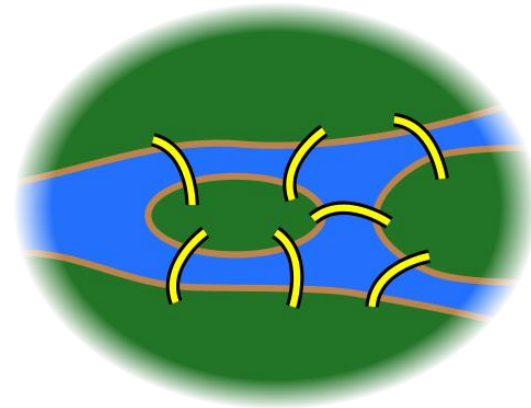
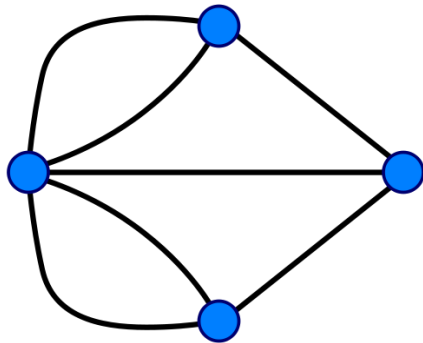
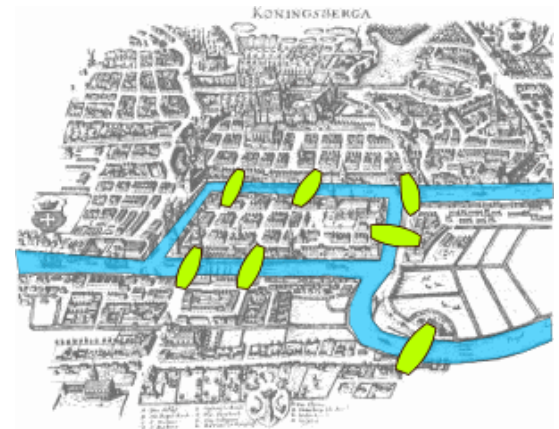
Michael Tsai

2013/4/2



# Königsberg Seven Bridge Problem

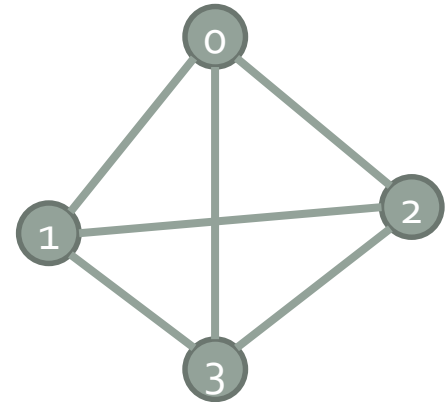
- 西元1736年, Euler嘗試著要解答這個問題:
- 右邊地圖中, 有沒有可能找出一條路徑, 使得每一條橋都走過一次之後又回到出發點?



答案: Graph必須要是connected & 必須有0個或2個node有odd degree

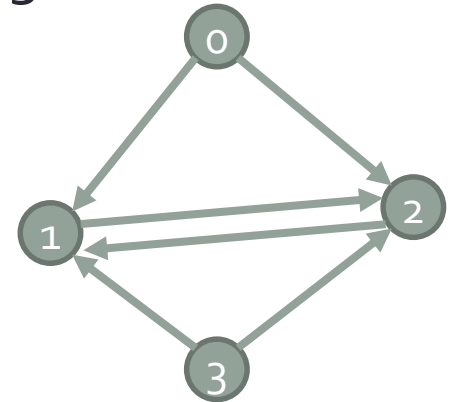
# A graph

- $G$ : a graph, consists of two sets,  $V$  and  $E$ .
- $V$ : a finite, nonempty set of vertices.
  - 單數: vertex; 複數: vertices
- $E$ : a set of pairs of vertices.
- $G=(V,E)$
- $V=\{0,1,2,3\}$
- $E=\{(0,1),(0,2),(0,3),(1,2),(1,3),(2,3)\}$



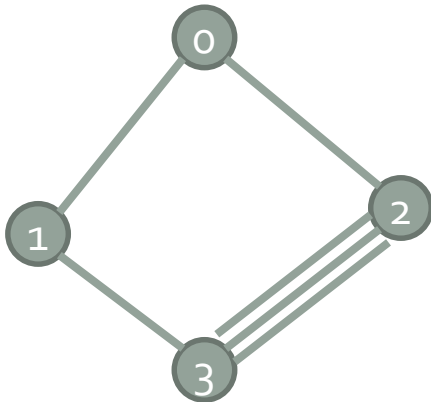
# Directed & undirected graph

- Graph中, edge有方向的叫做directed graph, 沒方向的叫做undirected graph
- Directed graph 通常又叫digraph – edge是**ordered** pairs
- Undirected graph 通常就只叫做graph – edge是**unordered** pairs
- $(1,2)$  和  $(2,1)$  在undirected graph是一樣的edge
- $\langle 1,2 \rangle$  和  $\langle 2,1 \rangle$  在digraph是不一樣的edge
- $V = \{0,1,2,3\}$
- $E = ?$
- $E = \{\langle 0,1 \rangle, \langle 0,2 \rangle, \langle 1,2 \rangle, \langle 2,1 \rangle, \langle 3,1 \rangle, \langle 3,2 \rangle\}$

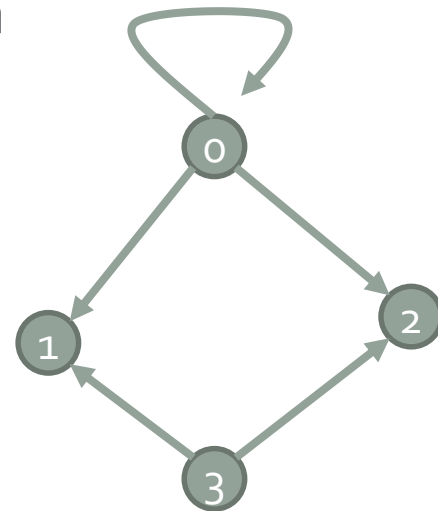


# Self Edge & Multigraph

Multigraph



Graph with  
self-edges  
 $\langle v, v \rangle$

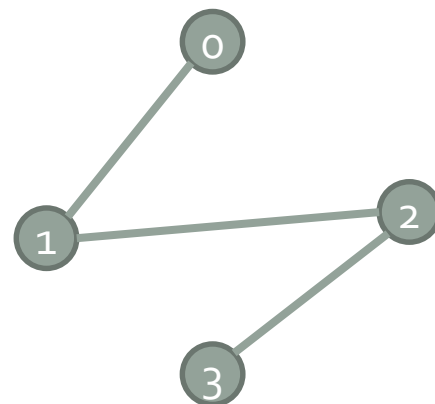


# Maximum number of edges

- 一個有n vertices的graph, 最多有幾個edges?
- 一個有n vertices的digraph, 最多有幾個edges?
- 答案: graph:  $\frac{n(n-1)}{2}$ ,
- digraph:  $n(n-1)$

# 怎麼這麼多名詞XD

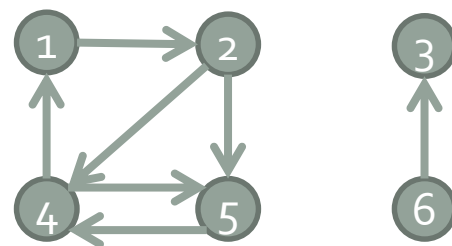
- 相鄰(adjacent):  
如果有edge  $(u,v)$ , 那麼 $u, v$ 兩vertices就是adjacent.
- 如果有edge  $\langle u,v \rangle$  那麼我們說  
 $v$  is adjacent to  $u$ .  
(有些課本用相反的定義T\_T)
- 作用(incident): 如果有edge  $(u,v)$  那麼 $u, v$ 兩vertices就是incident (作用) on (在)  $u$ 和 $v$ 上面
- $\langle u,v \rangle$  is incident **from**  $u$  and is incident **to**  $v$ .
- $\langle u,v \rangle$  **leaves**  $u$  and **enters**  $v$ .
- Subgraph: 如果 $G=(V,E)$ ,  $G'=(V',E')$ 是 $G$ 的subgraph, 則 $V' \subseteq V$ 且 $E' \subseteq E$



# Degree of vertex

- Vertex的**degree**:
- 有幾個edge連在vertex上
- Digraph中
- 又可分為in-degree and out-degree
- in-degree: 進入vertex的edge數
- out-degree: 出去vertex的edge數
- $\text{degree} = \text{in-degree} + \text{out-degree}$
- 一個degree=0的vertex可稱為isolated

- Edge數和degree的關係: 
$$e = \frac{(\sum_{i=0}^{n-1} d_i)}{2}$$



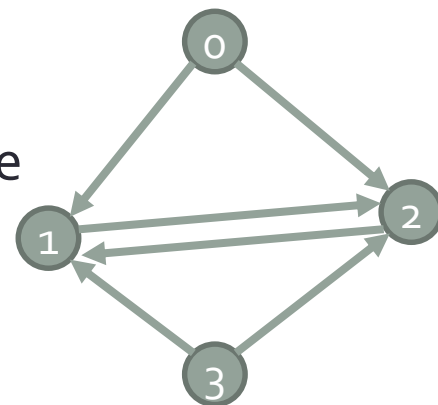


# 路徑 (path)

- Path: 一條從 $u$ 到 $v$ 的path, 是一連串的vertices,  $u, i_1, i_2, \dots, i_k, v$ , 而且 $(u, i_1), (i_1, i_2), \dots, (i_{k-1}, i_k), (i_k, v)$ 都是edge.

(digraph的定義自行類推)

- 如果有一條path  $p$  從 $u$ 到 $u'$ , 則我們說 $u'$  is reachable from  $u$  via  $p$ .

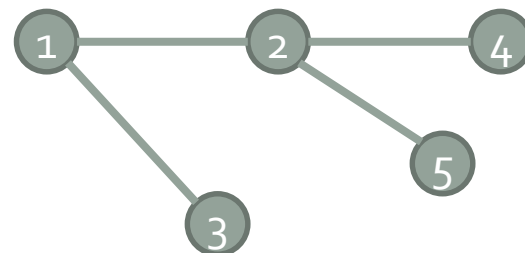
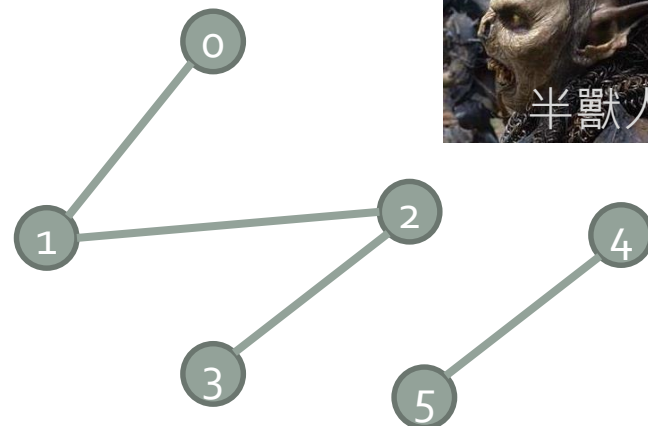


- Path的length: 裡面有幾條edge
- Simple path: 除了 $u, v$ (起點和終點)以外, 其他的vertices都沒有重複過.
- Cycle: 一個 $u$ 和 $v$ 一樣的simple path
- Subpath:??
- 答案: 定義path的vertex sequence裡面的連續的一段.



# 有連接的 (connected)

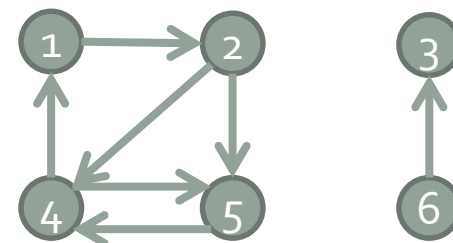
- **In undirected graph:**
  - Vertices  $u$  and  $v$  are said to be **connected** iff
  - there is a path from  $u$  to  $v$ . (graph & digraph)
- **Connected graph:**
  - iff every pair of distinct vertices  $u$  and  $v$  in  $V(G)$  is connected
- **Connected component** (也有人直接叫component):
  - A maximal connected subgraph
- **Maximal:** no other subgraph in  $G$  is both connected and contains the component.
- 問: Tree是一個怎麼樣的graph?
- 答: connected and acyclic (沒有cycle) graph
- 問: Forest是一個怎麼樣的graph?
- 答: acyclic graph



# 強連接 (strongly connected)



- In digraph:



- **Strongly connected:**

- $G$  is strongly connected iff
- for every pair of distinct vertices  $u$  and  $v$  in  $V$
- there is a directed path from  $u$  to  $v$  and also a directed path from  $v$  to  $u$ .

- **Strongly connected component:**

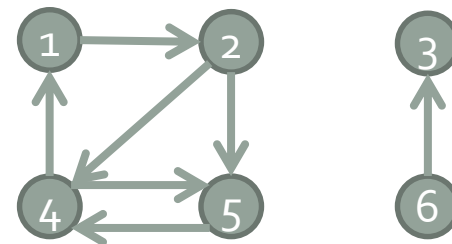
- A maximal subgraph which is strongly connected.

# 要怎麼表示一個graph呢?

- 主流表示法:
- Adjacency matrix (用array)
- Adjacency lists (用linked list)
- 兩者都可以表示directed & undirected graph

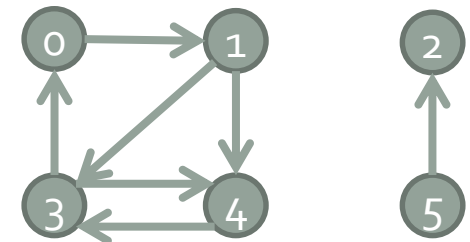
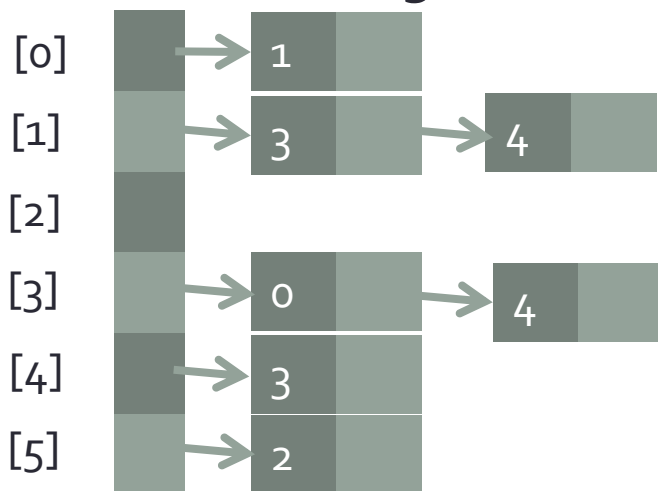
# 表示法I 用Array

- 本方法叫做adjacency matrix
- index當作vertex號碼
- edge用matrix的值來表示:
- 如果有 $(i,j)$ 這條edge, 則 $a[i][j]=1$ ,  $a[j][i]=1$ . (graph)
- 如果有 $\langle i,j \rangle$ 這條edge, 則 $a[i][j]=1$ . (digraph)
- 對undirected graph,  $a = a^T$  (也就是對對角線對稱)
- 請同學解釋要怎麼建adjacency matrix☺
- 粉簡單. 那麼來看看好不好用:
- 如果要看總共有多少條edge?  $O(??)$
- 答:  $O(|V|^2)$
- 有沒有可能跟edge數 $(e)$ 成正比呢?
- 通常 $|E| \ll |V|^2$  所以adjacency matrix裡面很空



# 表示法II 用linked list

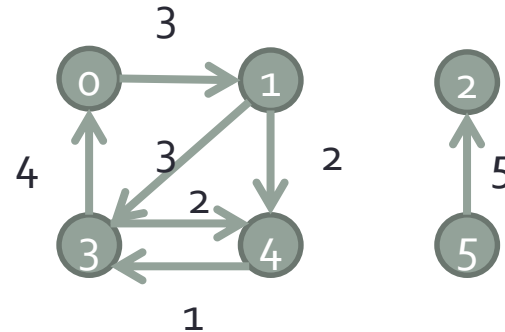
- 本方法叫做adjacency lists
- 建立一個list array  $a[n]$  ( $n$ 為vertex數目)
- 每個list裡面紀錄通往別的vertex的edge
- 問: 怎麼算in-degree?



- 答: 如果要比較容易的話, 要另外建“inverse adjacency lists”
- 請同學告訴我怎麼畫☺

# Weighted Edge

- Edge可以有“weight”
- 表示長度, 或者是需要花費
- 一個edge有weight的graph
- 又叫做network



- 想想看: 用剛剛的representation要怎麼儲存weight?
- 答:
- adjacency matrix可以用array的值來存
- adjacency list可以在list node裡面多開一個欄位存

# 兩者比較

- Adjacency lists:
  - 用來表示sparse graphs時使用比較少空間
  - 使用空間:  $O(|V| + |E|)$
- Adjacency matrix:
  - 要找某個edge  $(u,v)$ 有沒有在graph裡面比較快
  - 一個entry只需要1 bit (unweighted graph)
  - 簡單容易, graph小的時候用adjacency matrix比較方便
  - 使用空間:  $O(|V|^2)$



# Breadth-First Search (BFS)

- 給一個graph  $G=(V,E)$ 及一個source vertex  $s$
- 找出所有從 $s$  reachable的vertices
- 計算從 $s$ 到每一個reachable的vertex的最少edge數目
- 產生breadth-first tree,  $s$ 為root, 而其他reachable的vertices都在樹裡面
- Directed graph & undirected graph皆可
- 此方法會先找到所有距離 $s$  distance為 $k$ 的vertex, 然後再繼續找距離為 $k+1$ 的vertex

# BFS Pseudo-code

```

BFS (G, s)
for each vertex  $u \in G.V - \{s\}$ 
    u.color=WHITE
    u.d= $\infty$ 
    u.pi=NIL
s.color=GRAY
s.d=0
s.pi=NIL
Q={ }
ENQUEUE (Q, s)
while Q!={ }
    u=DEQUEUE (Q)
    for each  $v \in G.Adj[u]$ 
        if v.color==WHITE
            v.color=GRAY
            v.d=u.d+1
            v.pi=u
            ENQUEUE (Q, v)
    u.color=BLACK
  
```

初始所有vertex的值

初始開始search的vertex s的值

對每一個和u相連vertex

v.color: 用顏色來區別discover的狀況

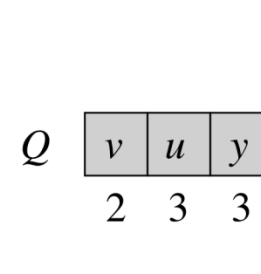
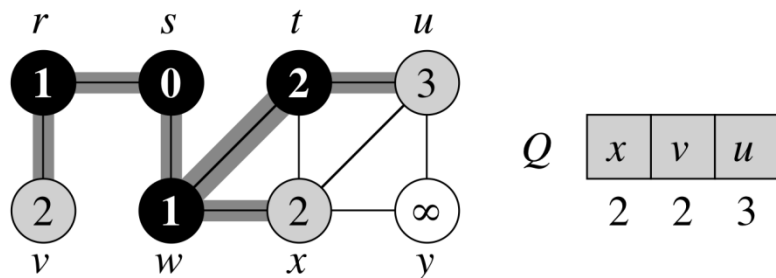
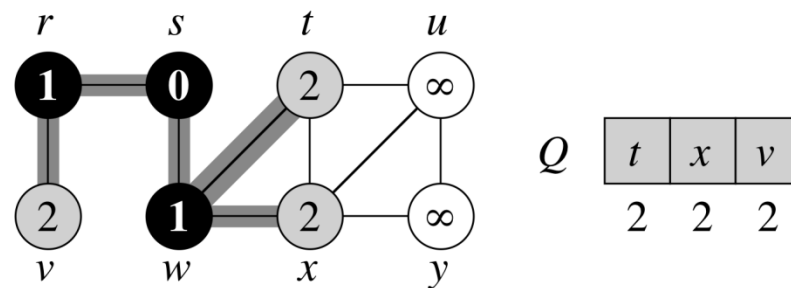
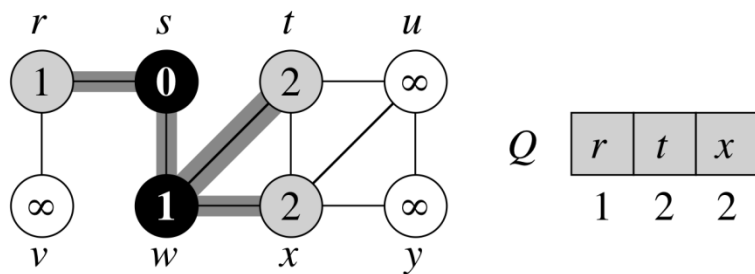
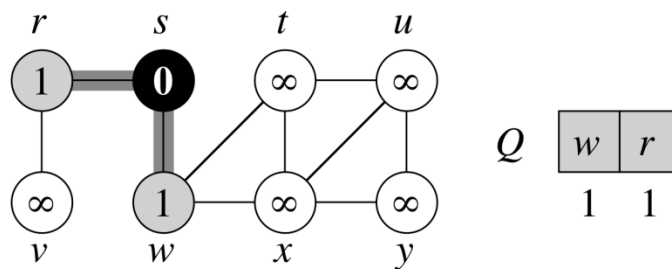
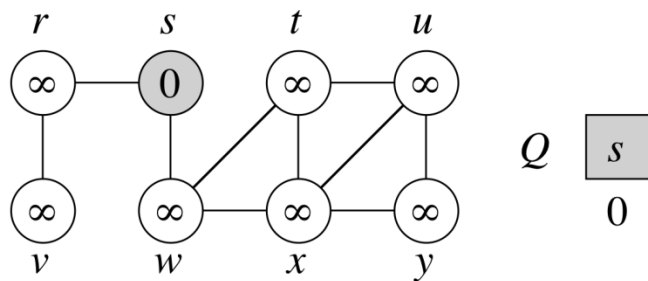
WHITE: 還沒discovered

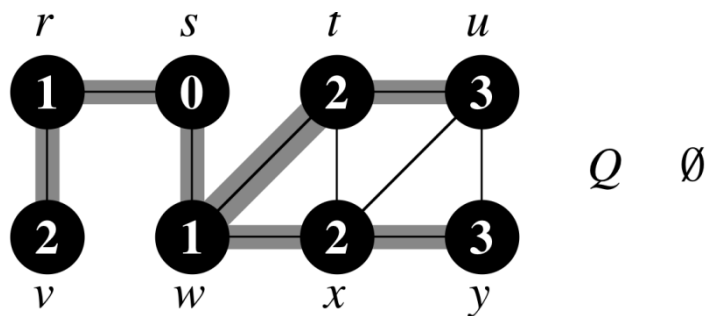
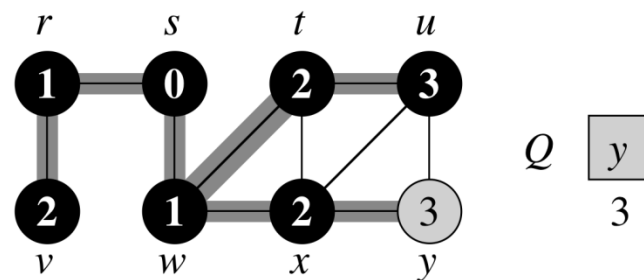
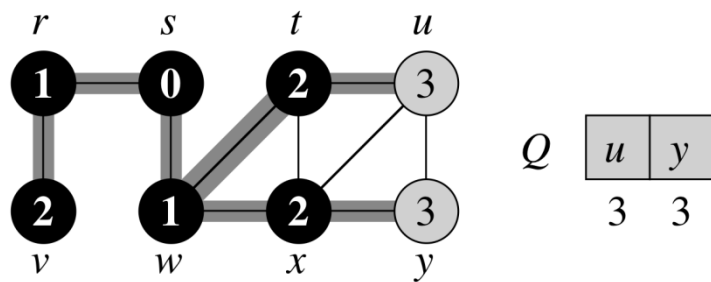
GRAY: discovered了, 但是和該vertex相連的鄰居還沒有都discovered

BLACK: discovered了且和該vertex相連的鄰居都已discovered

v.d: 和root的距離

v.pi: 祖先 (predecessor)





# BFS Pseudo-code

```

BFS (G, s)
  for each vertex  $u \in G.V - \{s\}$ 
    u.color=WHITE
    u.d= $\infty$ 
    u.pi=NIL
  s.color=GRAY
  s.d=0
  s.pi=NIL
  Q={ }
  ENQUEUE (Q, s)
  while Q!={ }
    u=DEQUEUE (Q)
    for each  $v \in G.Adj[u]$ 
      if v.color==WHITE
        v.color=GRAY
        v.d=u.d+1
        v.pi=u
        ENQUEUE (Q, v)
    u.color=BLACK

```

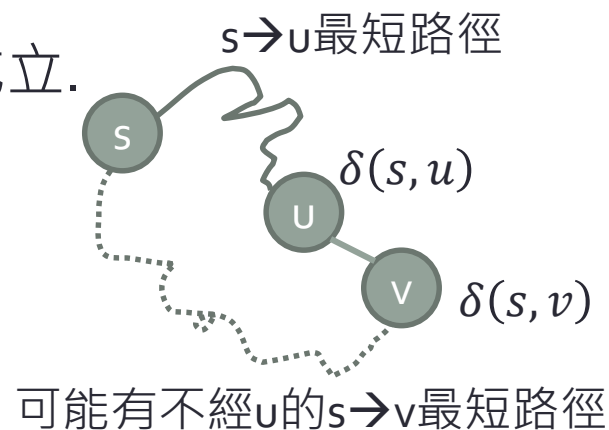
$O(V)$

$O(V + E)$

$O(E)$

# 證明BFS可以找到最短路徑

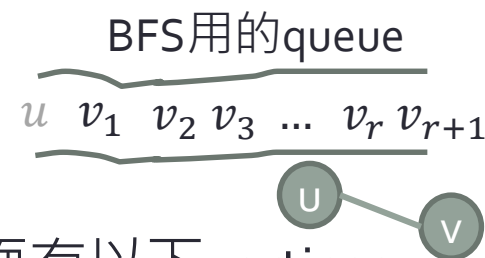
- 定義:  $\delta(s, v)$ :  $s$ 到 $v$ 的最短路徑的長度(邊的數目)
- Lemma 22.1:  $G=(V,E)$ 是一個directed或undirected graph.  $s$ 是一個任意vertex. 則對任何edge  $(u, v) \in E$ ,  $\delta(s, v) \leq \delta(s, u) + 1$ .
- 證明:
- 如果從 $s$ 開始,  $u$ 是reachable, 那麼 $v$ 也是.
- 這個狀況下,  $s \rightarrow v$ 的最短路徑只可能比 $\delta(s, u) + 1$ 短 (最短路徑可能不經由 $u$ 過來). 不等式成立.
- 如果 $u$ 不是reachable的話, 那麼不等式一定成立.



# 證明BFS可以找到最短路徑

- Lemma 22.2:  $G=(V,E)$ 是一個directed或undirected graph. 對 $G$ 及vertex  $s$ 跑BFS. 則結束的時候, 每一個 $v \in V$ , BFS計算的 $v.d \geq \delta(s, v)$ .
- 證明:
- 使用歸納法證明. 假設為“每一個 $v \in V$ , BFS計算的 $v.d \geq \delta(s, v)$ ”.
- 一開始把 $s$ 丟進queue的時候, 成立.  $s.d = \delta(s, s) = 0$ . 而其他的vertex  $v.d = \infty > \delta(s, v)$ . (起始條件)
- 當找到由edge  $(u,v)$ 找到vertex  $v$ 時, 我們可以假設 $u.d \geq \delta(s, u)$ . (k的時候成立)
- 且我們知道 $v.d = u.d + 1 \geq \delta(s, u) + 1 \geq \delta(s, v)$  Lemma 22.1 (證出k+1的時候成立)
- 每個 $v$ 都只做以上步驟(更改 $v.d$ 值)一次, 歸納法證明完成.

# 證明BFS可以找到最短路徑



- Lemma 22.3: 假設BFS執行在 $G=(V,E)$ 上, queue裡面有以下vertices  $\langle v_1, v_2, \dots, v_r \rangle$ ,  $v_1$ 是queue的頭, 而 $v_r$ 是queue的尾. 則 $v_r.d \leq v_1.d + 1$  and  $v_i.d \leq v_{i+1}.d$  for  $i = 1, 2, \dots, r - 1$ .
- 證明:
- 使用歸納法.
- 當queue一開始裡面只有s的時候成立. (起始條件)
- 假設某個時候queue裡面的東西是符合條件的. (k的時候)
- 現在我們必須證明每次dequeue或enqueue的時候, 都還是成立.
- (k+1的時候)
- (1) dequeue的時候,  $v_2$ 變成新的queue頭. 但 $v_1.d \leq v_2.d$ . 且 $v_r.d \leq v_1.d + 1 \leq v_2.d + 1$ . 其他不等式都不變. 因此成立.
- (2) enqueue的時候, 新加入的vertex  $v$ 變成 $v_{r+1}$ .
- 此時我們剛剛把 $u$ 拿掉(當時是queue的頭). 所以應該 $u.d \leq v_1.d$ . 所以 $v_{r+1}.d = v.d = u.d + 1 \leq v_1.d + 1$ .
- 且 $v_r.d \leq u.d + 1$ , so  $v_r.d \leq u.d + 1 = v.d = v_{r+1}.d$ .
- 其他的不等式都不變, 因此成立.



# 證明BFS可以找到最短路徑

- Corollary 22.4: vertices  $v_i$  和  $v_j$  在執行BFS時被enqueue且 $v_i$ 在 $v_j$ 之前被enqueue. 則當 $v_j$ 被enqueue的時候 $v_i.d \leq v_j.d$ .
- 證明: 直接從Lemma 22.3就可以得到. 因為對任一vertex  $v$ 來說,  $v.d$ 只會被指定值一次(enqueue之前).

# 證明BFS可以找到最短路徑

- Theorem 22.5: 證明BFS正確性. BFS執行在 $G=(V,E)$ 上, 從 $s \in V$ 開始. BFS執行的時候會找出所有從 $s$  reachable的vertex  $v \in V$ . 結束的時候, 每個 $v.d = \delta(s, v), \forall v \in V$ .

• 證明:

- (反證法)假設有一些 $v.d$ 不是 $\delta(s, v)$ . 讓 $v$ 是其中 $\delta(s, v)$ 最小的一個.

- Lemma 22.2說  $v.d \geq \delta(s, v)$ , 所以現在 $v.d > \delta(s, v)$ .

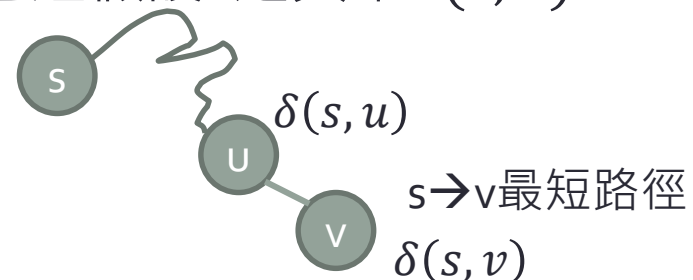
- 此時 $v$ 一定是從 $s$  reachable, 不然 $\delta(s, v) = \infty \geq v.d$ .

- 假設 $u$ 是 $s \rightarrow v$ 最短路徑上 $v$ 的前一個vertex, 則 $\delta(s, v) = \delta(s, u) + 1$

- (隱含意思:  $s \rightarrow v$ 上最短路徑上 $s \rightarrow u$ 的部分也是 $s \rightarrow u$ 的最短路徑)

- 因為 $\delta(s, u) < \delta(s, v)$ , 所以 $u.d = \delta(s, u)$  (已經假設 $v$ 是其中 $\delta(s, v)$ 最小的一個).

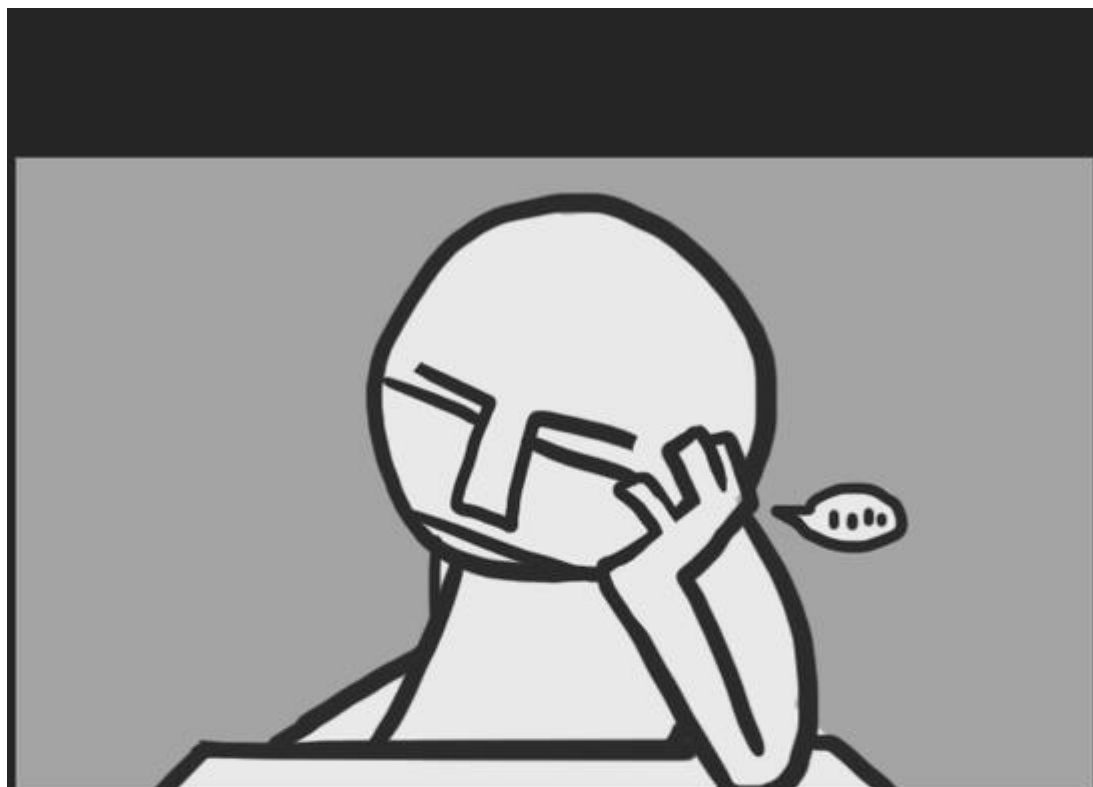
- $v.d > \delta(s, v) = \delta(s, u) + 1 = u.d + 1$



# 證明BFS可以找到最短路徑

- 上頁得到  $v.d > \delta(s, v) = \delta(s, u) + 1 = u.d + 1$
- 考慮BFS從Queue裡面把u dequeue出來的時候.
- u的鄰居v們, 可能是WHITE, GRAY, 或BLACK
- 如果是WHITE, 則會設  $v.d = u.d + 1$ , 矛盾.
- 如果是BLACK, 則它之前已經被dequeue過. Corollary 22.4說  $v.d \leq u.d$ , 矛盾.
- 如果是GRAY, 則它是剛剛dequeue某個vertex w的時候被改成GRAY的(比u dequeue的時間早). 所以  $v.d = w.d + 1$ . 但 Corollary 22.4說  $w.d \leq u.d$ , 因此  $v.d = w.d + 1 \leq u.d + 1$ , 矛盾.
- 因此原假設不成立.證明完畢.

# 下課休息



無聊型  
bor ing

新奇度 ★☆☆☆☆

通常此姿勢在聲音單調乏味的老師課堂上相當常見,學生聽了整節課無聊到都快把橡皮擦啃完了

身體姿勢彷彿說著:你的課實在無聊.

# 下課休息



不屑型  
disdain

新奇度 ★★☆☆☆

此姿勢常見於功課優秀, 不太需要聽課之學生  
連課都懶的聽, 自信似乎充滿了全身

身體姿勢彷彿說著: 你講的我都會啦.

# Today's Reading Assignment

- Cormen B.4 (Appendix) and 22.1-22.2