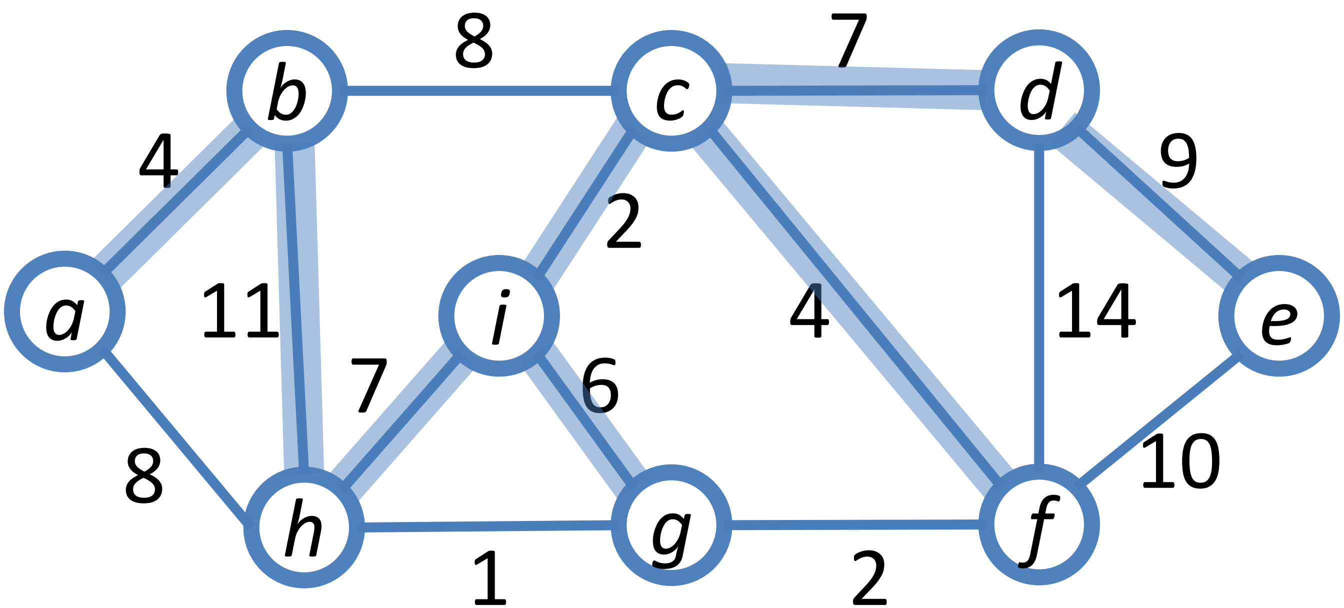
Minimum Spanning Trees
Chapter 23

Mei-Chen Yeh

A spanning tree

connects all of the vertices

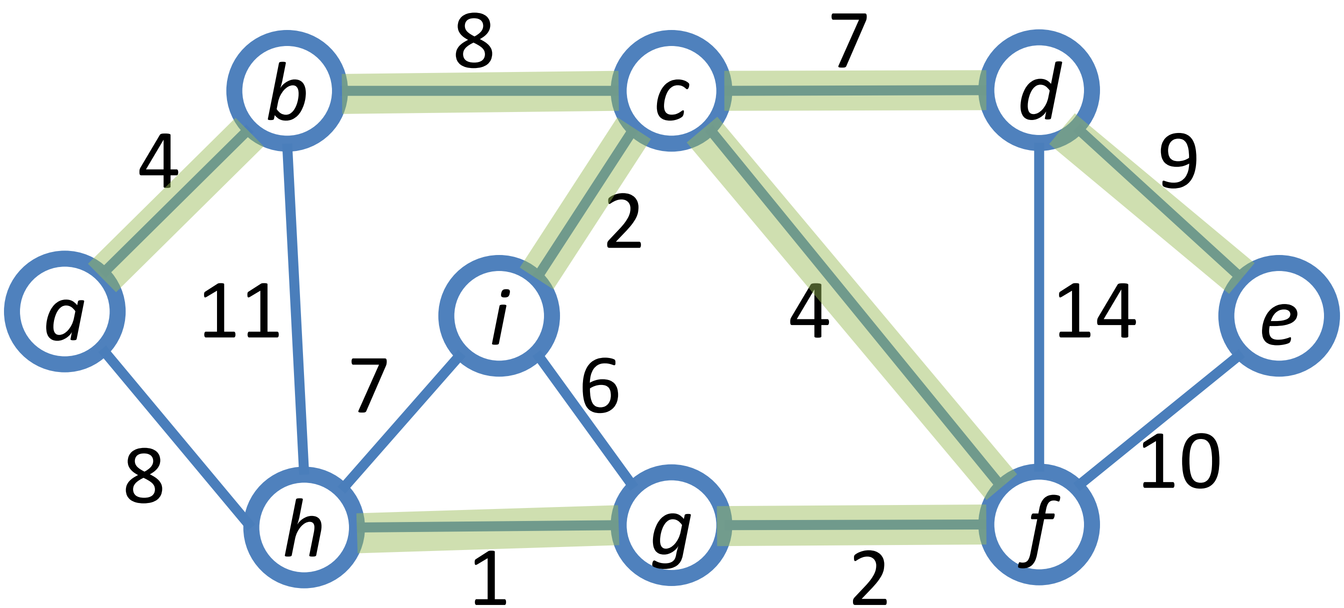


$w(T) = 50$

開始要處理權重，把所有選到的邊，上面的數字都加起來，就會是全部的cost

A spanning tree

connects all of the vertices

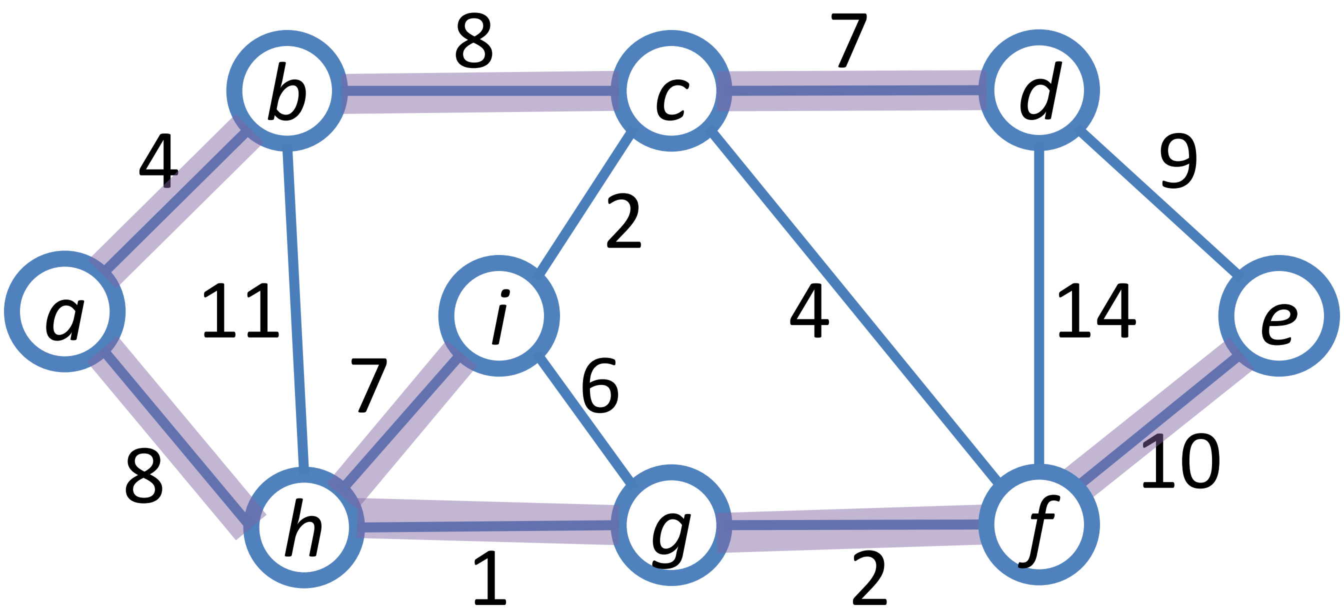


$w(T) = 37$

也是把所有選到的邊上的數字
都加起來，就是cost

A spanning tree

connects all of the vertices

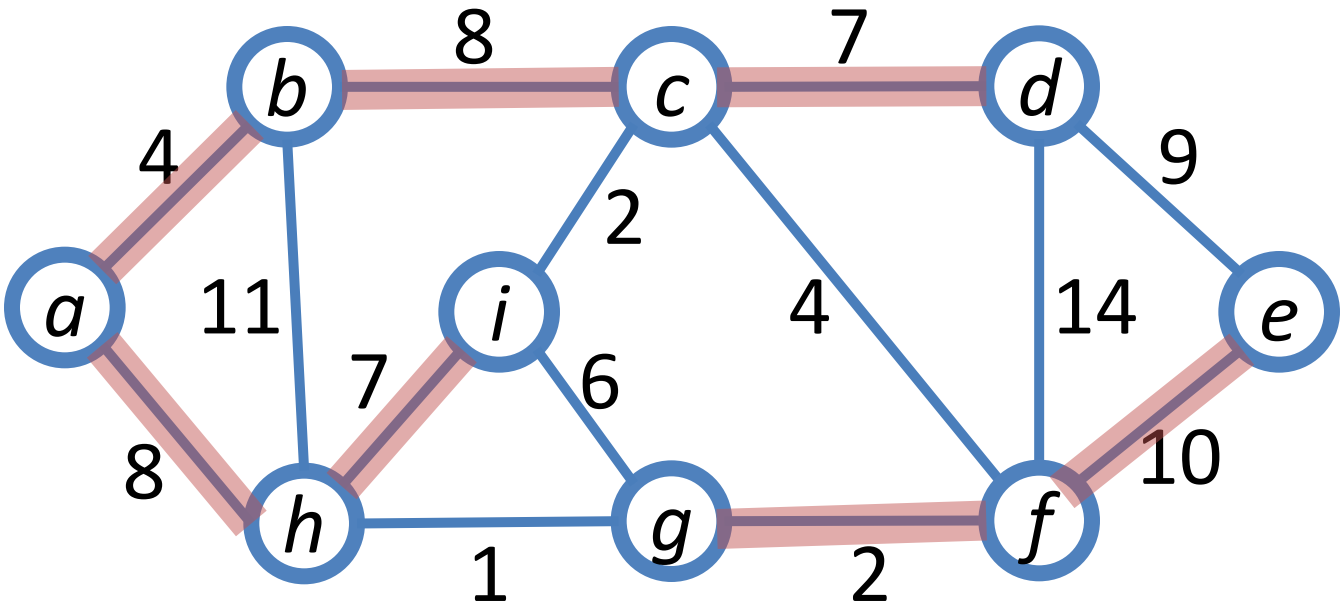


$$w(T) = 47$$

權重值是47

A spanning tree

connects all of the vertices

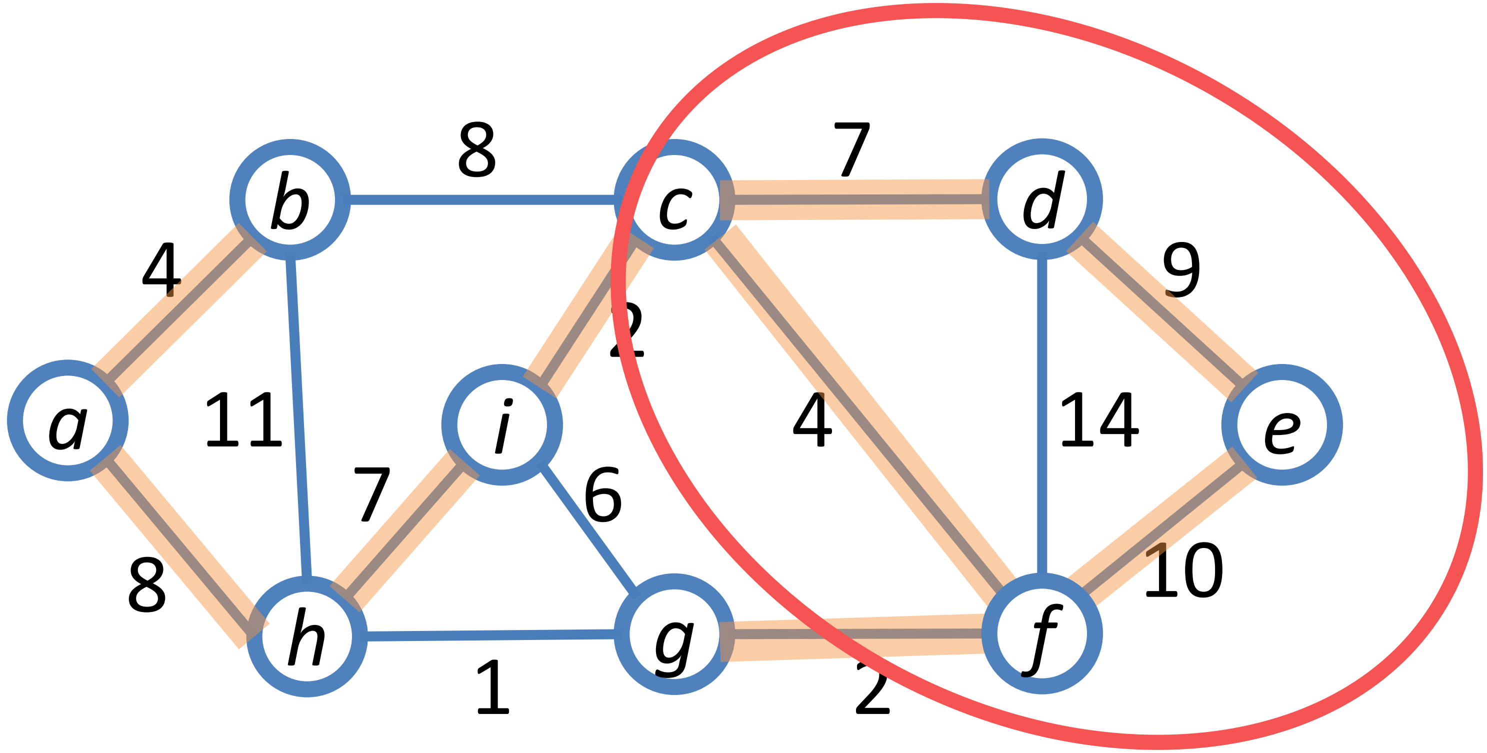


Two trees

這個不行，這是兩棵樹

A spanning tree

connects all of the vertices

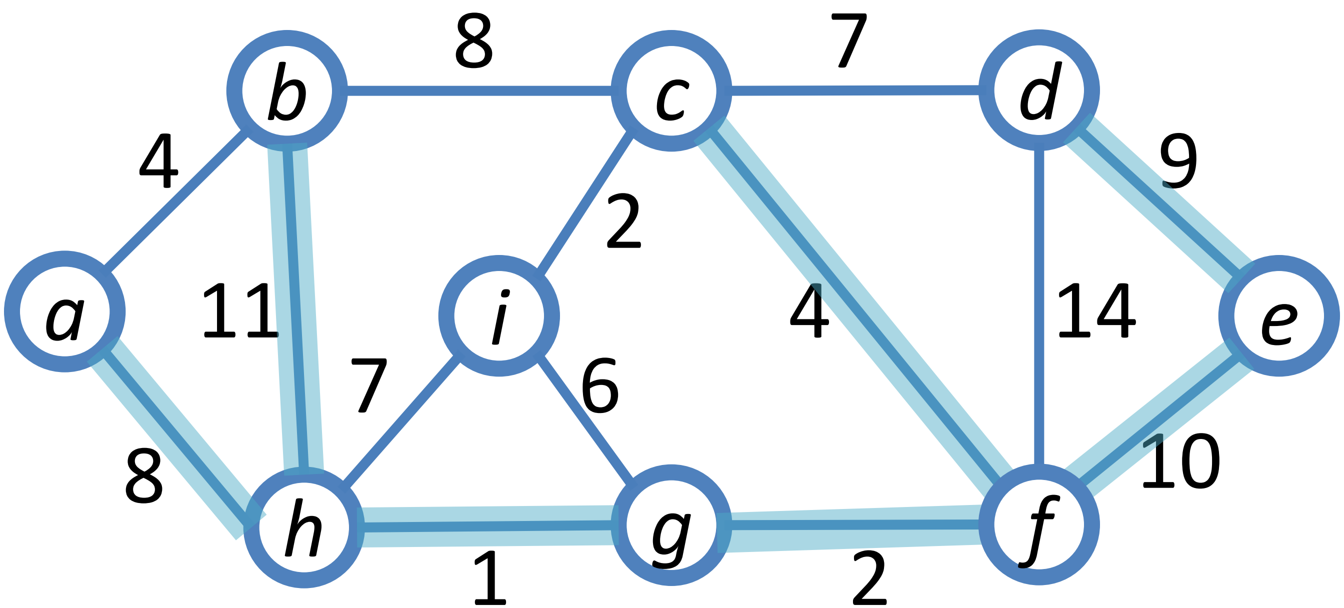


Not a tree

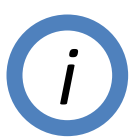
這個不是樹，他有迴圈

A spanning tree

connects all of the vertices



必須連接所有節點，
這裡節點 i 沒有被連到



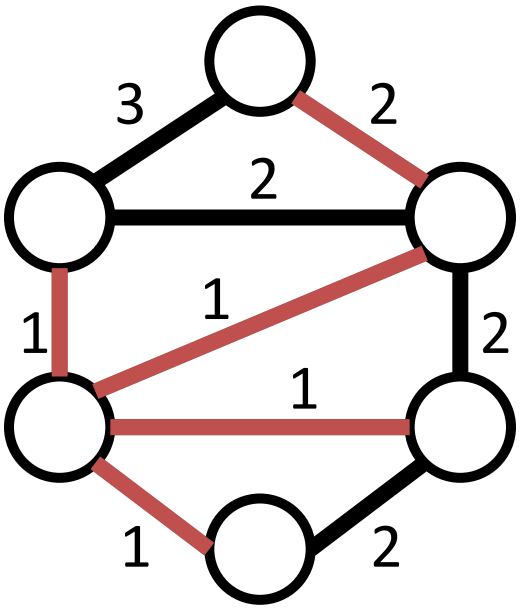
is not connected!

The minimum-spanning-tree problem

- Input:
  - A connected, undirected, ***weighted*** graph $G = (V, E)$
- Output:
  - An acyclic subset $T \subseteq E$ that connects all of the vertices and whose total weight is ***minimized***.

每個邊都可選可不選，要找出權重值最小的樹

One more example



每個點都要接起來，選總權重值最小的

$$w(T) = 6$$

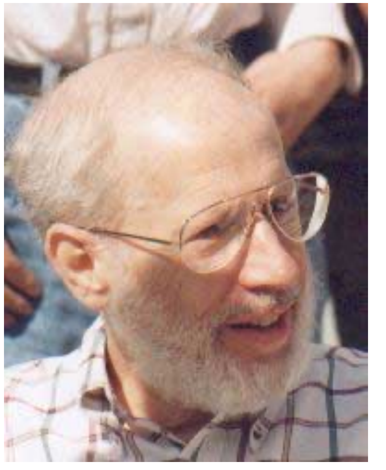
A little history

- Otakar Borůvka introduced the problem and wrote the original paper in 1926.
- The purpose was to efficiently provide electric coverage of Moravia.

Source: wikipedia.org

Two commonly used algorithms

- Kruskal's algorithm
- Prim's algorithm



$O(E \log V)$

Greedy methods

Growing a MST

- Grows the MST ***one edge at a time***

GENERIC-MST (G, w)

1. $A = \emptyset$

2. while A does not form a spanning tree

3. find an edge (u, v) that is safe for A

4. $A = A \cup \{(u, v)\}$

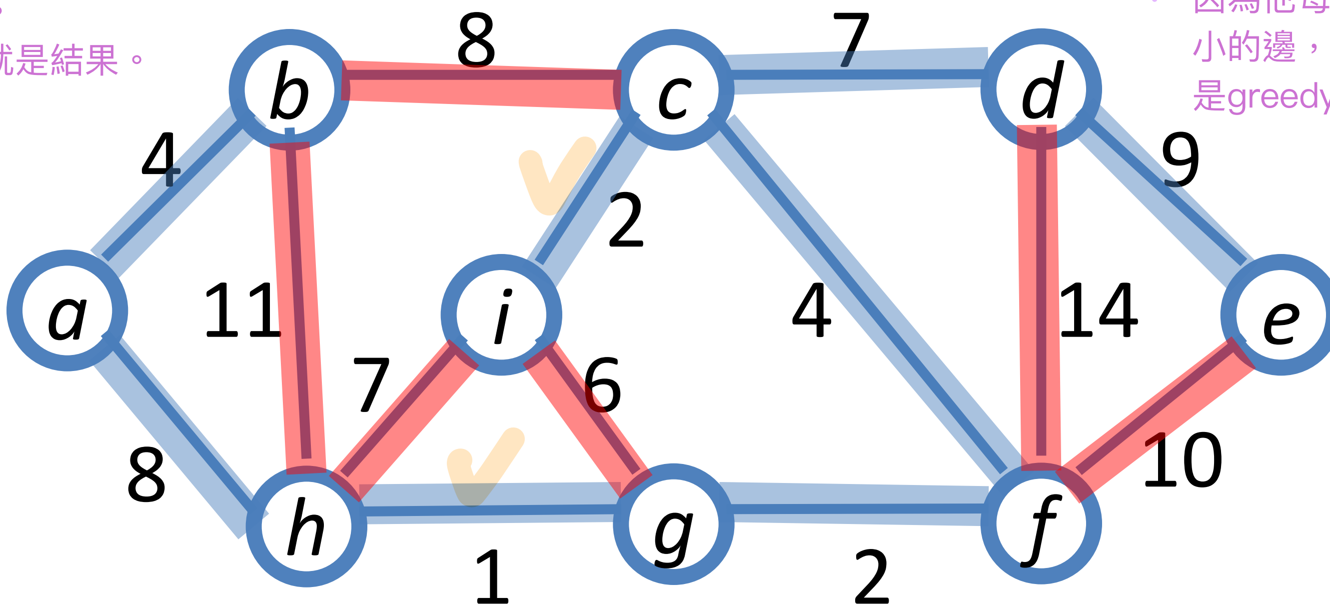
5. return A

Kruskal's algorithm

1. 先把所有邊列出來，從小排到大。
2. 放進A集合，樹已經開始長了。
3. 做到gi這一輪的時候，會造出一個迴圈，所以不行。
4. hi也會產生迴圈，也不行。
5. 以此類推。
6. 全部跑完就是結果。

Q. 為什麼是Greedy method?

A. 因為他每次都先挑權重最小的邊，不行再換，所以是greedy method



不能選的都是因為他會產生環

gh	ci	gf	ab	cf	gi	hi	cd	ah	bc	de	ef	bh	df
1	2	2	4	4	6	7	7	8	8	9	10	11	14

Kruskal's algorithm: Pseudo code

```
MST-KRUSKAL (G, w)
A = Ø
for each u ∈ G.V
    MAKE-SET(u)
sort the edges of G.E into non-decreasing order by weight w
for each edge (u, v) ∈ G.E, taken in non-decreasing order by w
    if FIND-SET(u) ≠ FIND-SET(v)  // check if (u, v) are in different sets
        A = A U {(u, v)}  // add the edge
        UNION(u, v)       // construct the union of the disjoint sets
                          // containing u and v
return A
```

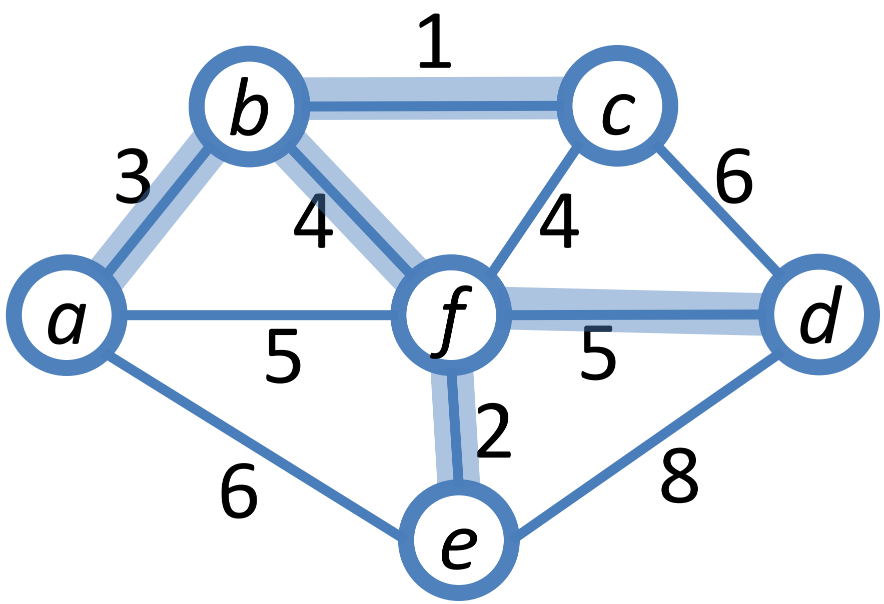
Example of the operations

- MAKE-SET(v): creates a one-element set {v} 初始化會用到
- FIND-SET(u): returns a subset containing u
  找出這個節點屬於哪一個集合
- UNION(u, v): constructs the union of the disjoint sets U and V containing u and v
  把包含 u 的集合，跟包含 v 的集合，取聯集
- *Example*
  - S = {1, 2, 3, 4, 5, 6}
  - {1}, {2}, {3}, {4}, {5}, {6} MAKE-SET(.)
  - {1, 4}, {2}, {3}, {5}, {6} UNION(1, 4)
  - {1, 4, 5}, {2}, {3}, {6} UNION(1, 5) or UNION(4, 5)

How to implement the operations?

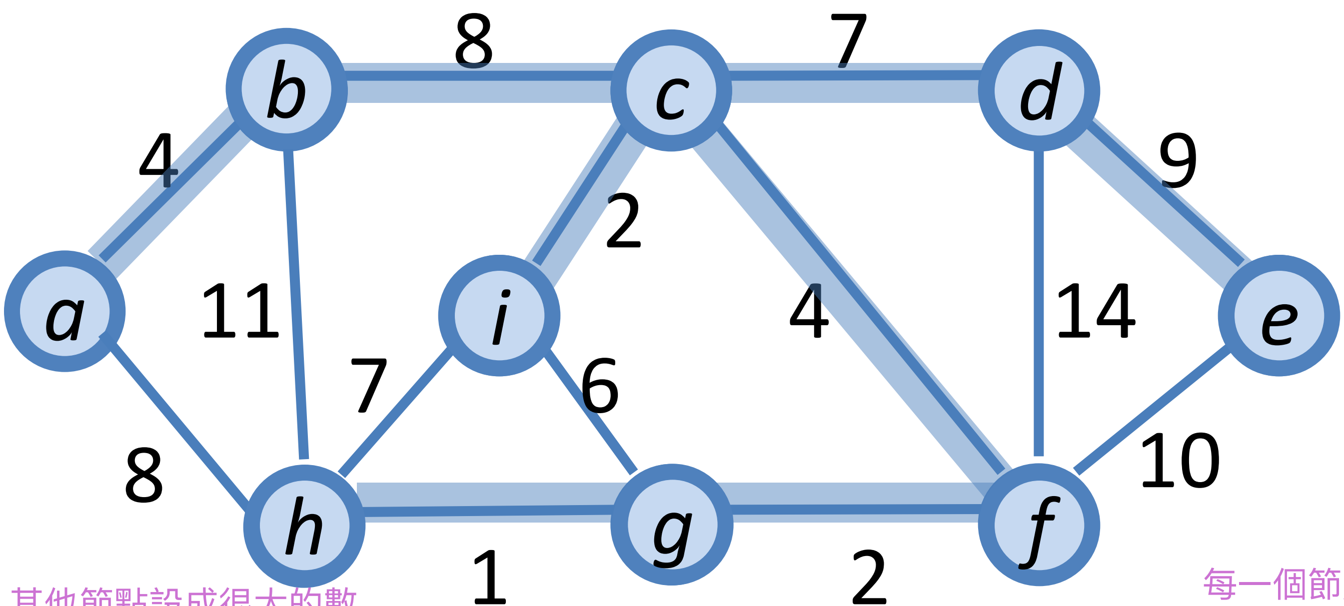
Exercise

- Find the minimum spanning tree using Kruskal's algorithm.



5/2 (=) End

Prim’s algorithm



每一個節點要記兩個東西，一個是權重值，跟上一個節點（從哪裡來的）

1. 把a設成0，其他節點設成很大的數
2. 在a可能去的節點（b, h）記錄下權重值
3. 選b~i裡面權重值最小的
4. 直到每個節點都被選到

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*
0	*a*,4	*b*,8	*c*,7	*d*,9	*c*,4	*f*,2	*g*,1	*c*,2

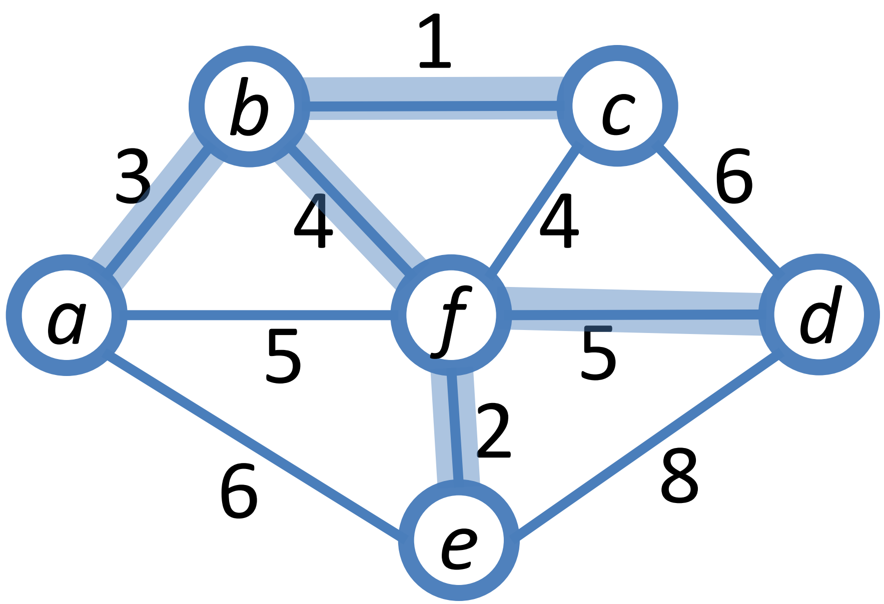
存的這些數字是該節點到這棵樹其中一節點的距離

Prim’s algorithm: Pseudo code

```
MST-PRIM (G, w, r)
for each u ∈ G.V                              // initialize
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ Ø
    u = EXTRACT-MIN(Q)                        // get one vertex for each iteration
    for each v ∈ G.Adj[u]                     // update the minimum weight
        if v ∈ Q and w(u,v) < v.key
            v.π = u
            v.key = w(u, v)
```

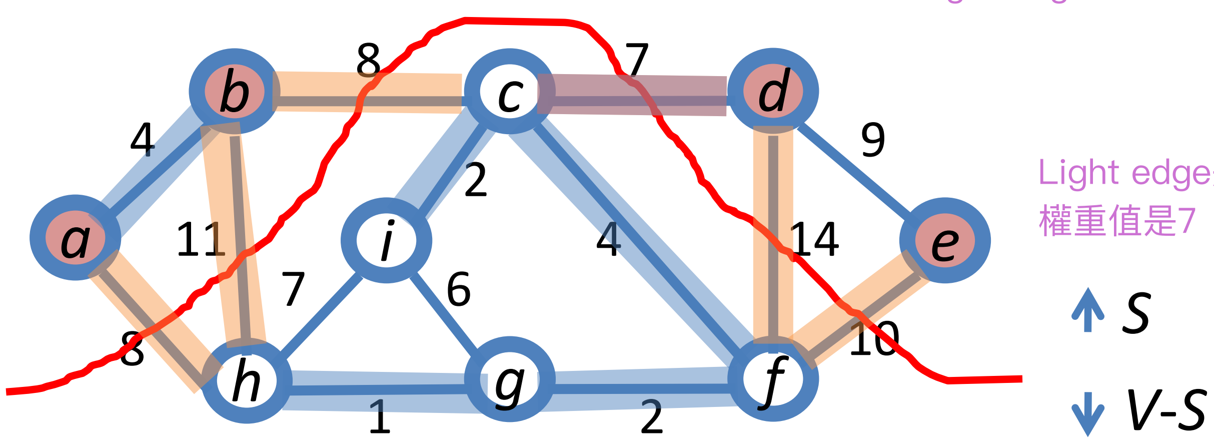
Exercise

- Find the minimum spanning tree using Prim's algorithm.



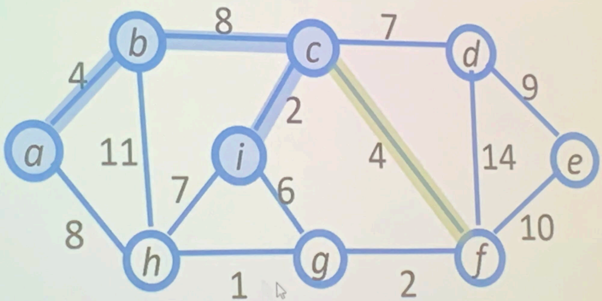
Correctness of the Algorithms

- **Cut** (S, V-S): a partition of V Cut：把集合分成兩坨
- A cut **respects** a set A of edges: no edge in A crosses the cut 不會一邊連接白色一邊連接粉紅色：當今天Ａ集合以外的邊都不會穿過Ａ，這樣就可以說這個cut respects A
- **Light edge**: a edge crossing the cut with the minimal weight 把粉紅色的點跟白色的點連上，挑出這些邊中權重值最小的，就是Light edge



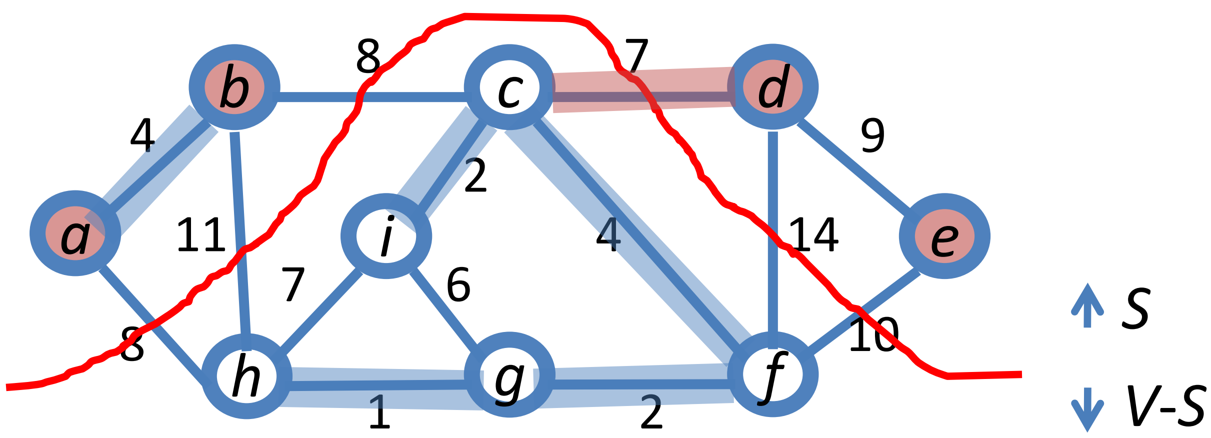
Light edge是c, d的邊，權重值是7

Example (Prim's)

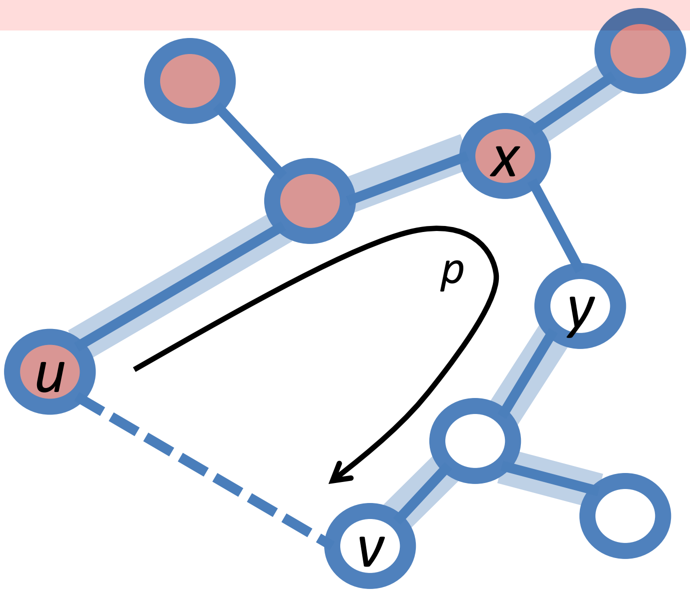


- Light edge?

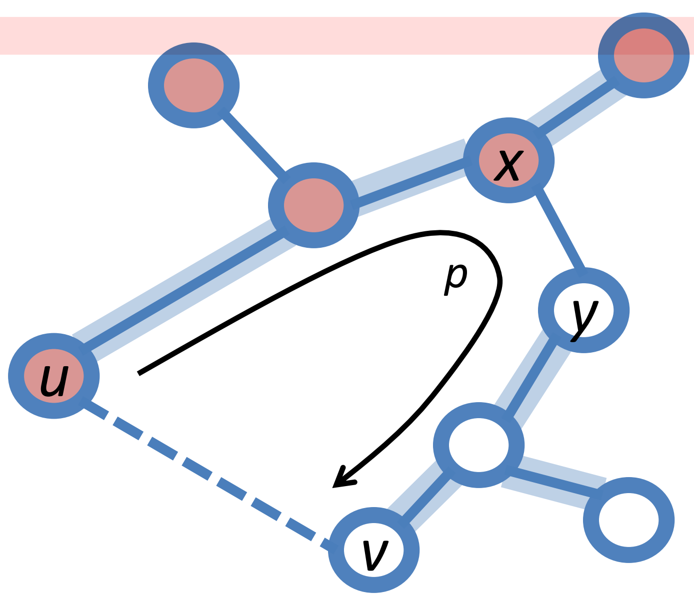
- Theorem
  - $G = (V, E)$: a connected, undirected, weighted graph
  - A: a subset of E that is included in some minimum spanning tree for G.
  - $(S, V\text{-}S)$: any cut of G that respects A
  - (u, v): a light edge crossing $(S, V\text{-}S)$
  - Then, **edge (u, v) is safe for A**.



- Proof
  - Let *T* be a MST that includes *A*, and assume that *T* does **not** contain the light edge (*u*, *v*). 虛線那條邊是Light edge
  - (*u*, *v*) forms a cycle with the edges on the simple path from *u* to *v* in *T*. (*why?*)
  - This cycle must contain (*x*, *y*) that is not in *A*. (*why?*)



- Proof
  - Removing (x, y) breaks T into two components.
  - Cut and paste! Create $T' = T - \{(x, y)\} \cup (\{u, v\})$.
  - $w(T') = w(T) - w(x, y) + w(u, v) \leq w(T)$
  - T' must be a minimum spanning tree also. (*why?*)
  - T' contains (u, v).



即使不是照貪婪法去得到這棵樹，也可以經由改造，把一個邊拔掉，把Light edge放進去，得到一棵minimum spanning tree

只有一個最佳解嗎？

jigsaw piece

5/4 (四) End