21.Minimum spanning tree
最小生成樹

中國文化大學
資訊工程學系 張耀鴻
112學年度第2學期

Scenario 情境

- A town has a set of houses and a set of roads.
- A road connects 2 and only 2 houses.
- A road connecting houses u and v has a repair cost $w(u,v)$
- ***Goal: Repair enough (and no more) roads such that***

1. everyone stays connected:

   can reach every house from all other houses, and

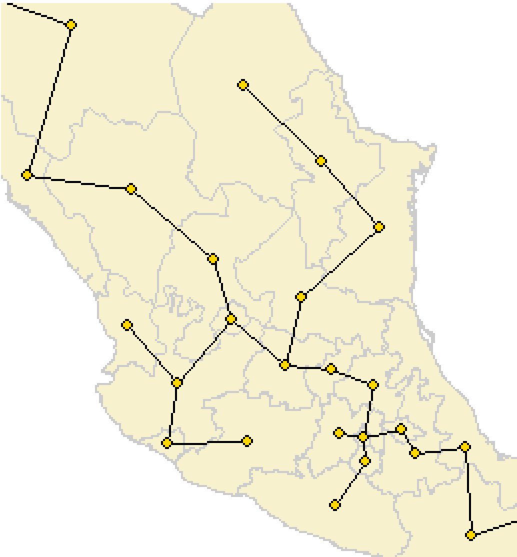
2. total repair cost is minimum.

Scenario 情境

- 某個小鎮有些房子和幾條路.
- 每條路只能連接2間房子.
- 房子 u 和 v 之間的路, 其維修成本為 $w(u,v)$
- ***目標: 儘可能維修最多的路並且滿足以下條件***

  1. 讓任何一間房子都能到達其他的房子 (connected), and

  2. 總維修成本最小.



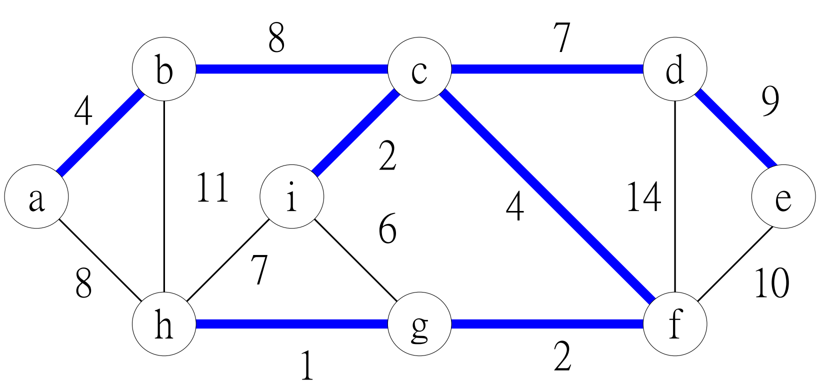
最小生成樹問題 (MST)

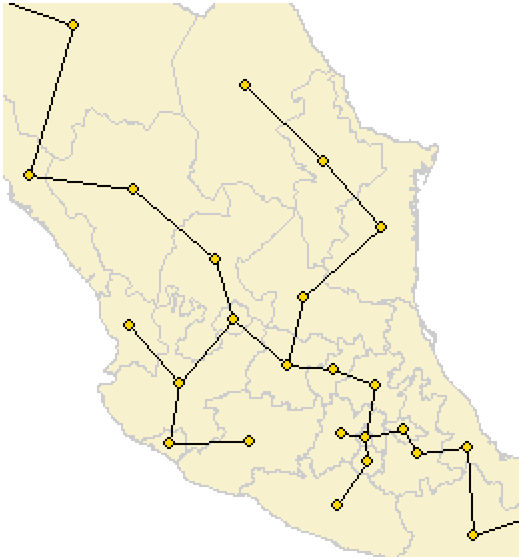
Minimum Spanning Tree Problem

Model as a graph:

一個加權圖之最小生成樹
1. 為此圖之生成樹
2. 且權重和為最小者

- Undirected graph $G = (V, E)$. 無向圖
- **Weight** $w(u,v)$ on each edge $(u,v) \in E$.
  每條邊上權重為 $w(u,v)$
- Find $T \subseteq E$ such that
  - ✓ 1. T connects all vertices (T is a **spanning tree**), and
  - ✓ 2. $w(T) = \sum_{(u,v)\in T} w(u,v)$ is minimized.





21.1 Growing a MST 建構MST(一般式)

Generic_MST(G, w)

1. $A \leftarrow \phi$
2. while (A 還不是生成樹)
3. do 找一條safe edge (u, v)
4. $\quad A \leftarrow A \cup \{(u, v)\}$
5. return A

1. 建構一個邊集合 A.
2. Initially, A 為空集合
3. 加入邊的過程中, 維持迴圈不變量 (**loop invariant)**:
   A 為 MST 的子集.
4. 加入新的邊時不可違背此原則:
5. 若新邊為安全邊,
   則加入後A仍為MST的子集
   i.e.,

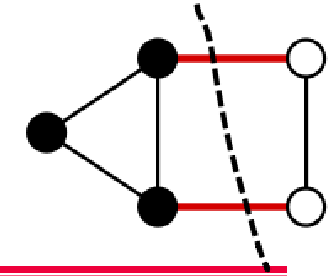
Edge (u, v) is safe

$\Leftrightarrow$

$A \cup \{(u, v)\}$ is a subset of MST

可參考 https://visualgo.net/en/mst 動畫解說

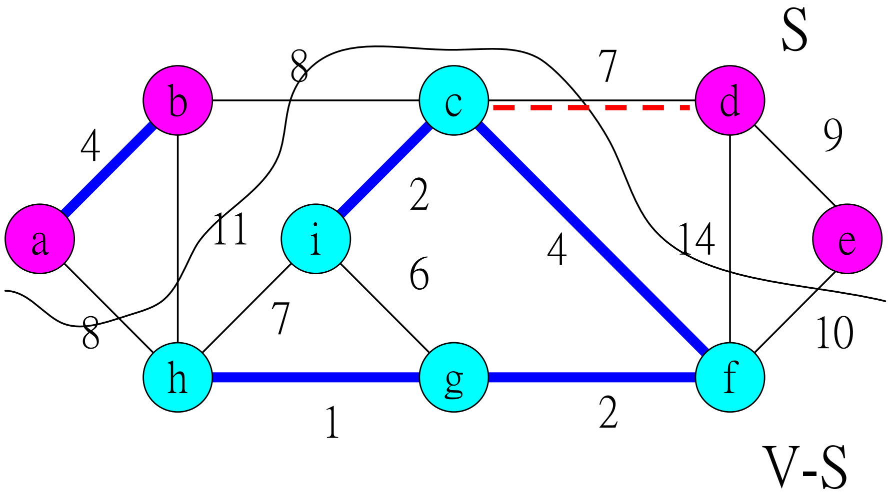
Some definitions 名詞定義

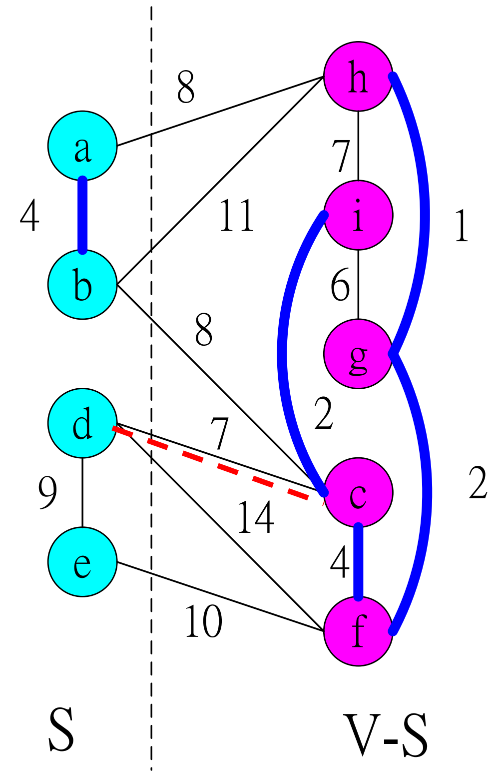


- A **cut** $(S, V - S)$ of an undirected graph $G = (V, E)$ is a partition of V.
  圖$G = (V, E)$上的cut將所有節點V分割為兩集合 S 和 $V - S$
- Edge (u,v) ***crosses*** the cut (S,V-S) if one of its endpoints is in S and the other is in V-S.
  邊 (u, v) 穿越 cut 表示兩頂點分別位於不同的集合 S 和 $V - S$
- A cut ***respects*** the set A of edges if no edge in A crosses the cut.
  一個 cut 尊敬某個邊所成的集合表示沒有邊會穿越這個cut
- An edge is a ***light edge*** crossing a cut if its weight is the minimum edge crossing the cut. (Note that there can be more than one light edge crossing a cut in case of ties.)
  所有穿越cut的邊中權重最小那個叫做輕邊

cut: $(S, V\text{-}S)$

light edge: min $(u,v) \in E$ cross the cut $(S, V\text{-}S)$





安全邊定理

Theorem 21.1 (Safe-edge theorem)

- Let A be a subset of some MST, $(S, V - S)$ be a cut that respects A, and (u, v) be a light edge crossing the cut.

令A為MST之子集
(S, V-S)為尊敬A的cut
(u,v)為穿越此cut的輕邊

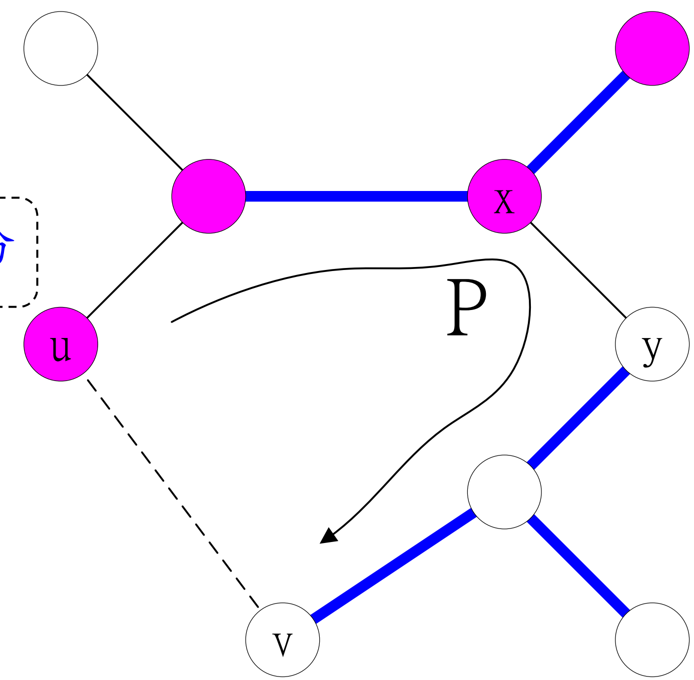
Then (u, v) is safe for A.

穿越cut的輕邊很安全

貪婪特性: 假設區域最佳解為全域最佳解的一部份

Greedy-choice property

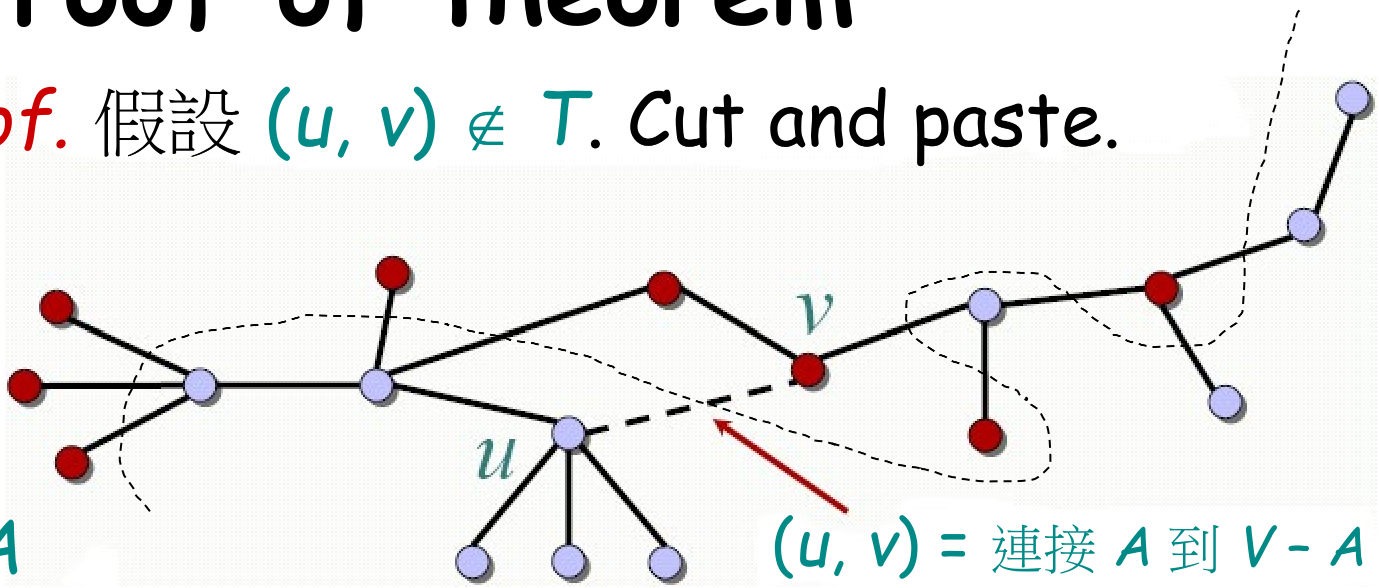
A locally optimal choice is globally optimal.



Proof of theorem

Proof. 假設 $(u, v) \notin T$. Cut and paste.

T:



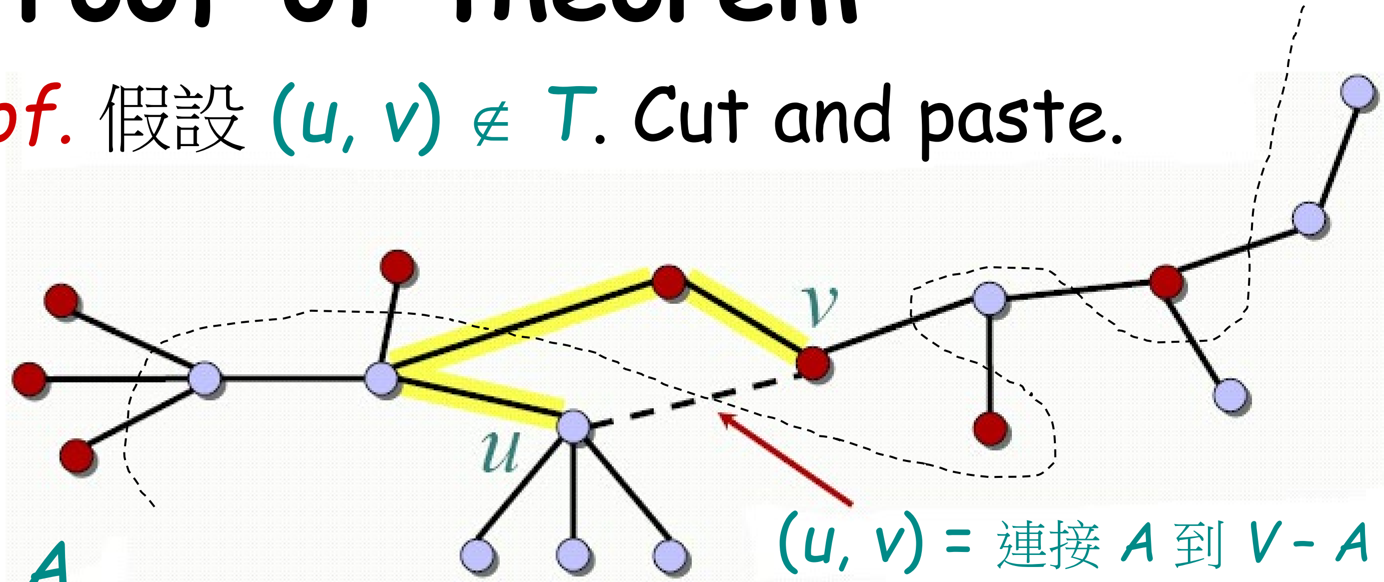
(u, v) = 連接 A 到 $V - A$ 的輕邊

- $\in A$
- $\in V - A$

Proof of theorem

Proof. 假設 $(u, v) \notin T$. Cut and paste.

T:



● $\in A$

● $\in V - A$

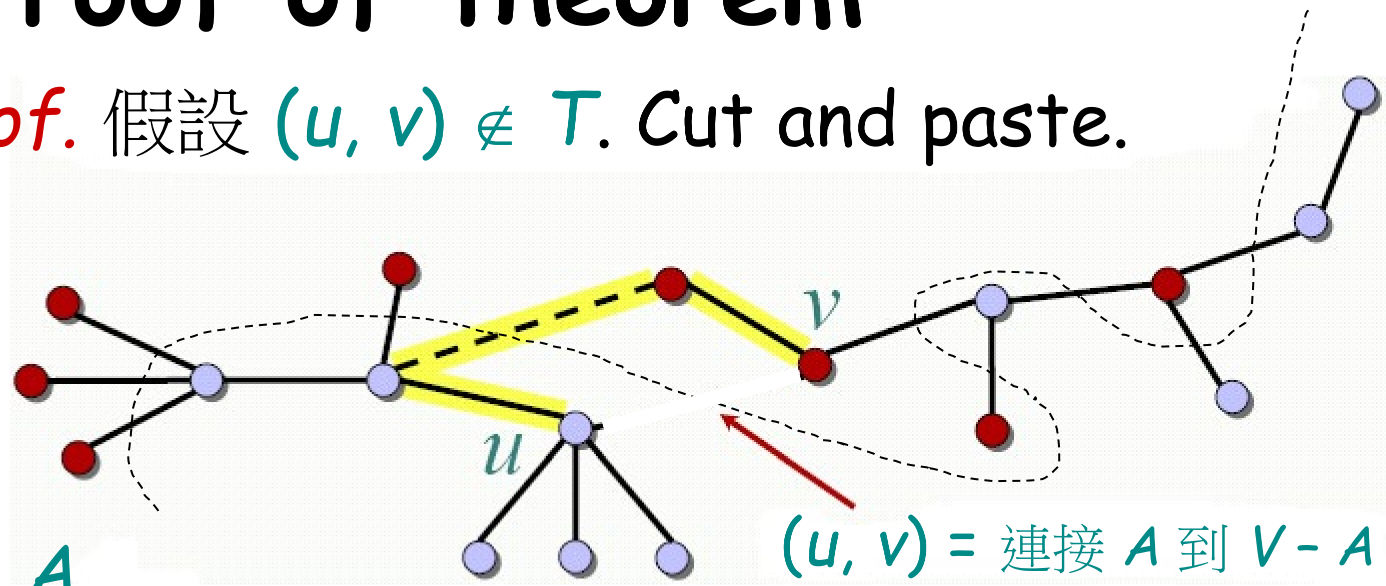
(u, v) = 連接 A 到 $V - A$ 的輕邊

考慮 T 中一條從 u 到 v 的簡單路徑

Proof of theorem

Proof. 假設 $(u, v) \notin T$. Cut and paste.

T:



- $\in A$
- $\in V - A$

(u, v) = 連接 A 到 $V - A$ *的輕邊*

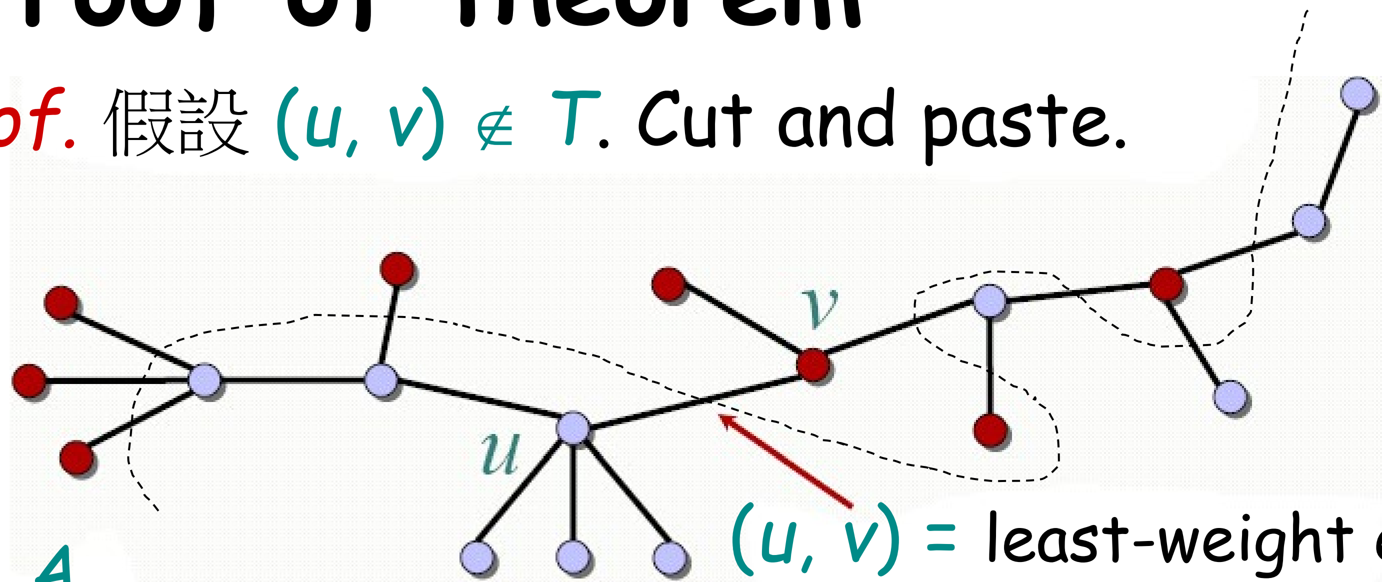
考慮 T 中一條從 u 到 v 的簡單路徑

把該路徑中連接 A 與 $V - A$ 的邊以 (u, v) 取代

Proof of theorem

Proof. 假設 $(u, v) \notin T$. Cut and paste.

T:



$\in A$

$\in V - A$

(u, v) = least-weight edge connecting A to $V - A$

考慮 T 中一條從 u 到 v 的簡單路徑

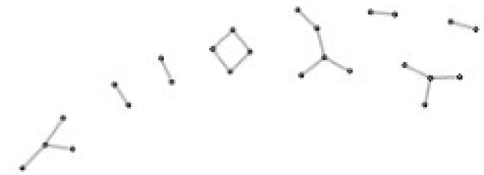
把該路徑中連接 A 與 $V - A$ 的邊以 (u, v) 取代

最後得出一個權重比 T 更小的生成樹. ■

有關 Kruskal 演算法的初步想法

Corollary 21.2 (Idea of Kruskal alg)

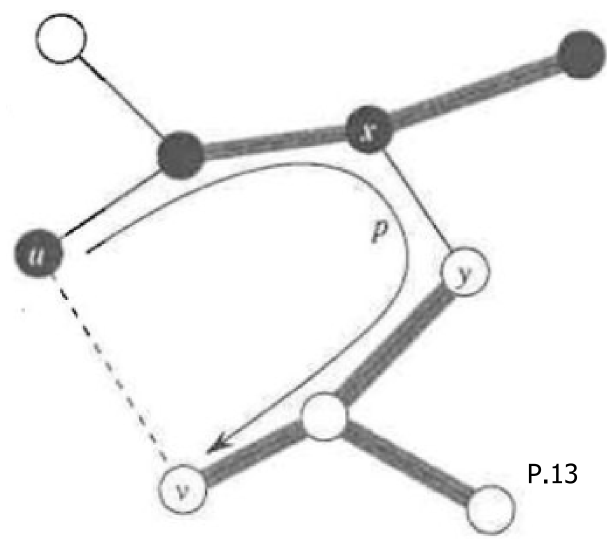
- A is a subset of some MST　　A 為 MST 之子集
- V_C is a set of vertex in a connected component in $G_A = (V, A)$　　V_C 為A中連通節點的集合



- **If** (u, v) is a light edge crossing the cut $(V_C, V - V_C)$　　若 (u, v) 為穿越 cut $(V_C, V - V_C)$ 的輕邊

  Then (u, v) is safe for A.　　則 (u, v) 為 A 之安全邊

Proof: Set S = V_C, in theorem 21.1



21.2 The algorithms of Kruskal and Prim

- **Kruskal's algorithm**
  - Greedy approach 以貪婪方式求解
  - Pick an edge with minimum weight into spanning tree and repeat 每次都選取權重最小的邊加入 MST
  - Make sure there is no cycle during the process 過程中確保不會產生循環
- **Prim's algorithm**
  - Build one tree A from an arbitrary "root" r 任意挑一個根節點
  - Maintain $V - V_A$ as a priority queue Q 以 priority queue 維護不在生成樹中的節點
  - Pick a light edge crossing cut $(V_A, V - V_A)$ into spanning tree and repeat 每次都選取權重最小的輕邊加入 MST
  - Make sure there is no cycle during the process 過程中確保不會產生循環

Kruskal's algorithm

MST-KRUSKAL(G, w)

```
A ← ϕ
for each vertex v ∈ G.V
    do MAKE-SET(v)
sort the edge of E by nondecreasing weight w
for each edge (u,v) ∈ E, taken from the sorted list
    do if FIND-SET(u) ≠ FIND-SET(v)
        then A ← A ∪ {(u,v)}
             UNION(u, v)
return A
```

$O(1)$ 一開始MST內無節點

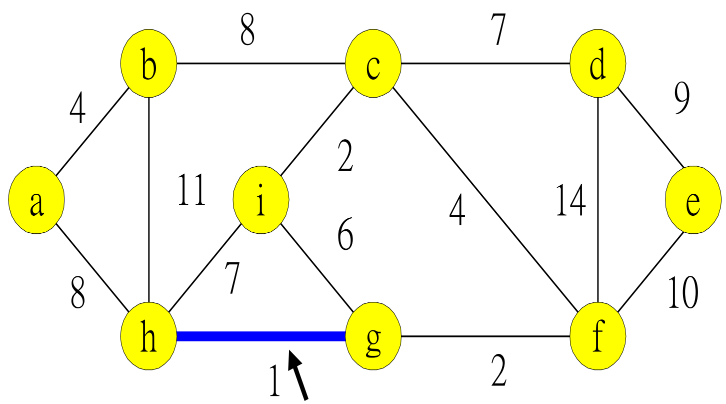
$|V|$ MAKE-SETs 每一個節點自成一個獨立集合

$O(E \lg E)$ 把所有邊依權重排序

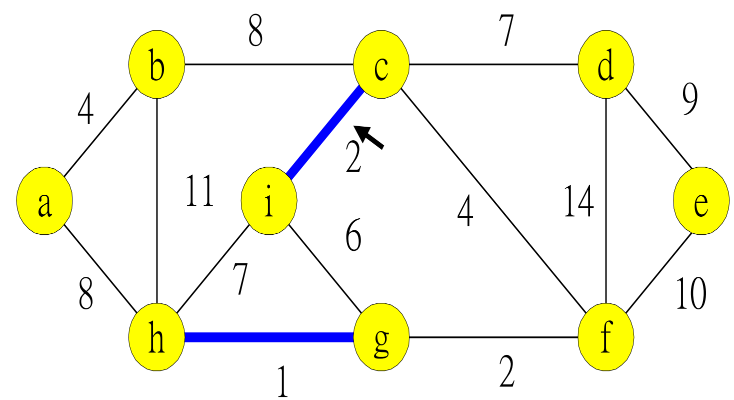
逐一檢查已排序的邊(u, v),
若u, v 在不同的集合，
則將(u, v) 納入 MST,
並將u, v 併入同一集合中

$O(E)$ FIND-SET
and UNION, $O(\alpha(V))$ each

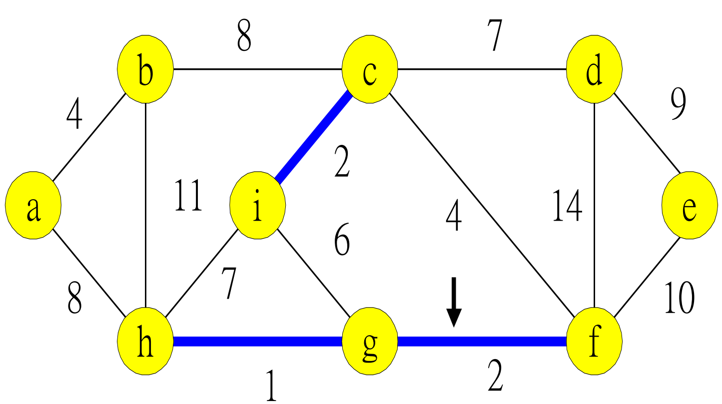
Complexity $O(E \lg E)$



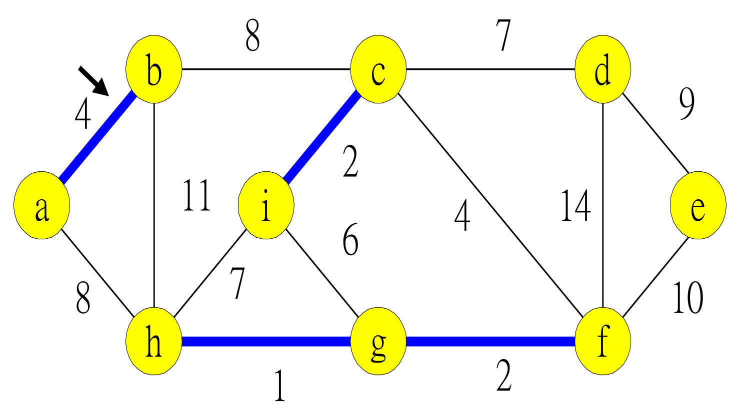
b
8
c
7
d
4
9
2
a
11
i
4
14
e
6
8
7
10
h
g
f
1
2



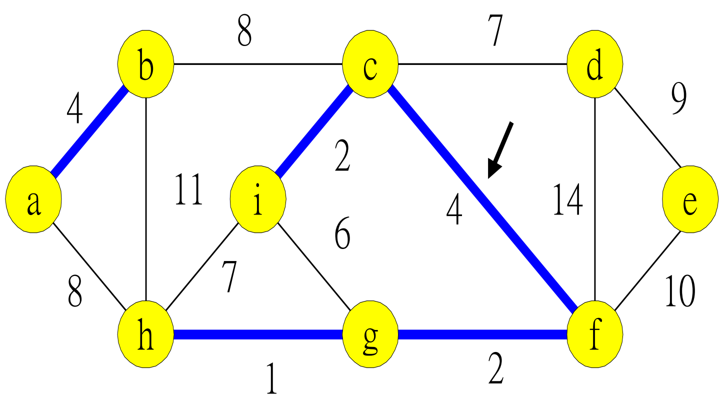
b
8
c
7
d
4
9
2
a
11
i
4
14
e
6
8
7
10
h
g
f
1
2

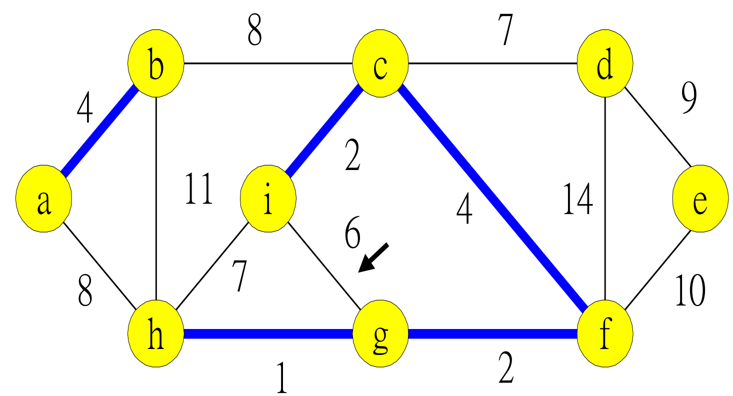


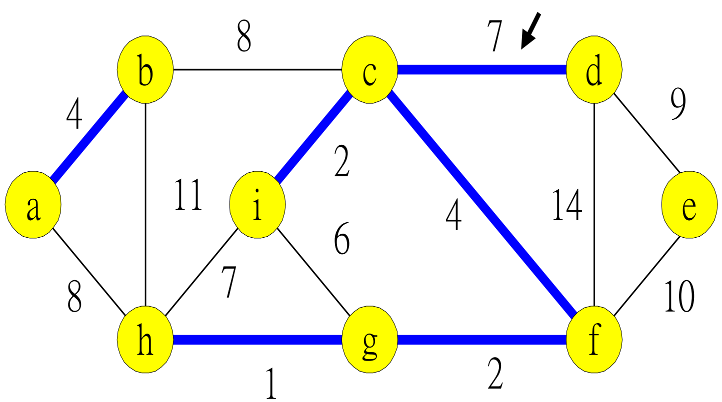
b
8
c
7
d
4
9
2
a
11
i
4
14
e
6
8
7
10
h
g
f
1
2

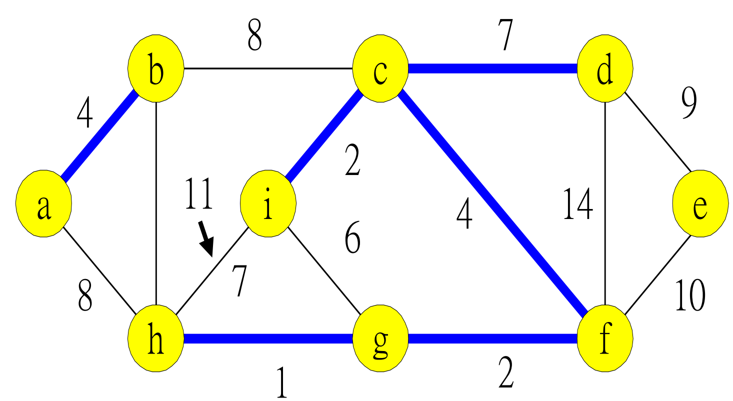


b
8
c
7
d
4
9
2
a
11
i
4
14
e
6
8
7
10
h
g
f
1
2









To be continue ……

Prim's algorithm

MST-PRIM(G, w, r)

```
Q = ϕ
for each u ∈ G.V
do  u.key = ∞
        u.π = NIL
        INSERT(Q, u)
DECREASE-KEY(Q, r, 0)
while Q ≠ ϕ
do u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
    do if  v ∈ Q and w(u, v) < v.key
        then v.π = u
            DECREASE-KEY(Q, v, w(u, v) )
```

將所有節點插入PQ中 $O(V \lg V)$

把根節點r的鍵值設成0 $O(\lg V)$

取出PQ中鍵值最小的節點u,
檢查所有Q中與u相連的節點

$|V|$ EXTRACT-MIN ➔ $O(V \lg V)$

$|\wedge$

$|E|$ DECREASE-KEY ➔ $O(E \lg V)$

若該邊的權重比v目前的鍵值更小,
則令u為v的前一點, 並將v的鍵值
改成新的權重

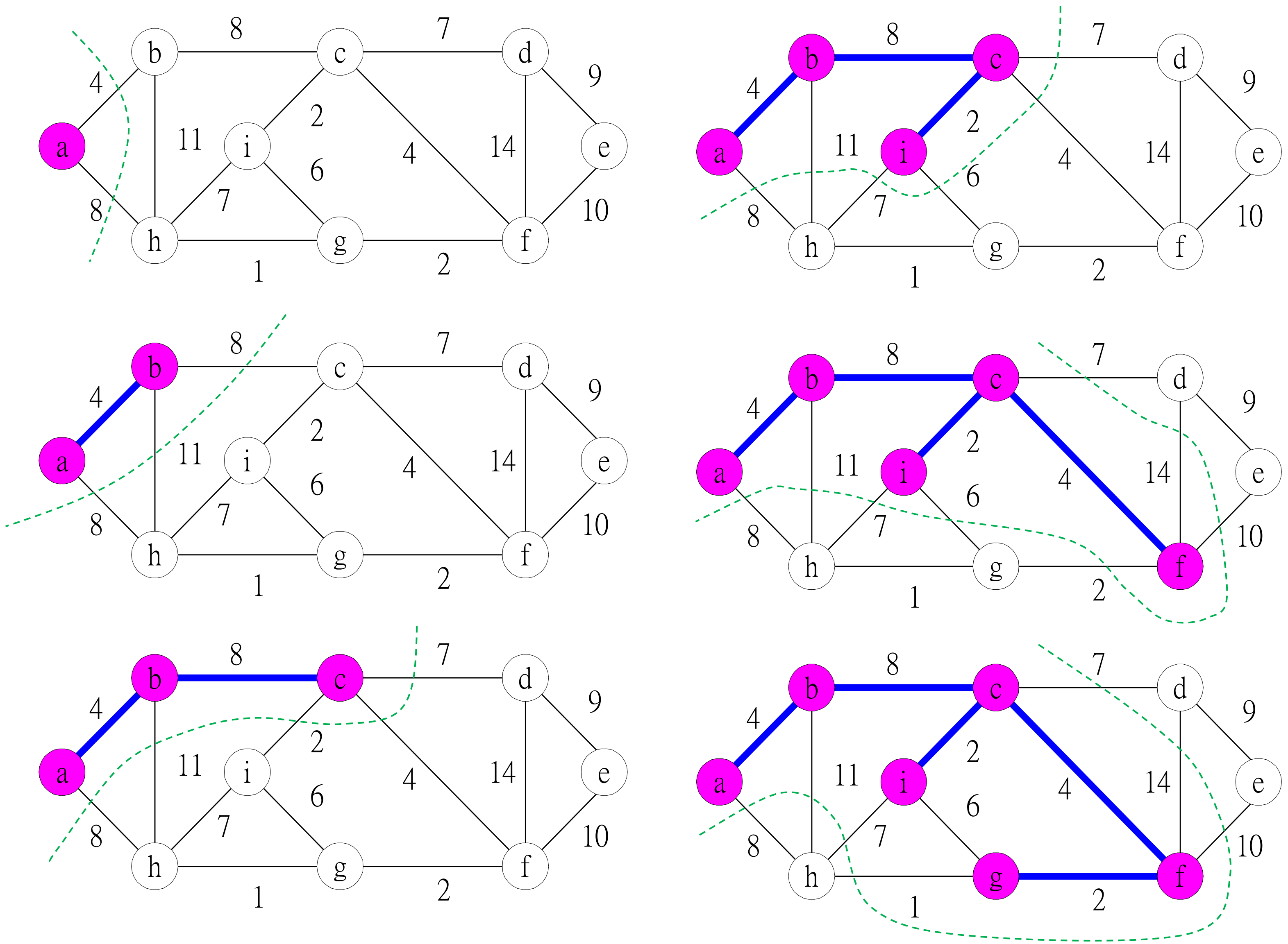
Complexity: $O(E \lg V)$

NOTE:
若將 **priority queue** 改成
Fibonacci heap (Ch 19),
則 DECREASE-KEY 可在
O(1) 時間內***攤銷***完畢
➔ **Complexity: $O(V \lg V + E)$**

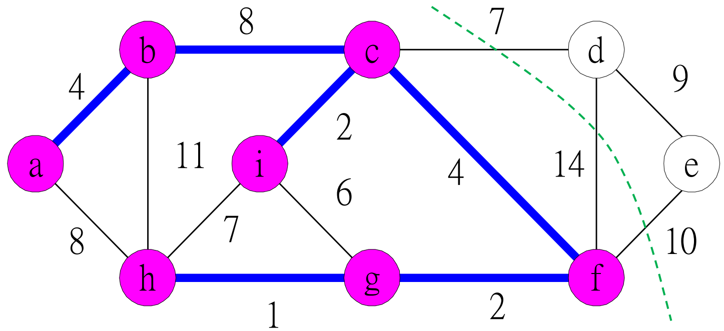
Analysis of Prim

Time = $\Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}$

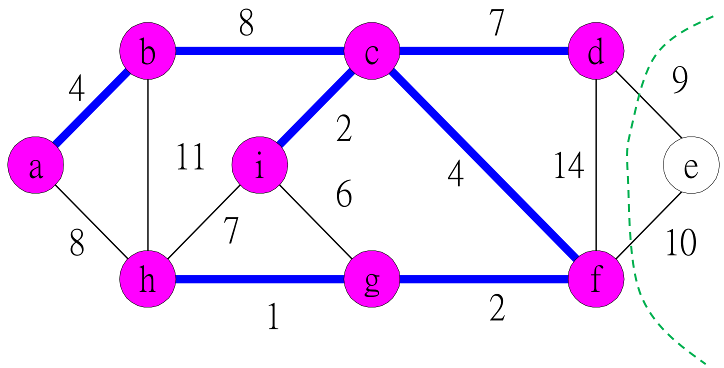
Q	$T_{\text{EXTRACT-MIN}}$	$T_{\text{DECREASE-KEY}}$	Total
array	$O(V)$	$O(1)$	$O(V^2)$
binary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$
Fibonacci heap	$O(\lg V)$ amortized	$O(1)$ amortized	$O(V \lg V + E)$ worst case



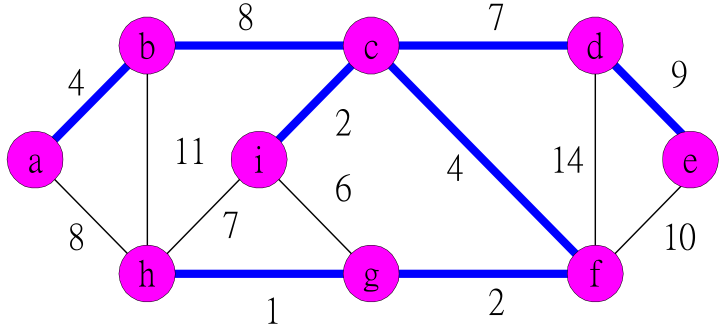
b c d a i e h g f
8 7 4 11 2 9 4 14 6 8 7 10 1 2
b c d a i e h g f
8 7 4 11 2 9 4 14 6 8 7 10 1 2
b c d a i e h g f
8 7 4 11 2 9 4 14 6 8 7 10 1 2
b c d a i e h g f
8 7 4 11 2 9 4 14 6 8 7 10 1 2
b c d a i e h g f
8 7 4 11 2 9 4 14 6 8 7 10 1 2
b c d a i e h g f
8 7 4 11 2 9 4 14 6 8 7 10 1 2



8
7
9
4
2
11
14
6
4
8
7
10
1
2



8
7
9
4
2
11
14
6
4
8
7
10
1
2



8
7
9
4
2
11
14
6
4
8
7
10
1
2

Summary

- Minimum spanning tree
- Definitions
  - ✓ safe, cut, cross, respect, light edge
- Theorem 21.1: Safe edge theorem
- Kruskal's algorithm **$O(E \lg E)$**
- Prim's algorithm **$O(E \lg V)$**

→ $O(V \lg V + E)$ Fibonacci heap