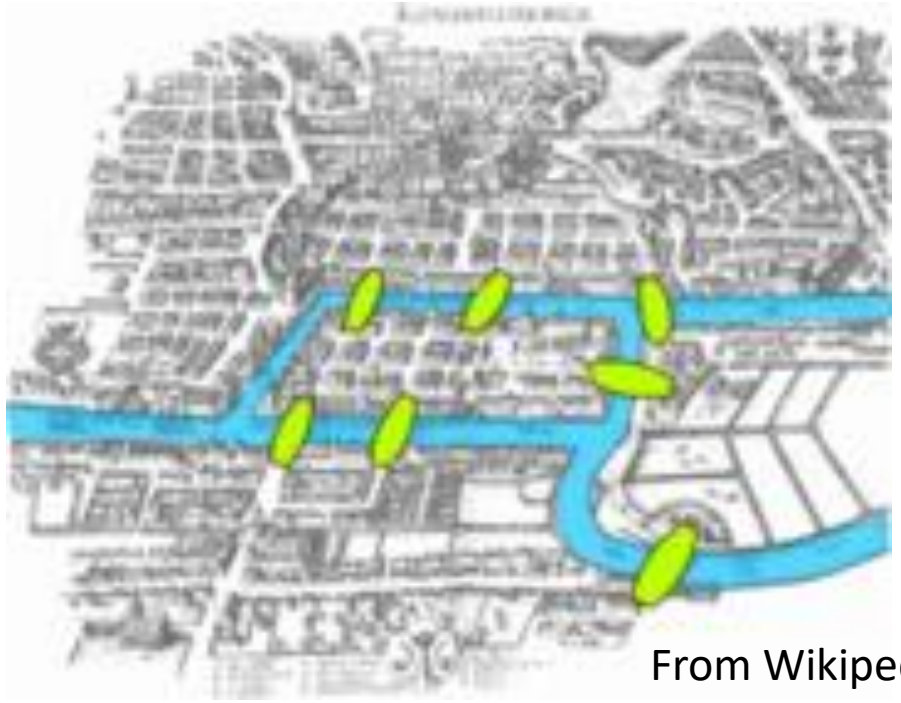
Graph Algorithms
Euler Circuit
Hamilton Circuit

for Sprout 2014 by Chin Huang Lin

Sprout

七橋問題

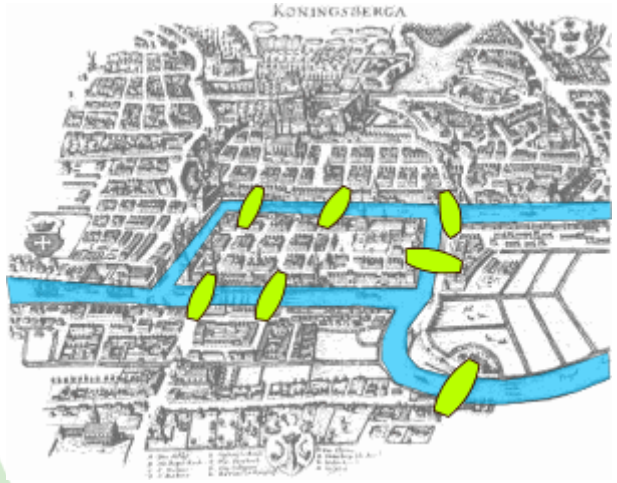
- 柯尼斯堡有七座橋，風光明媚，景色迷人
- 如果可以安排一條旅遊路線，恰好把七座橋都經過一次，那就不腳痠不無聊，一路順暢身心富足
- 只是，一直都沒有人找到滿足要求的路線

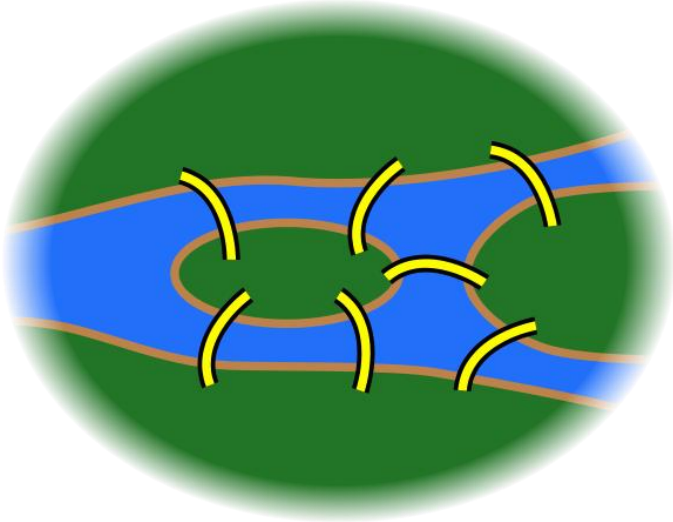


From Wikipedia

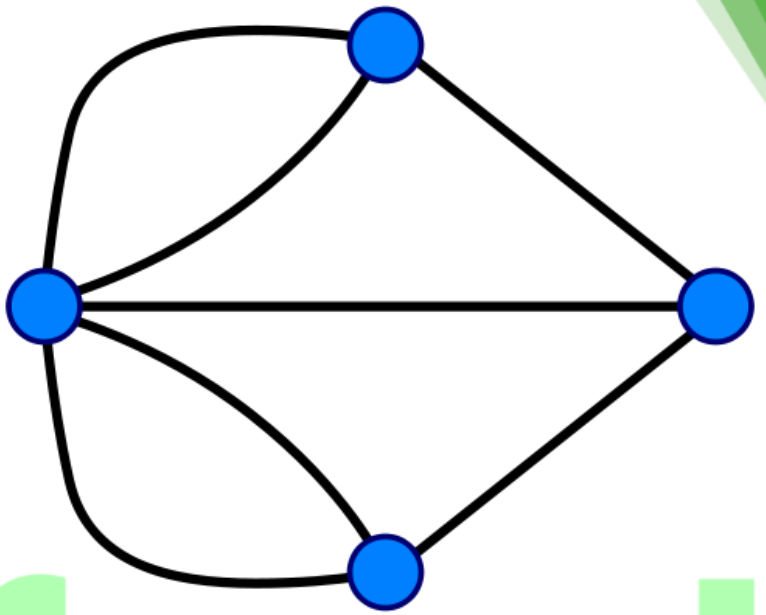
轉成圖論問題的話……

- 給定一個無向圖，是否存在一條每條邊恰走過一次的路徑？



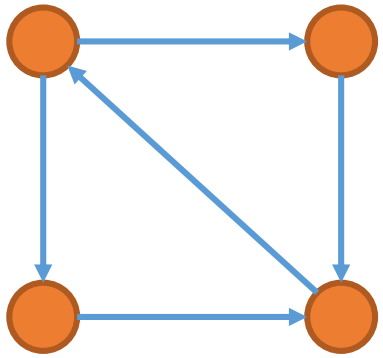


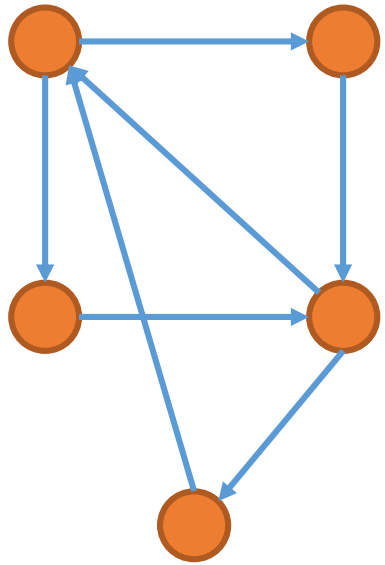
From Wikipedia



一筆畫問題！

- 一筆畫問題有分成兩種：
  - 一種形成一條路徑，稱為 *Euler path*
  - 一種形成一個環，稱為 *Euler circuit*





觀察一下

- 顯然至少所有的邊要互相連通
- 按照路徑經過的方向，可以把無向邊有向化，分成入度與出度
- 不難發現，除了起點與終點以外，所有的點的入度數一定要與出度數相同！
  - 如果入度 > 出度，代表有進去卻沒出來的情況，那麼該點應該是終點
  - 如果出度 > 入度，代表有莫名無中生有的情況，那麼該點應該是起點
- 入度與出度相同，也就是說原先的無向圖上一定度數為偶數
- 以上是必要條件

驗證一下

- 除了起點與終點外，所有的點度數都為偶數

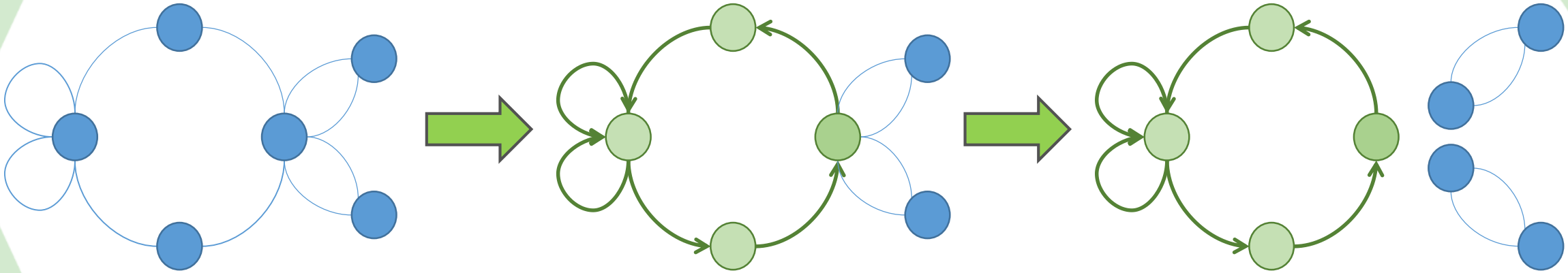
→ 整張圖上只能有 0 個或 2 個偶點

- 這樣是否足夠充分呢？
  - Yes!
  - 底下我們用簡單的數學歸納法來給出構造性證明

驗證一下

- 遞迴證明法！
- 當邊數只有 0 或者 1 時，結論顯然成立
- 假設邊數 $\leq k$ 時成立，那麼對於一個邊數為 $(k+1)$ 的圖……
  - 假設 $(k+1)$ 是偶數 (此時沒有奇點)，那麼圖上必定存在環 C。把 C 自圖上移除後，由於任意點都只會移除偶數度，剩餘圖必定形成所有點度數仍為偶數的若干連通塊。
  - 由於 $\leq k$ 時成立，剩餘的每個連通塊都存在歐拉迴路。
  - 把各個連通塊的歐拉迴路都「接在」與 C 的共同節點上，即形成原圖的歐拉迴路

驗證一下



From DJWS

驗證一下

- 遞迴證明法！
- 當邊數只有 0 或者 2 時，結論顯然成立
- 假設邊數 $\leq k$ 時成立，那麼對於一個邊數為 $(k+1)$ 的圖……
  - 假設 $(k+1)$ 是奇數 (此時有二奇點)，那麼先在兩奇點之間新增一假想邊 e 形成圖 G'，接著一樣在圖上選擇一個環 C 移除
  - 因為環上邊數至少為 2，$(k+1)+1-2 \leq k$，又可以遞迴求解了！
  - 完成 G' 的歐拉迴路後，把 e 移除，並且選擇兩奇點作為起點終點即可

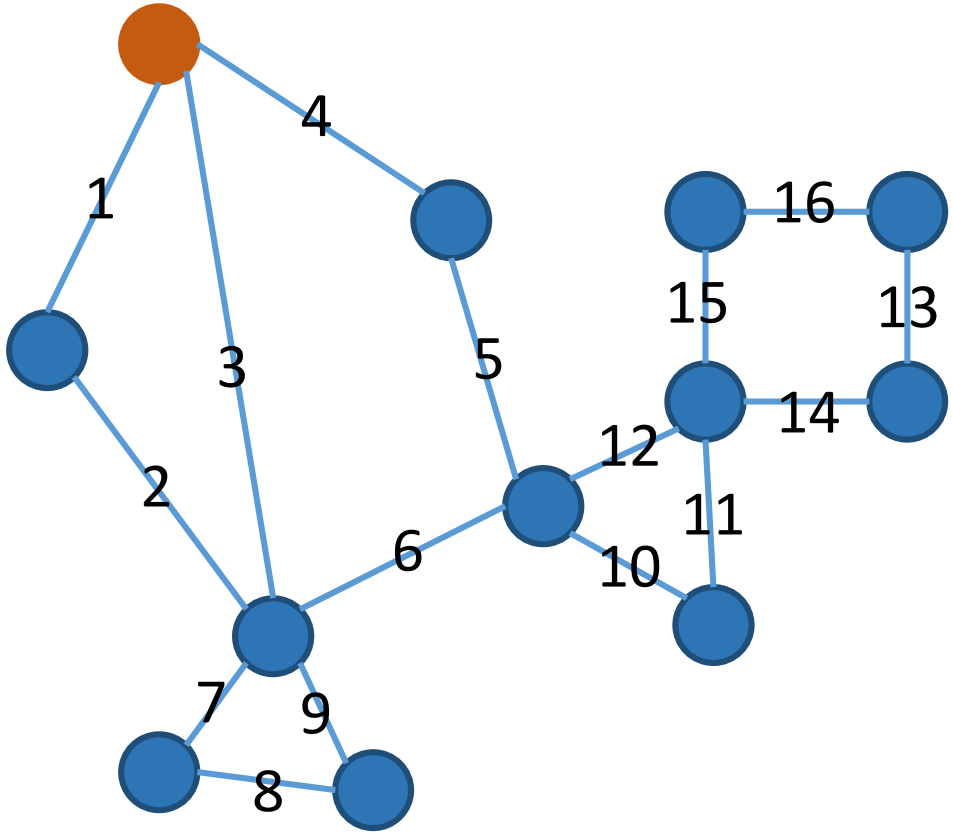
實作時間

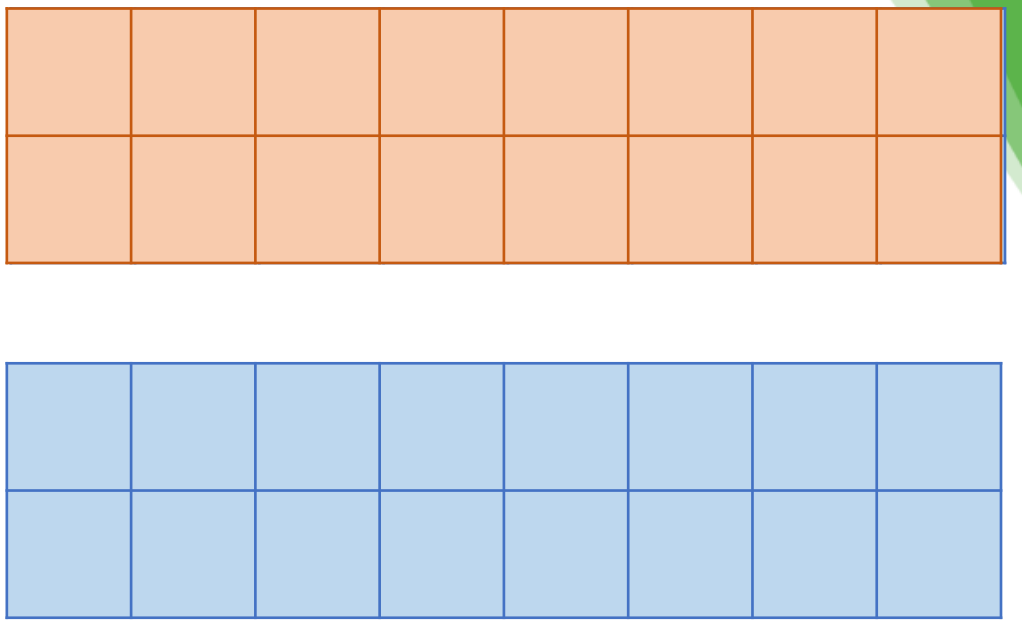
- 在證明中我們可以看到，其實 C 怎麼選，根本就不重要！
- 只要起點、終點選對，其實我們隨便走也可以！
- 走到山窮水盡，就代表一個 C (有可能包含虛擬邊 e)；接著遞迴把剩下的部份的歐拉路「黏」到 C 上即可

算法框架

1. 判斷奇點個數，若
   - ＞2，那麼無解
   - ＝2，則選擇其中一個奇點作為起點
   - ＝0，則選擇任意一個點作為起點
2. 在 DFS 框架內執行下列步驟
   1) 若當前節點還有尚未走過的邊，那麼拜訪該邊，並在拜訪完後輸出該邊
   2) 否則離開當前結點
3. 若還有節點尚未拜訪，則無解
4. 否則輸出順序即為一組解

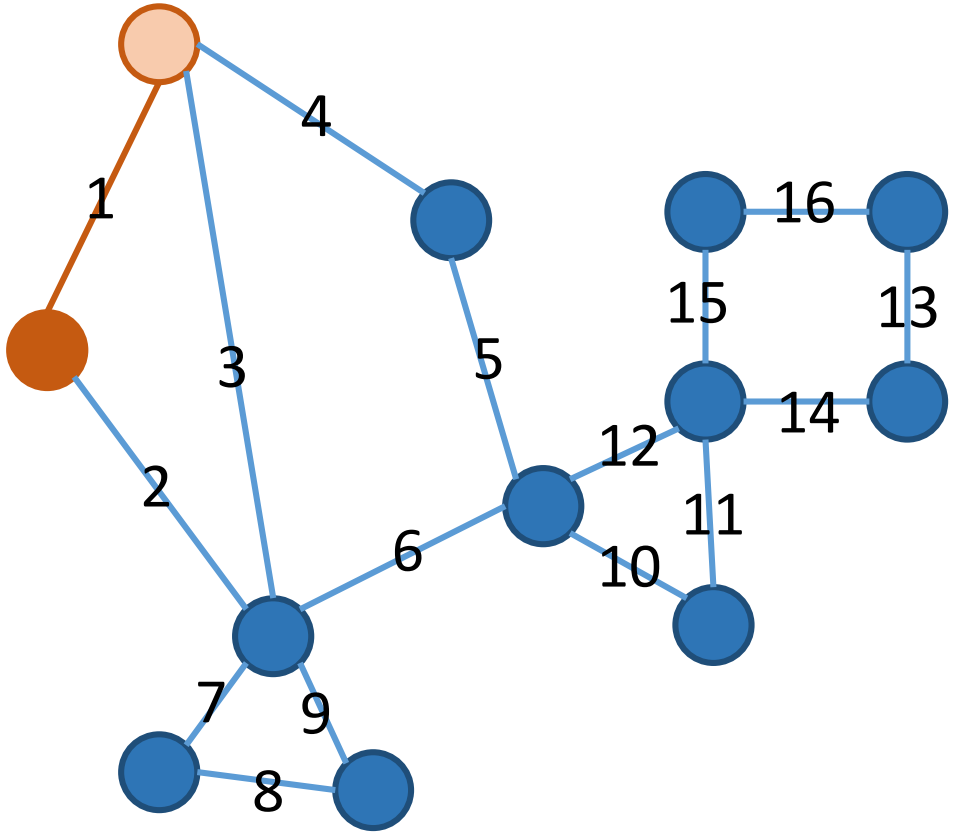
Show time!





Sprout

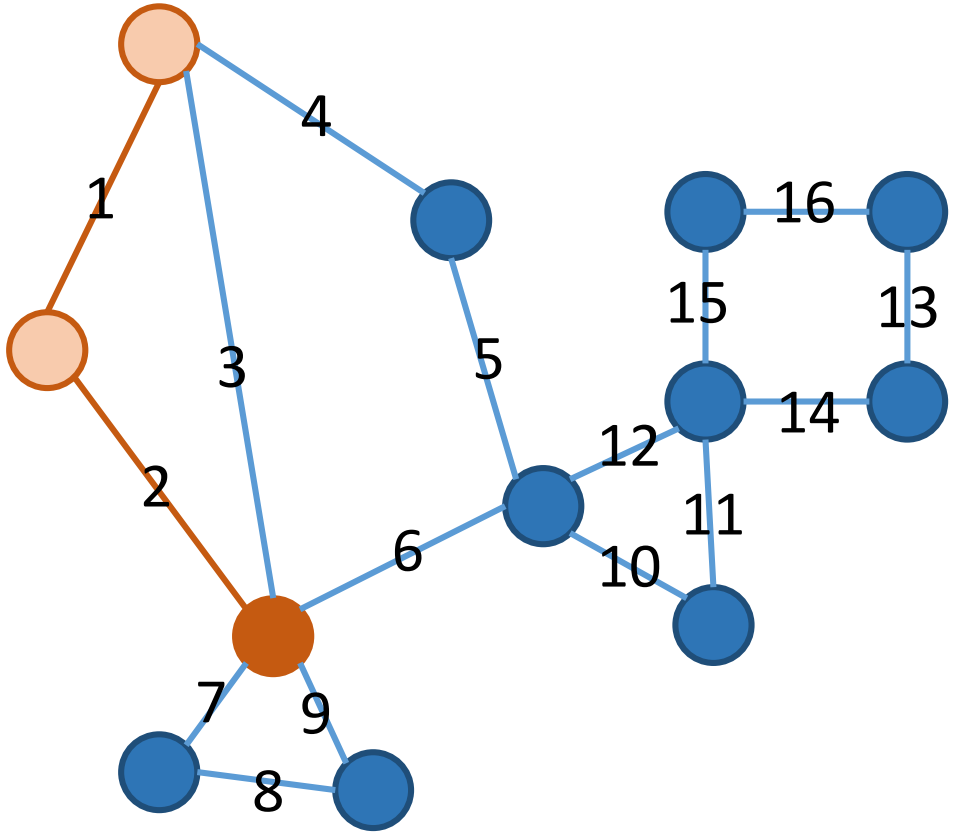
Show time!



1							

Sprout

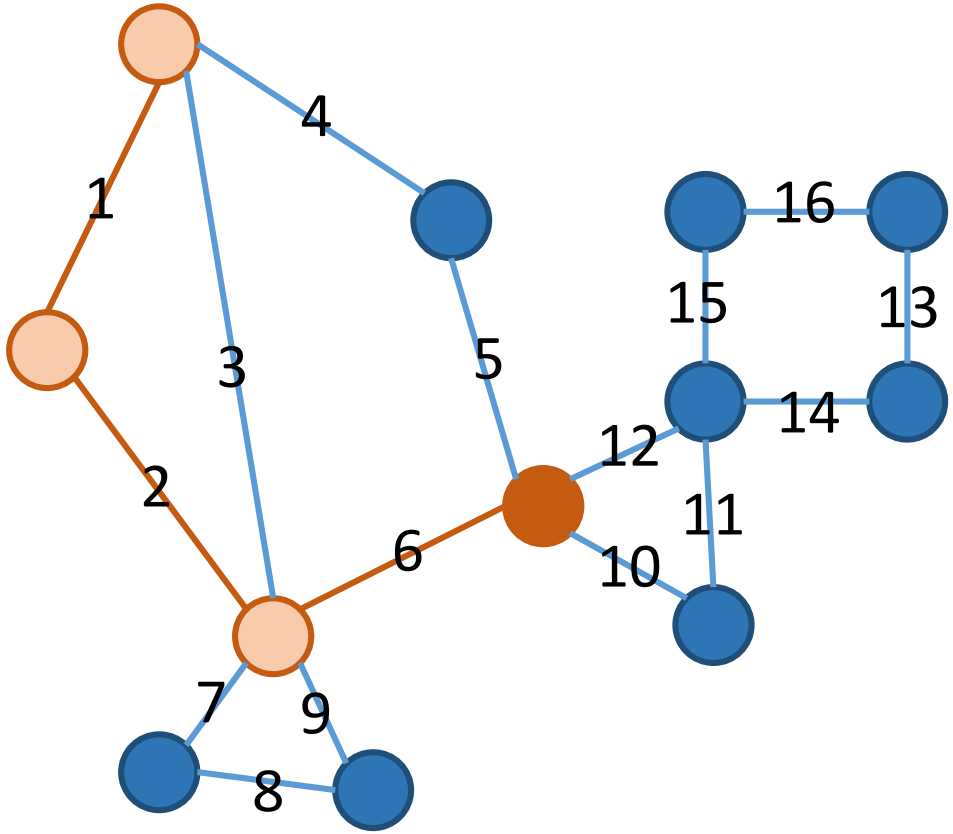
Show time!



1	2						

Sprout

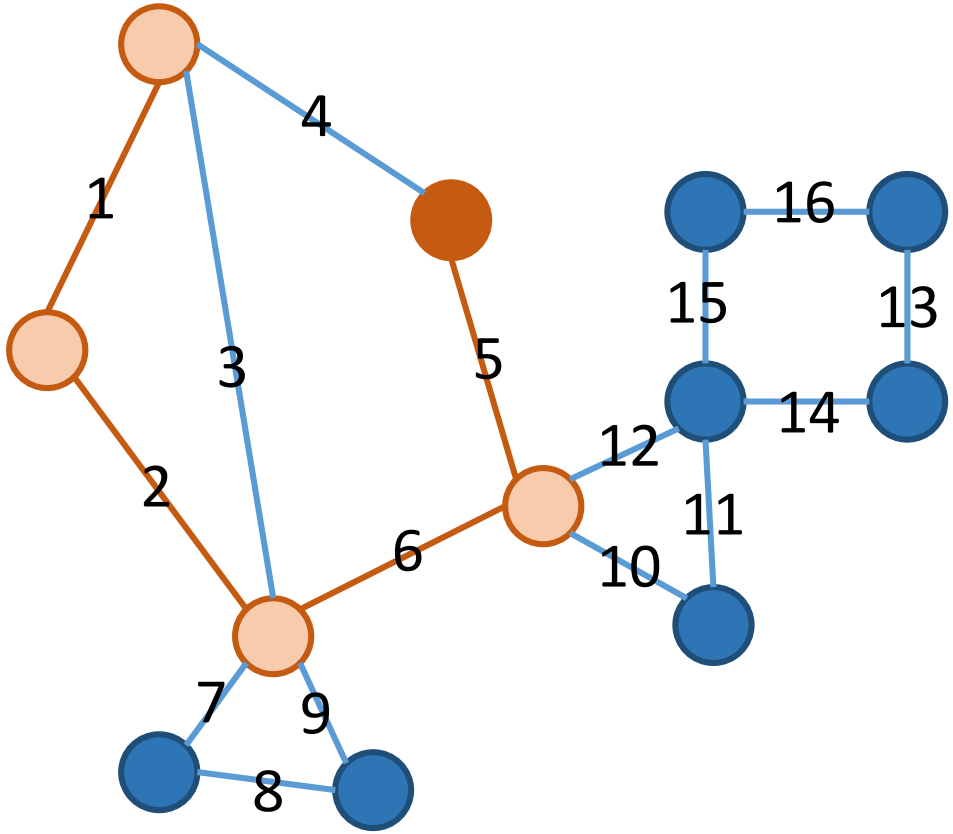
Show time!



1	2	6					

Sprout

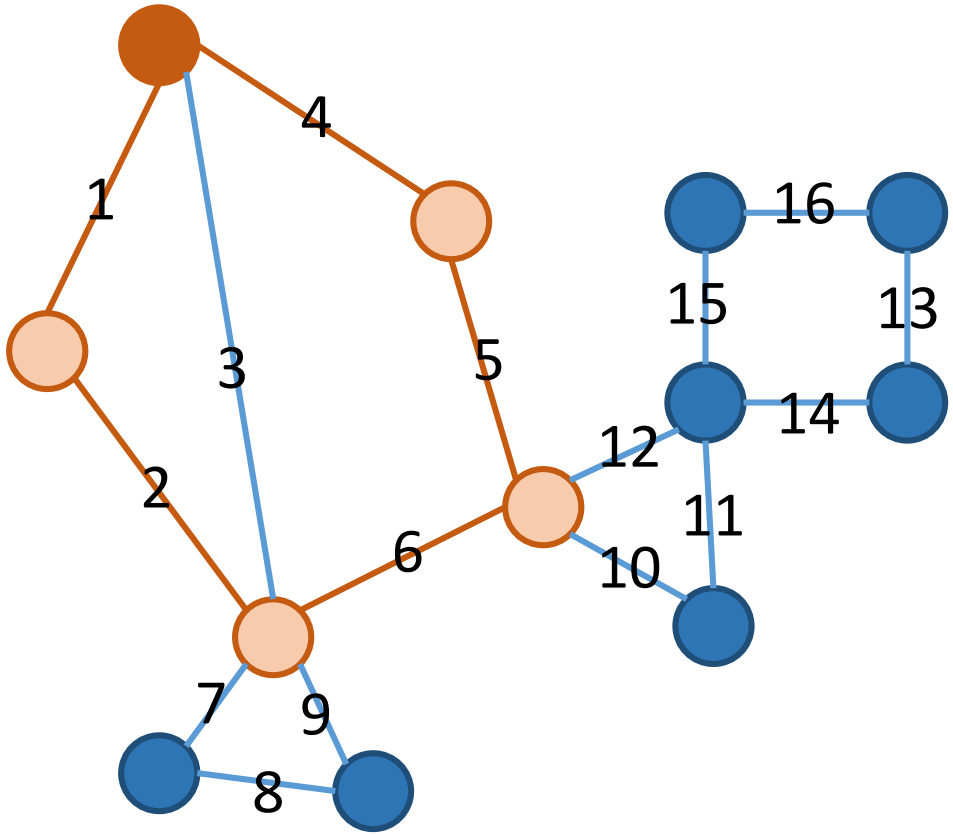
Show time!



1	2	6	5				

Sprout

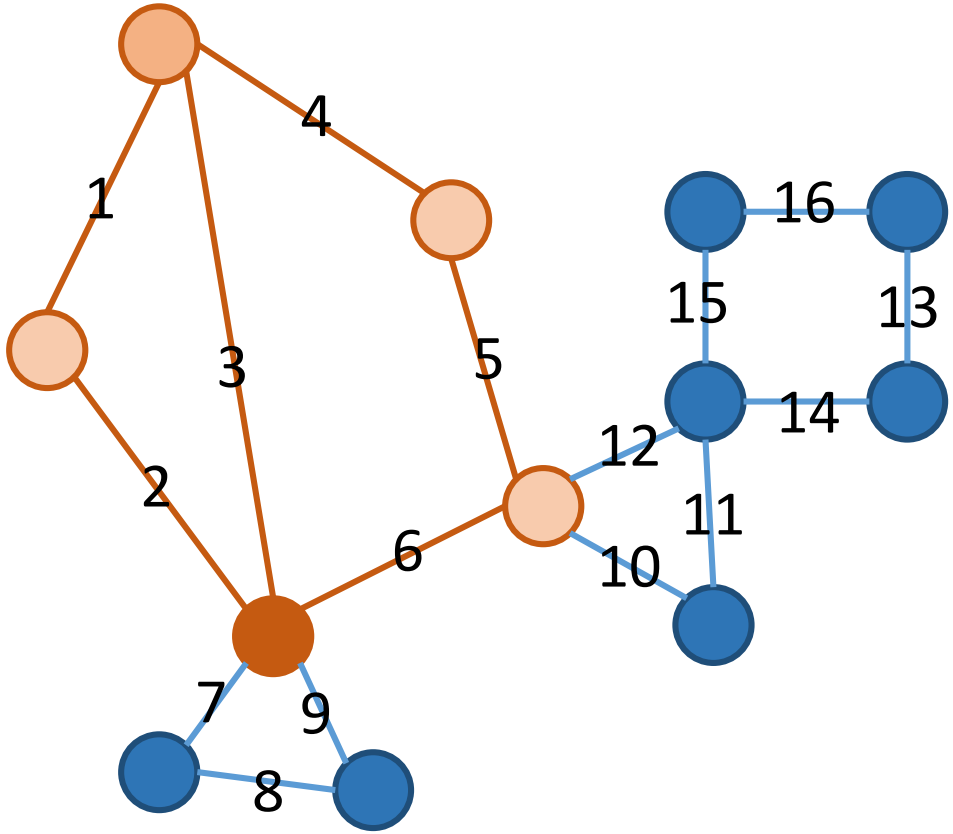
Show time!



1	2	6	5	4			

Sprout

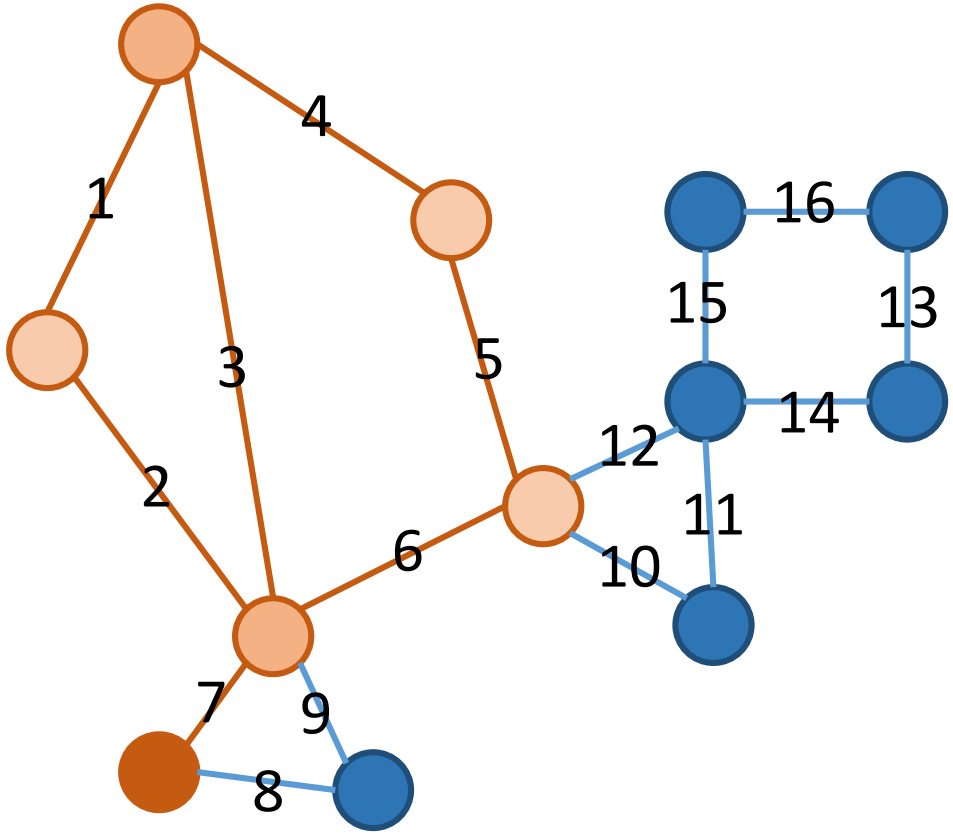
Show time!



1	2	6	5	4	3		

Sprout

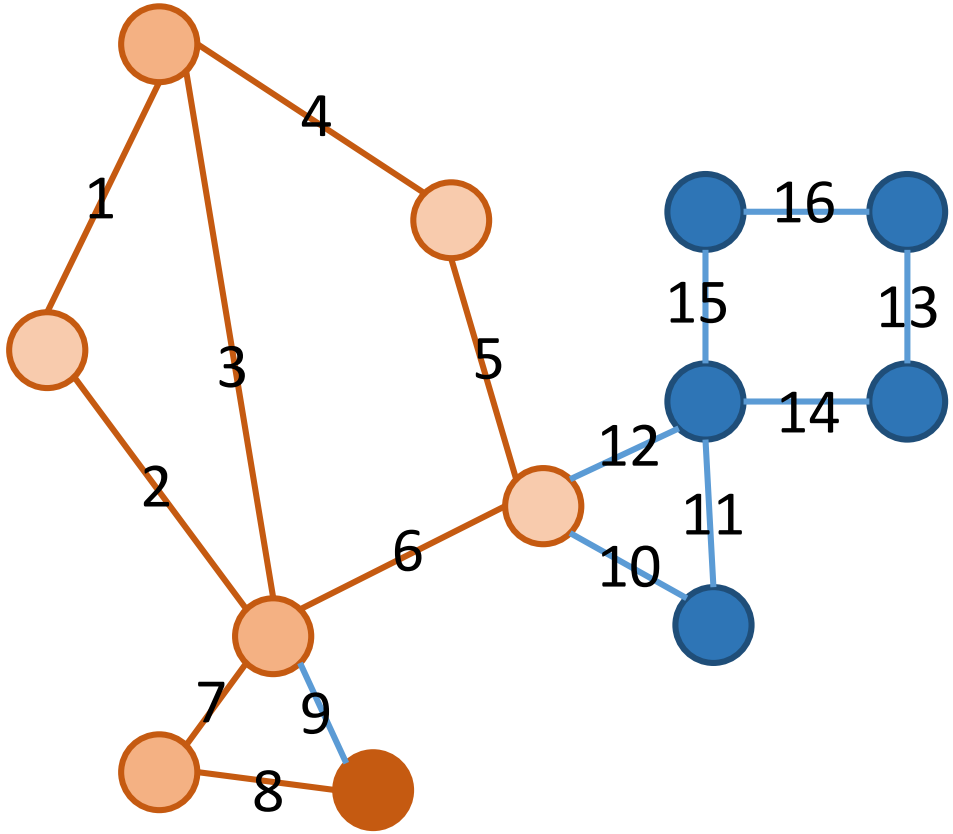
Show time!



1	2	6	5	4	3	7	

Sprout

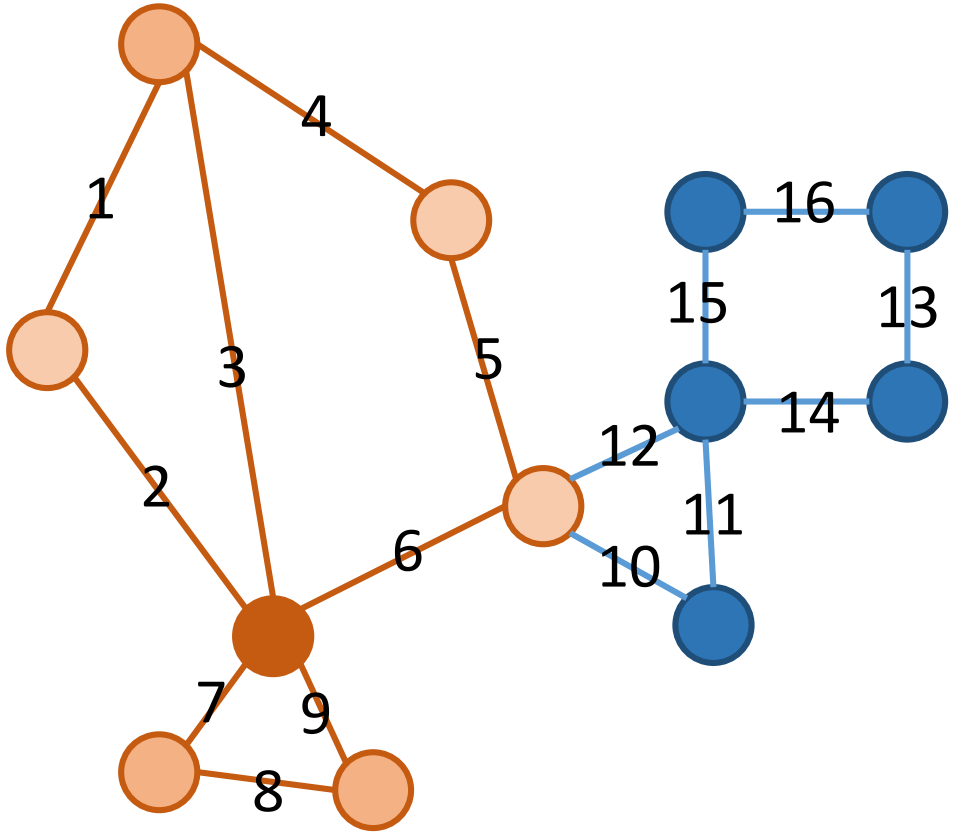
Show time!



1	2	6	5	4	3	7	8

Sprout

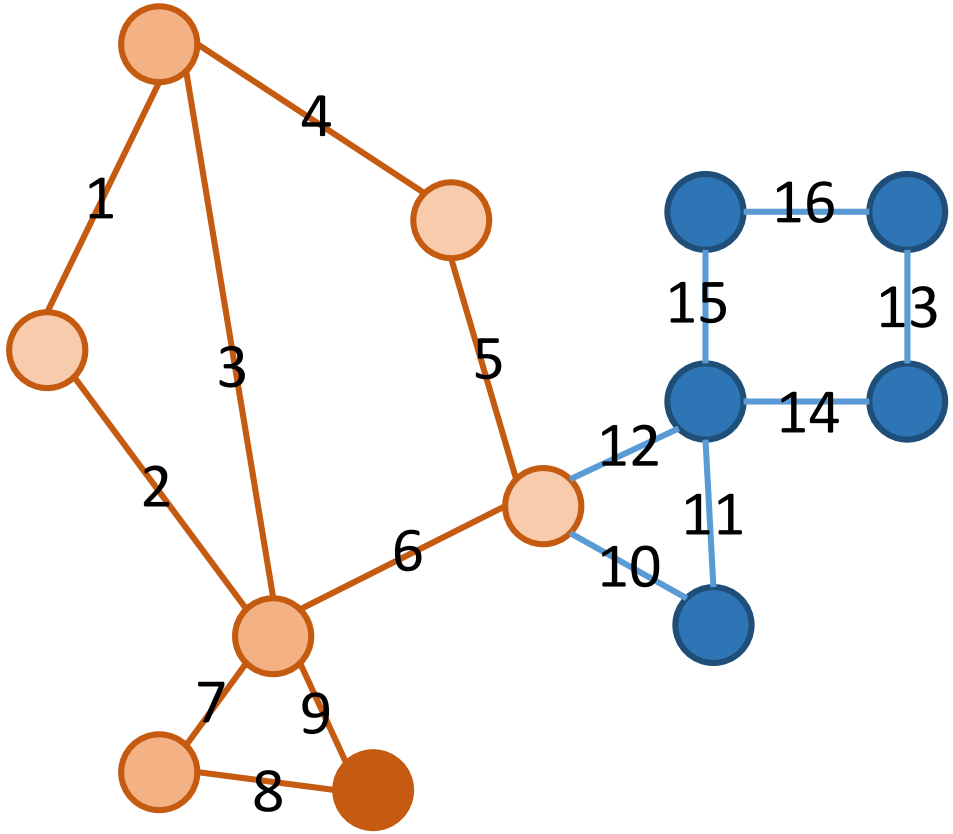
Show time!



1	2	6	5	4	3	7	8
9							

Sprout

Show time!

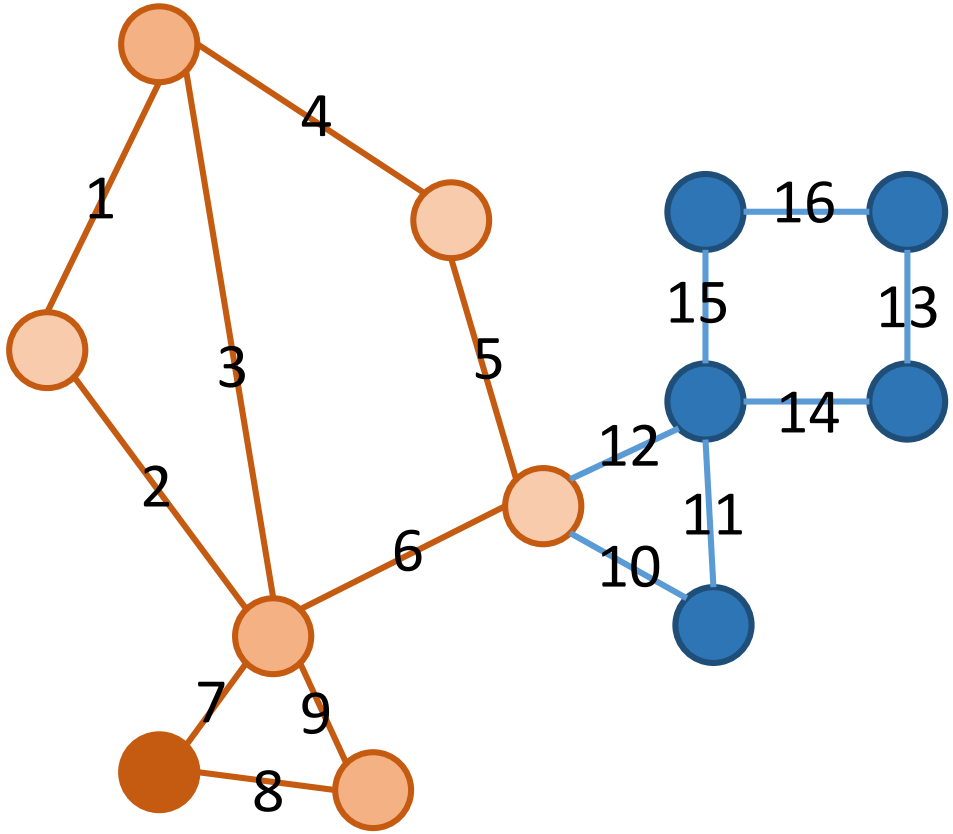


1	2	6	5	4	3	7	8

9							

Sprout

Show time!

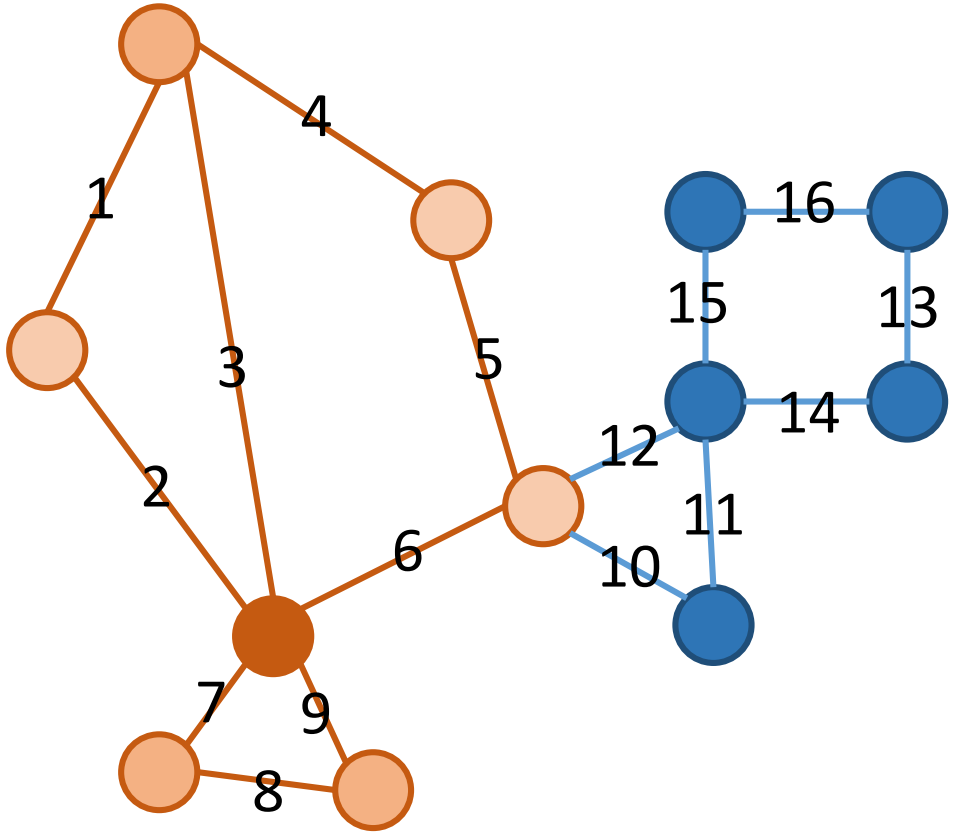


1	2	6	5	4	3	7	

9	8						

Sprout

Show time!

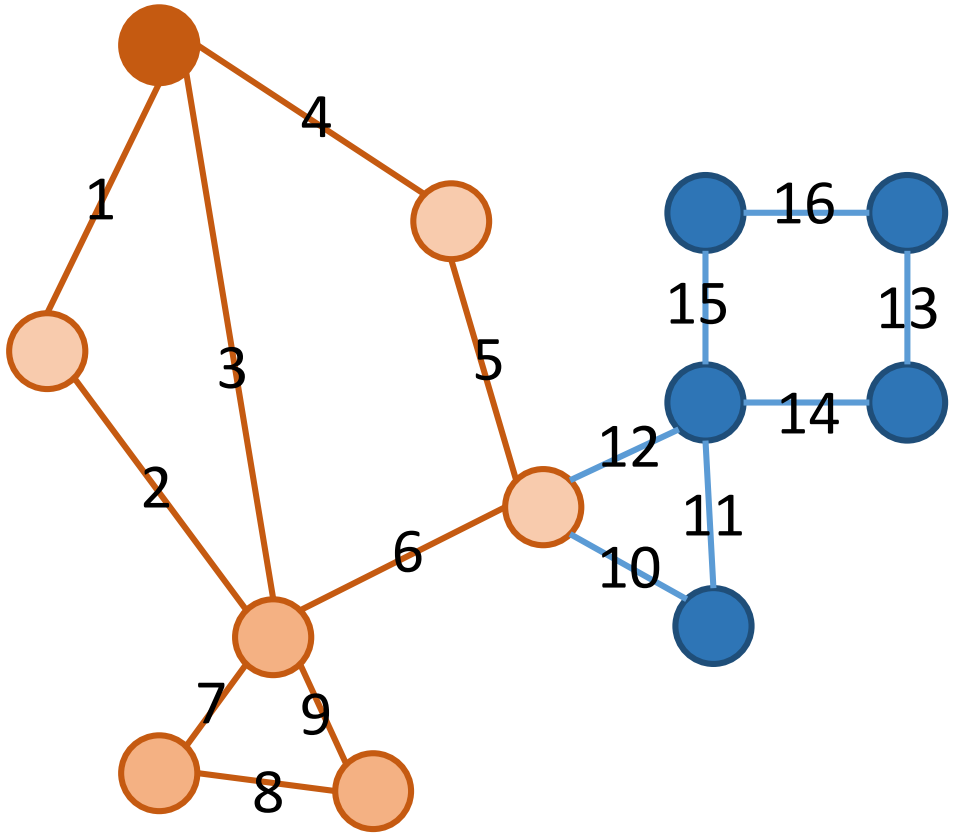


1	2	6	5	4	3		

9	8	7					

Sprout

Show time!

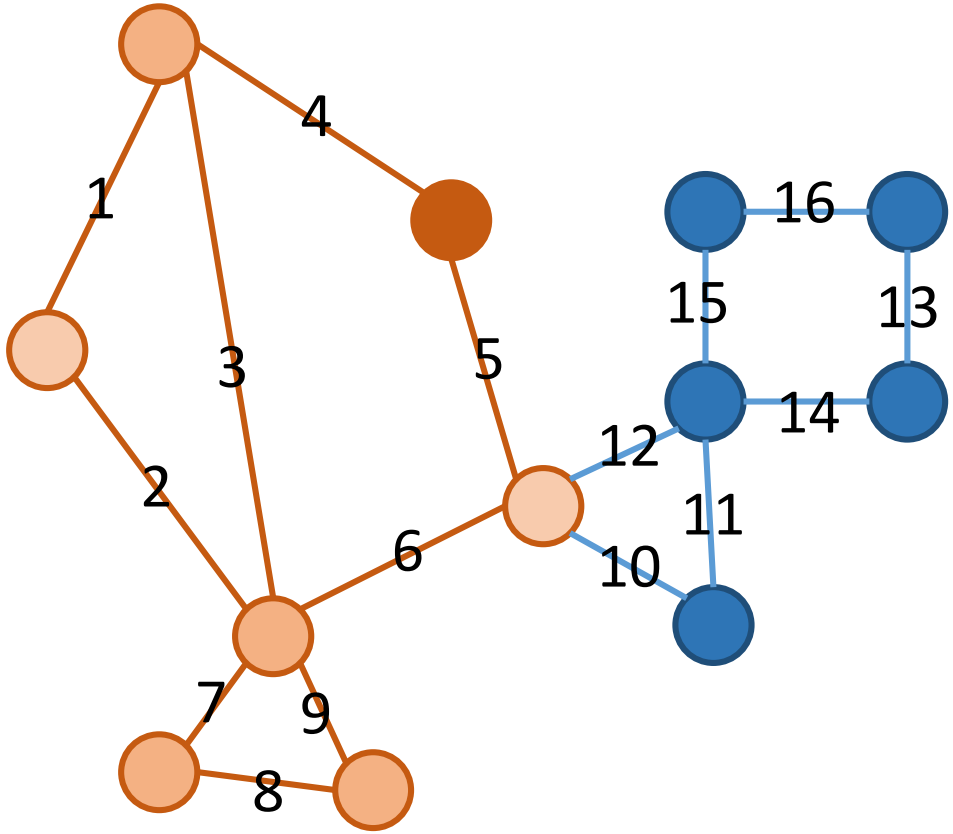


1	2	6	5	4			

9	8	7	3				

Sprout

Show time!

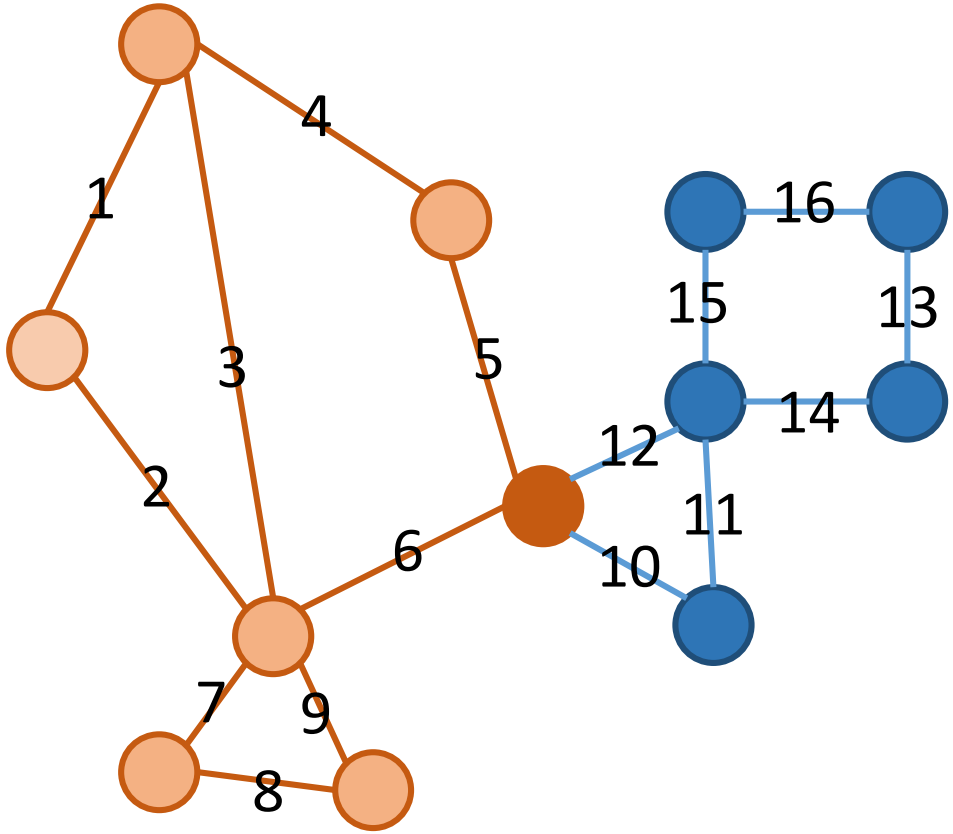


1	2	6	5				

9	8	7	3	4			

Sprout

Show time!

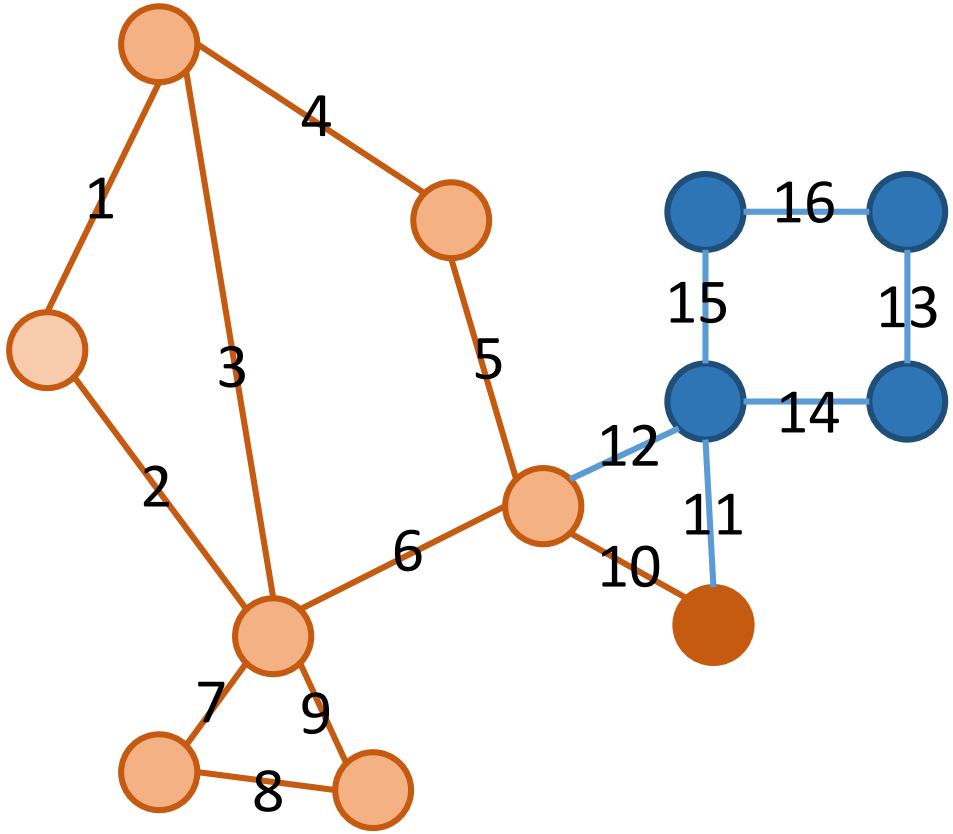


1	2	6					

9	8	7	3	4	5		

Sprout

Show time!

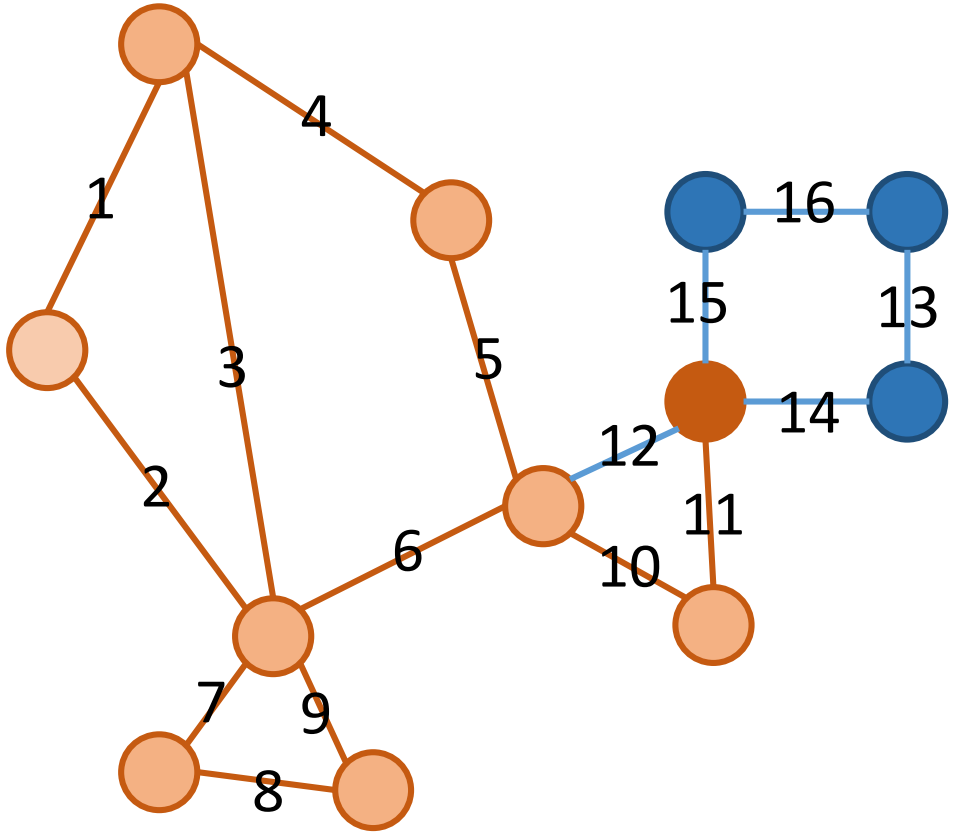


1	2	6	10				

9	8	7	3	4	5		

Sprout

Show time!

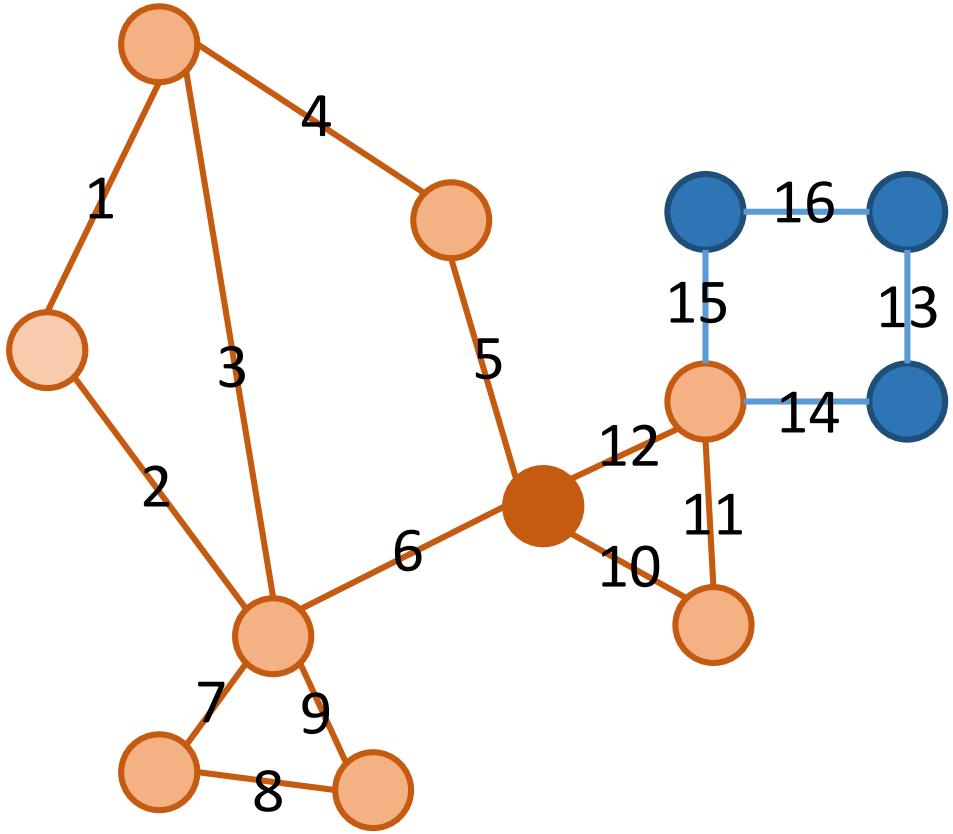


1	2	6	10	11			

9	8	7	3	4	5		

Sprout

Show time!

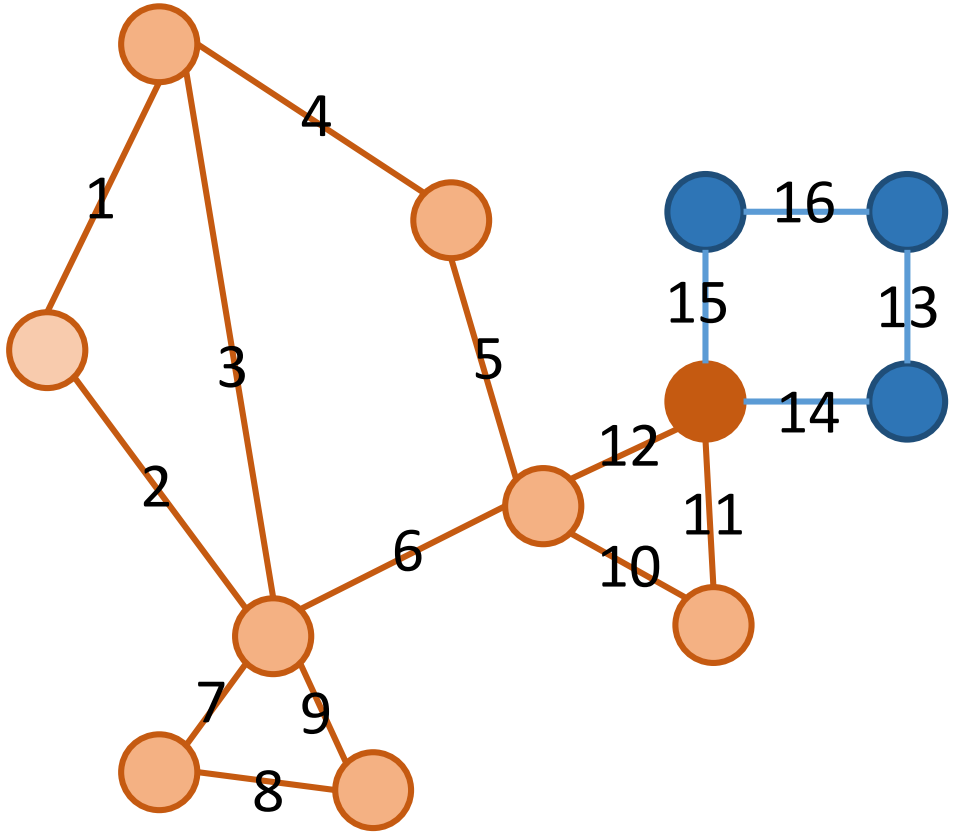


1	2	6	10	11	12		

9	8	7	3	4	5		

Sprout

Show time!

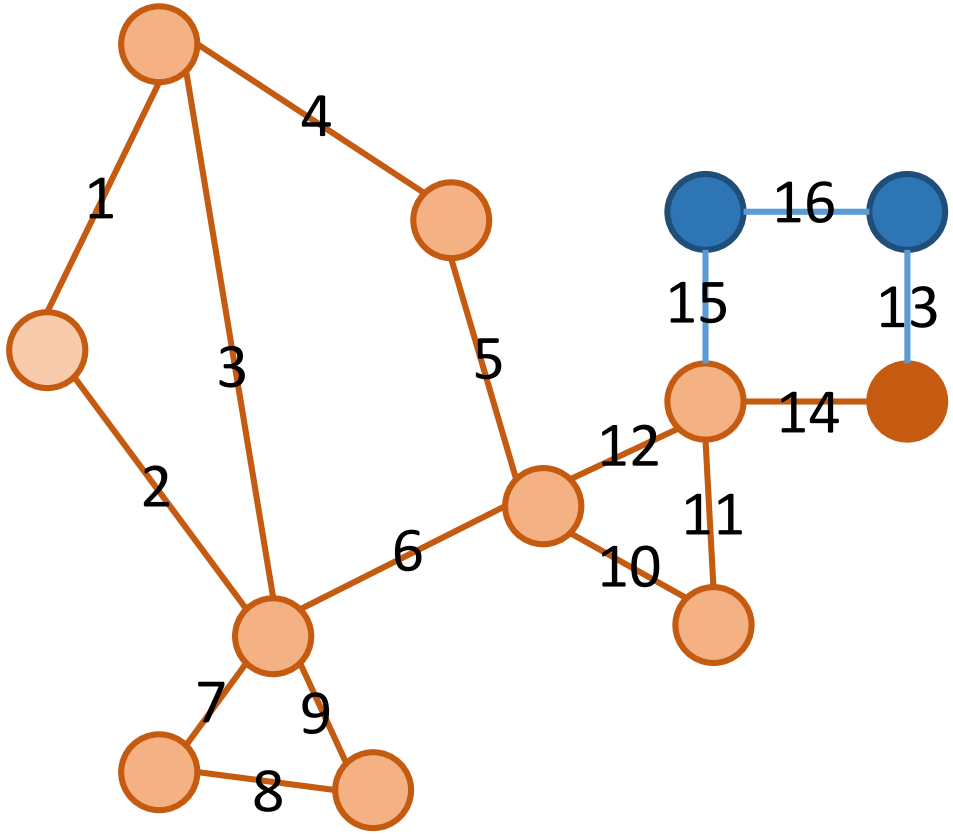


1	2	6	10	11			

9	8	7	3	4	5	12	

Sprout

Show time!

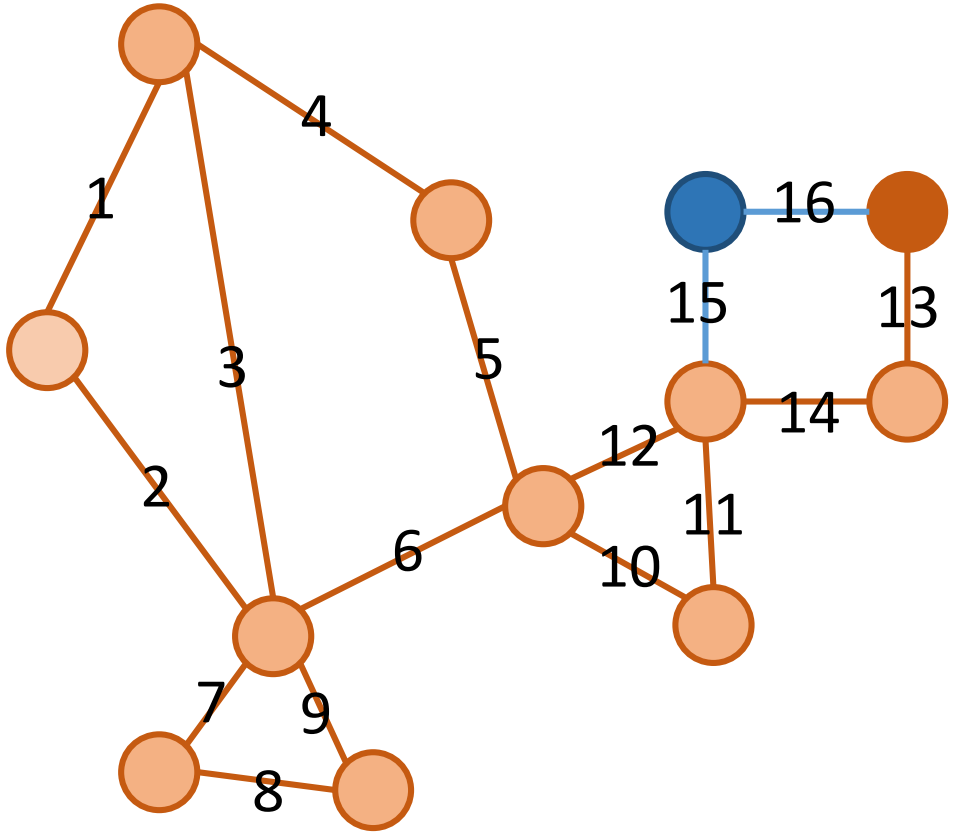


1	2	6	10	11	14		

9	8	7	3	4	5	12	

Sprout

Show time!

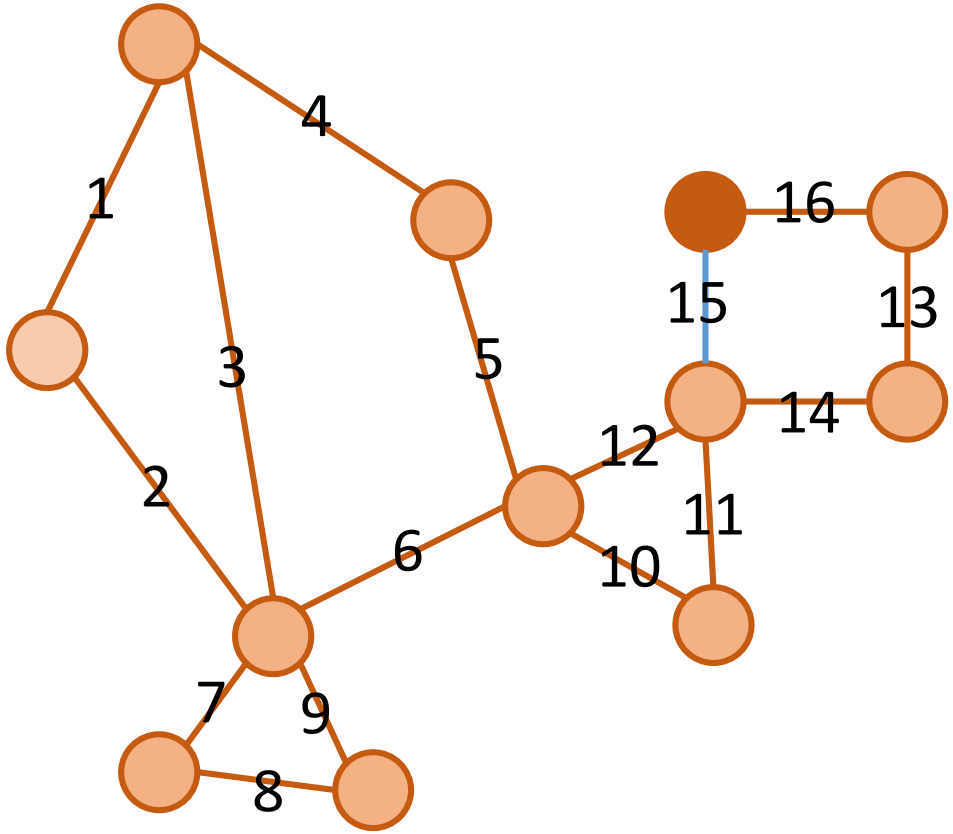


1	2	6	10	11	14	13	

9	8	7	3	4	5	12	

Sprout

Show time!

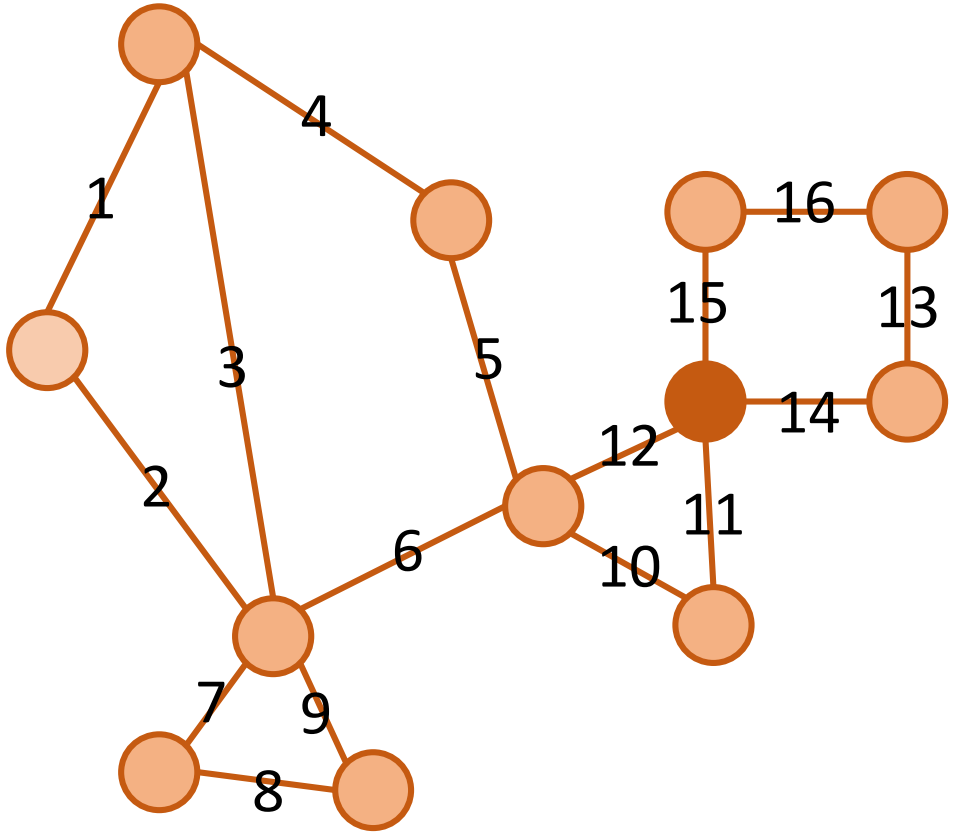


1	2	6	10	11	14	13	16

9	8	7	3	4	5	12	

Sprout

Show time!

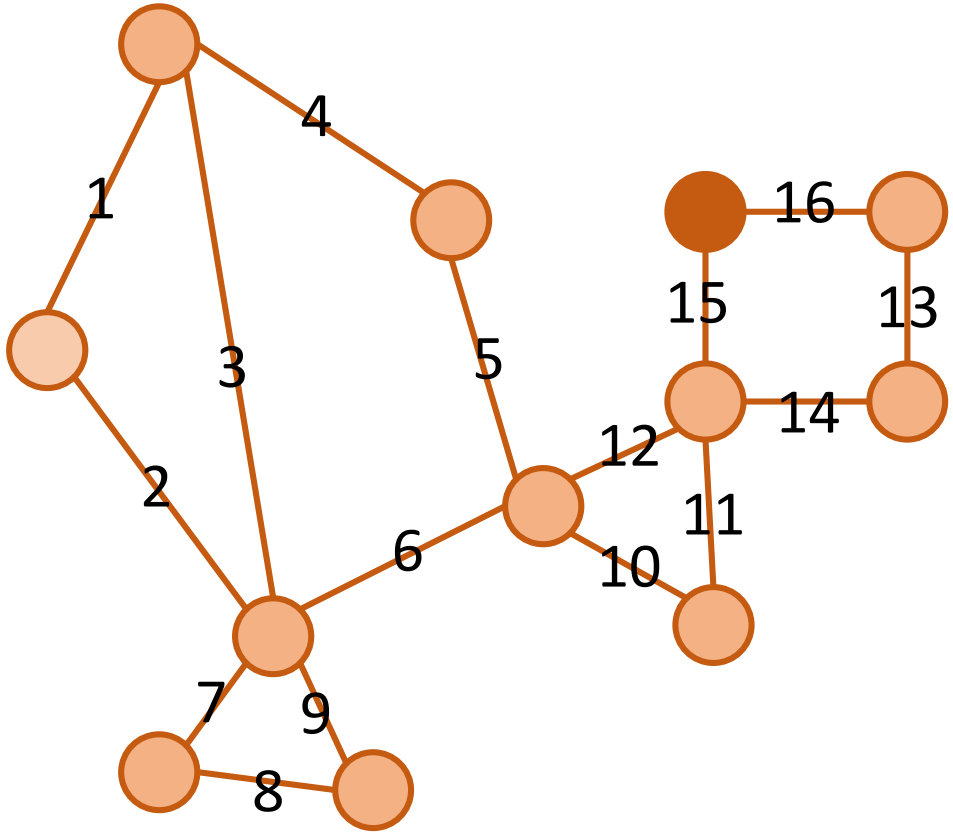


1	2	6	10	11	14	13	16
15							

9	8	7	3	4	5	12	

Sprout

Show time!

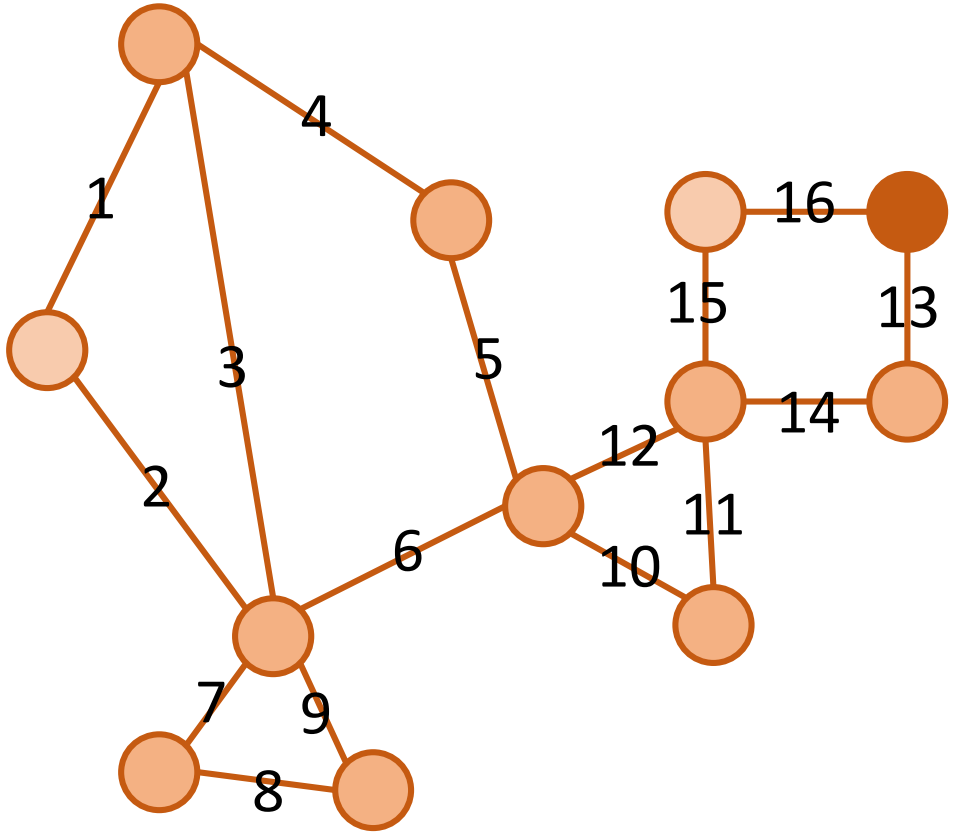


1	2	6	10	11	14	13	16

9	8	7	3	4	5	12	15

Sprout

Show time!

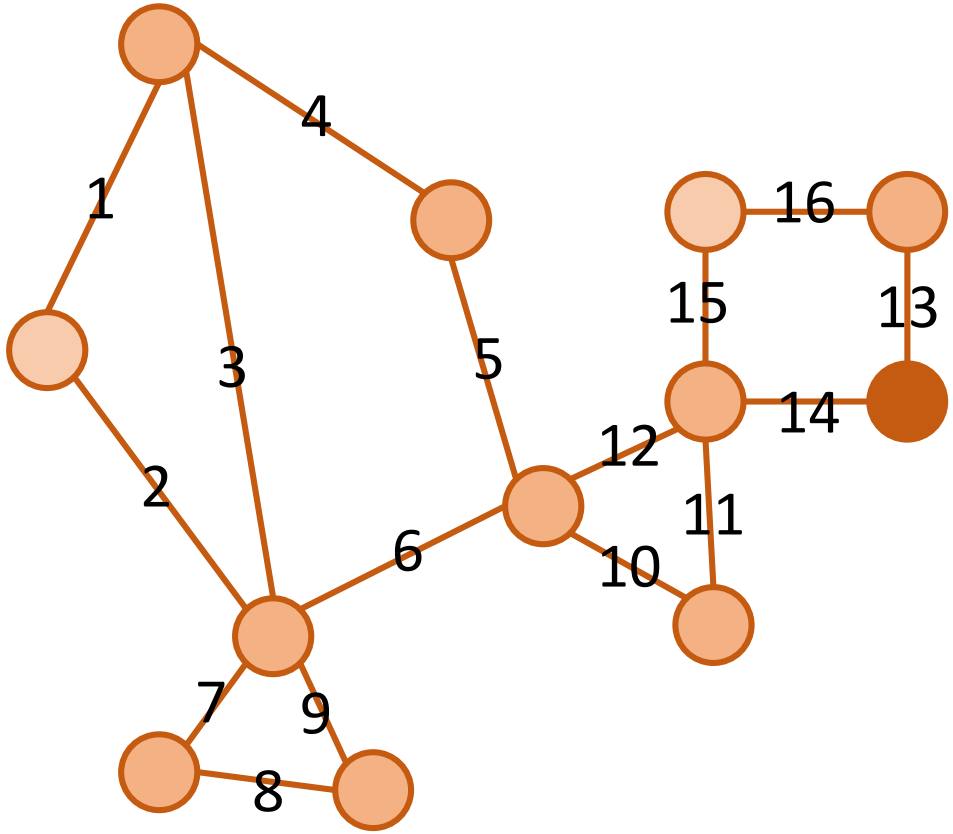


1	2	6	10	11	14	13	

9	8	7	3	4	5	12	15
16							

Sprout

Show time!

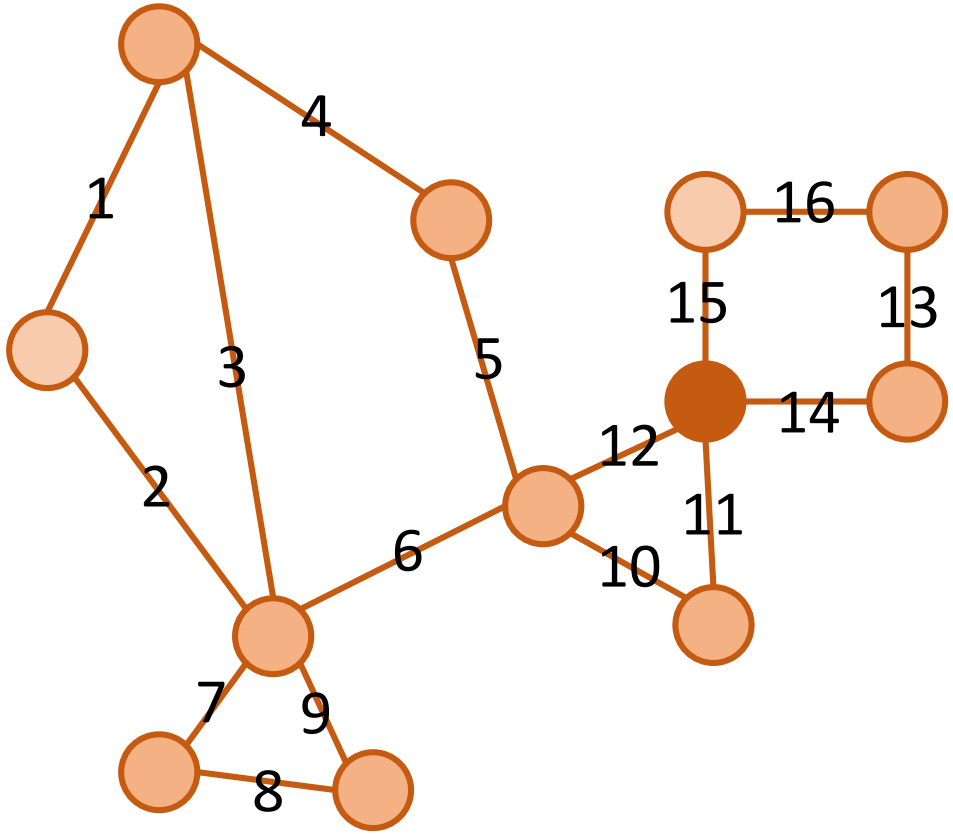


1	2	6	10	11	14		

9	8	7	3	4	5	12	15
16	13						

Sprout

Show time!

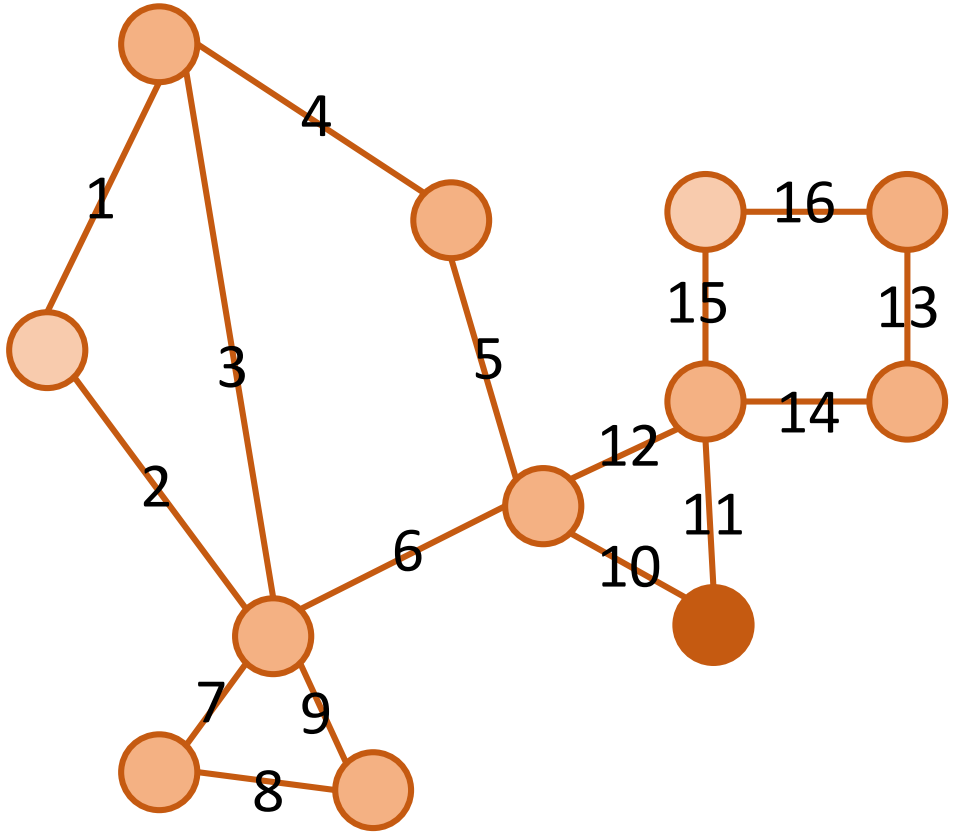


1	2	6	10	11			

9	8	7	3	4	5	12	15
16	13	14					

Sprout

Show time!

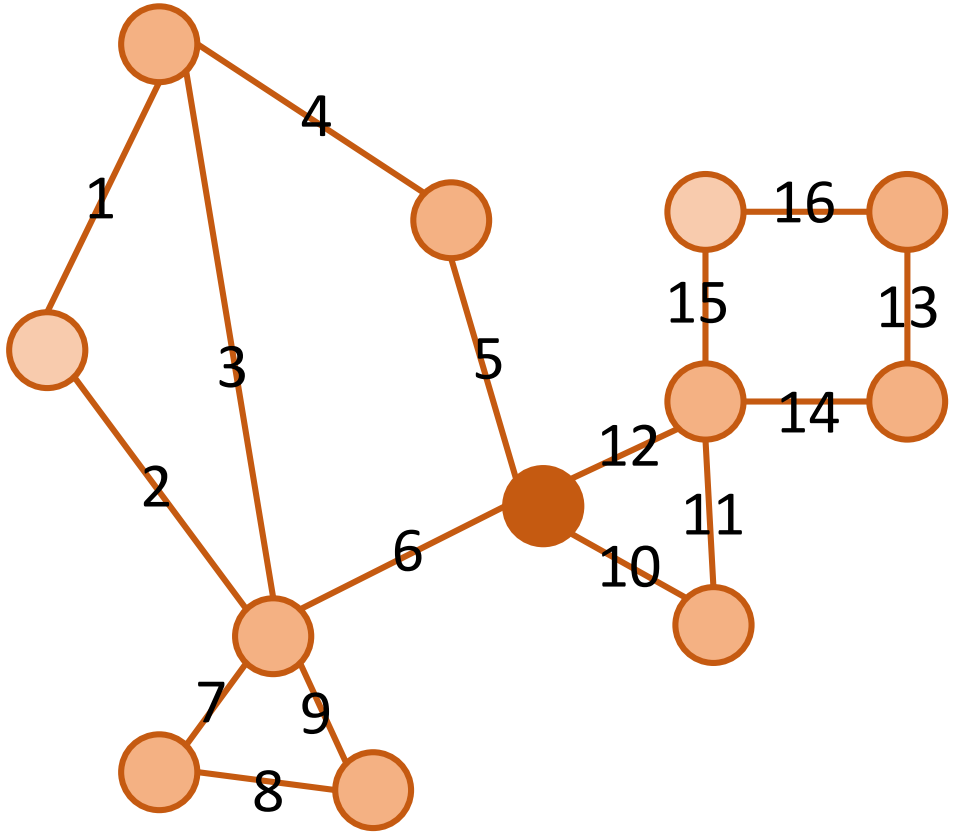


1	2	6	10				

9	8	7	3	4	5	12	15
16	13	14	11				

Sprout

Show time!

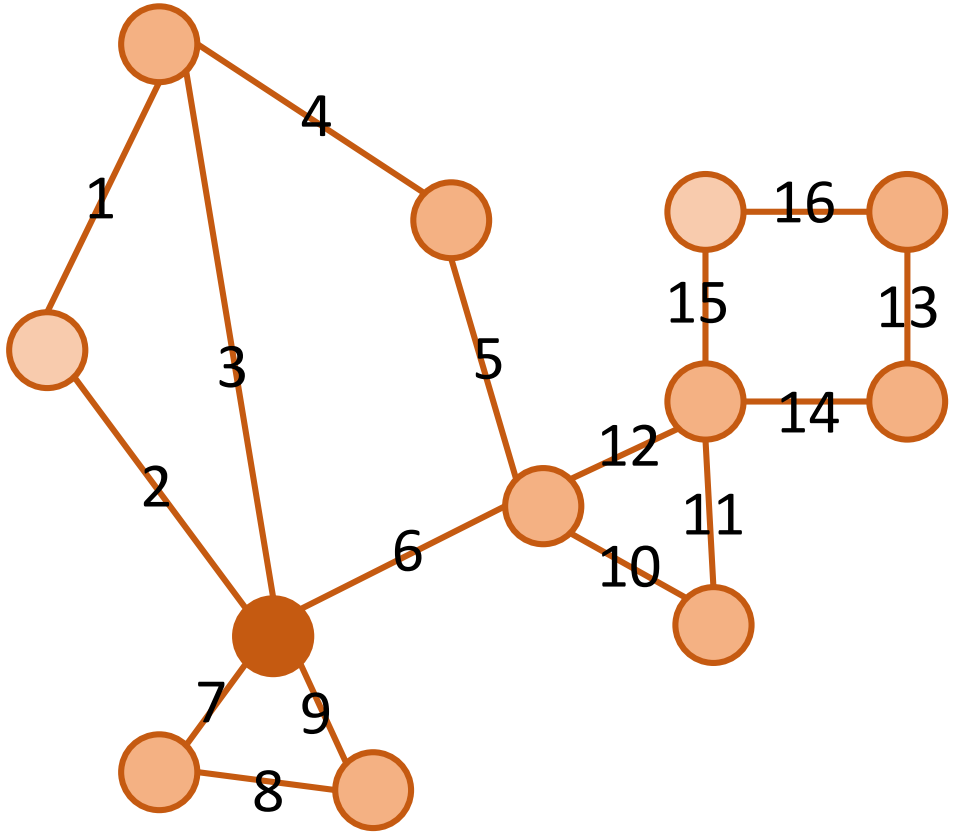


1	2	6					

9	8	7	3	4	5	12	15
16	13	14	11	10			

Sprout

Show time!

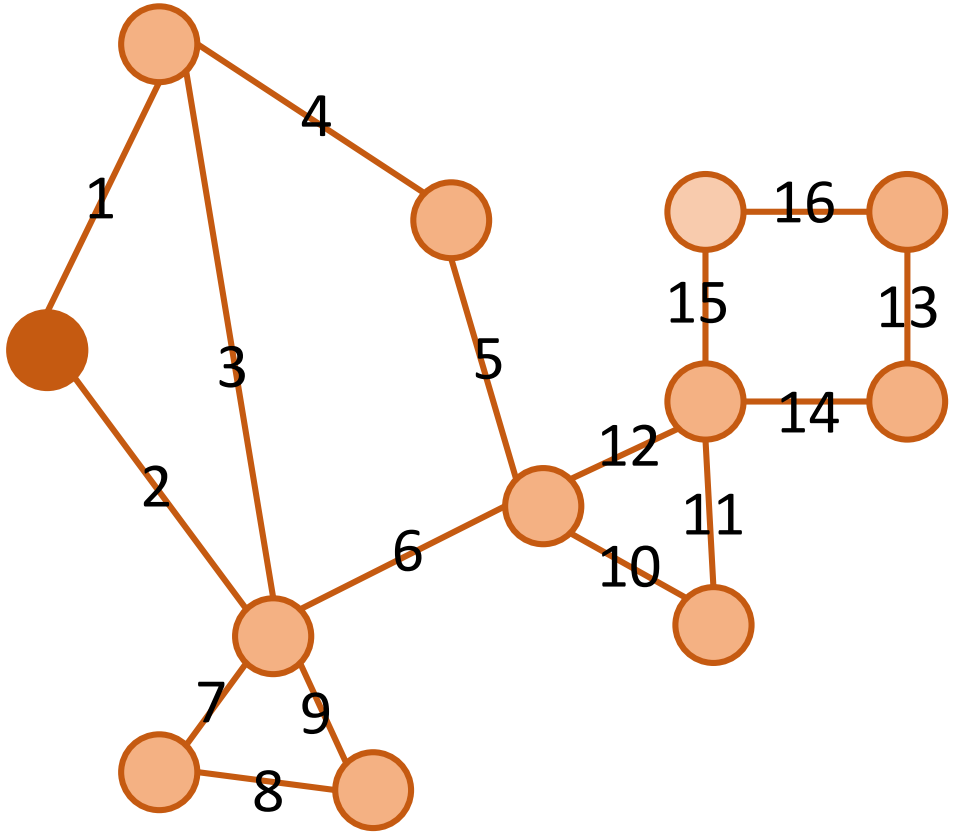


1	2						

9	8	7	3	4	5	12	15
16	13	14	11	10	6		

Sprout

Show time!

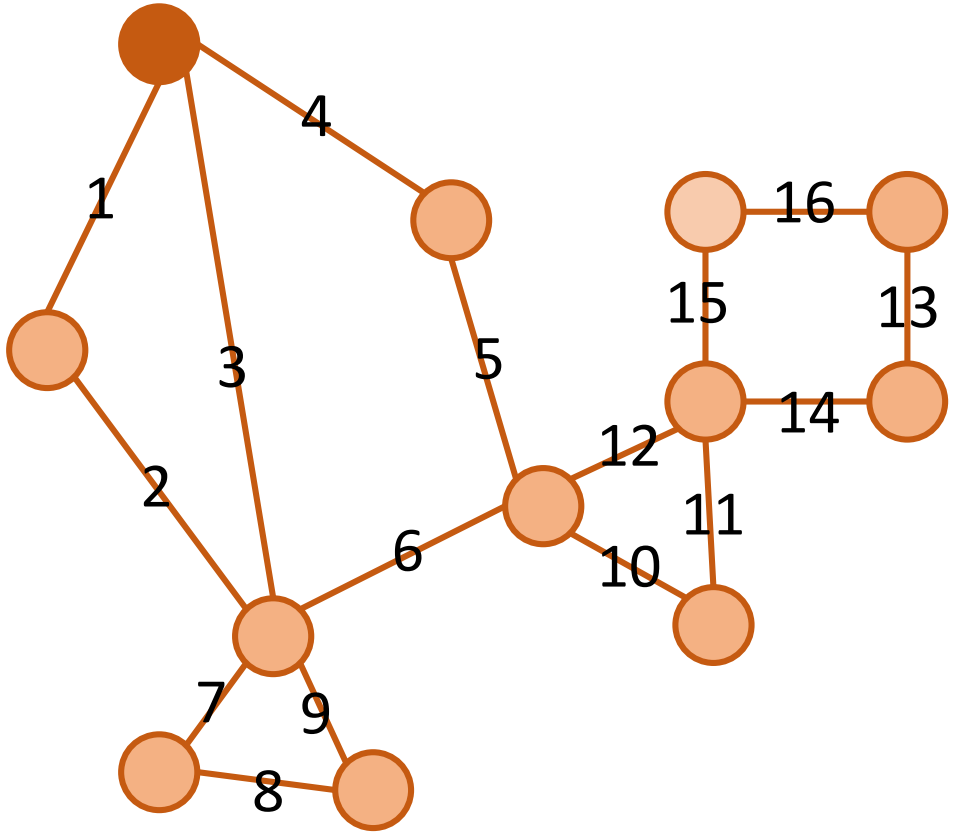


1							

9	8	7	3	4	5	12	15
16	13	14	11	10	6	2	

Sprout

Show time!



9	8	7	3	4	5	12	15
16	13	14	11	10	6	2	1

Sprout

有向圖的話……

- 在無向圖中，我們事實上是根據我們的需求自己幫邊定向
- 有向圖中，如果先天條件不良，我們就無能為力了
  - 入點：出度數 = 入度數 – 1
  - 出點：出度數 = 入度數 + 1
  - 平衡點：出度數 = 入度數
  - 不能有三種點以外的點
  - 入點、出點都最多只能有一個
  - 其餘點皆須為平衡點

有向圖的話……

- 如果整張圖連通（從起點開始可以走到任意一點），那麼解的存在性證明與無向圖完全相似
- 不同的是，在無向圖的算法中，輸出的順序之所以可以成立，來自於「任意無向圖上的路徑逆序輸出依然合法」
- 有向圖中，把輸出逆序後才是一組合法的解
- 複雜度：皆為 $O(n+m)$

哈密頓問題

- 目標為點的一筆畫問題！

- 給定一張圖（有向或無向），是否存在一條路徑使得所有節點恰好走過一次呢？
  - 如果最後不用回到起點，則稱之為 *Hamilton path*
  - 否則稱之為 *Hamilton circuit*
  - 若一張圖中存在 *Hamilton circuit*，則稱此圖為 *Hamilton graph*

- 圖上的點和邊關係密切，歐拉路問題有這麼好的解，是否可以套用到哈密頓問題上呢？

Terrible

- 很遺憾地，不論是有向圖還是無向圖，哈密頓問題都已經被證明是 NP-complete

- 目前僅知一個充分條件：
  - 設 $G = \{V, E\}$ 是一個無向簡單圖，$|V| = n \geq 3$。若對於任意的兩個頂點 $u, v \in V$ 都有 $d(u) + d(v) \geq n$，則 G 是 Hamilton graph
    (proved by Øystein Ore)
- 但這並不代表我們只能夠用 $O(n * n!)$ 暴力求解

Dynamic Programming!

- 反正每個點走過之後就不會走第二次，對應著邊也不會有走兩次的情形
- 只有「已經走過的點」和「當前走過的點」是重要的事情！
- 定義狀態 $dp[i][S]$ 為「是否有可能當前在點 i，其中 S 這個集合已經拜訪過」

Dynamic Programming!

- 初始時 $dp[i][\{i\}] = 1\ \forall i$，其餘皆為 0
- 對於一個狀態 $dp[i][S]$，如果 $(i,j) \in E$ 且 $j \notin S$，那麼我們可以從 i 走到 j，轉移到 $dp[j][S \cup \{j\}]$
- 其中 S 通常以狀態壓縮實作
  - ex. $n = 10, S = \{1,3,5,6\}$，則可以用 0000110101 表示
- 若使用 adjacent matrix 存邊方式，則每個狀態只需要 $O(n)$ 轉移
  - 反正 n 沒辦法太大，不用考慮空間 $O(n^2)$ 的問題 :p
- 時間複雜度為 $O(n^2 * 2^n)$