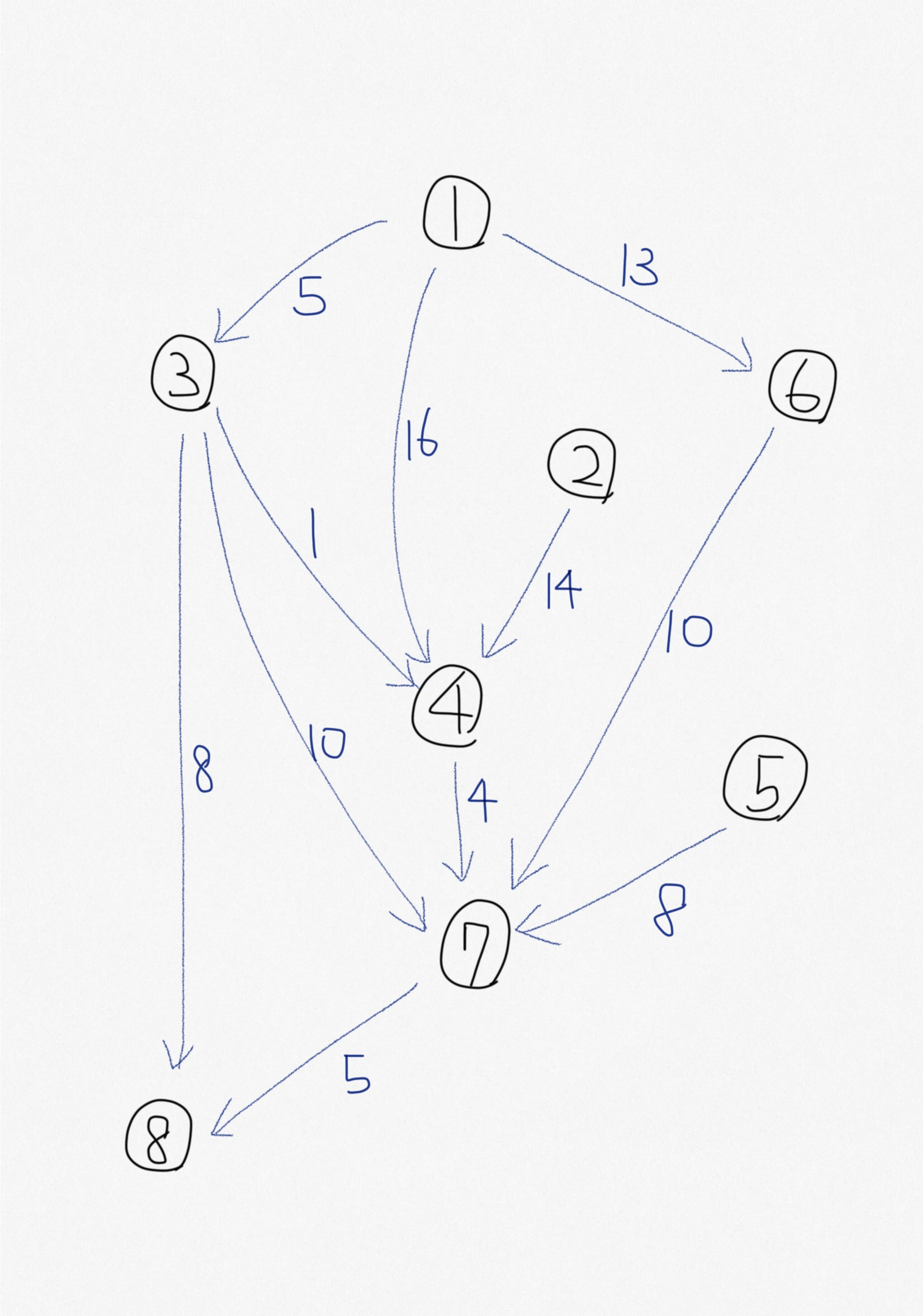
Algorithms HW7 Handwritten

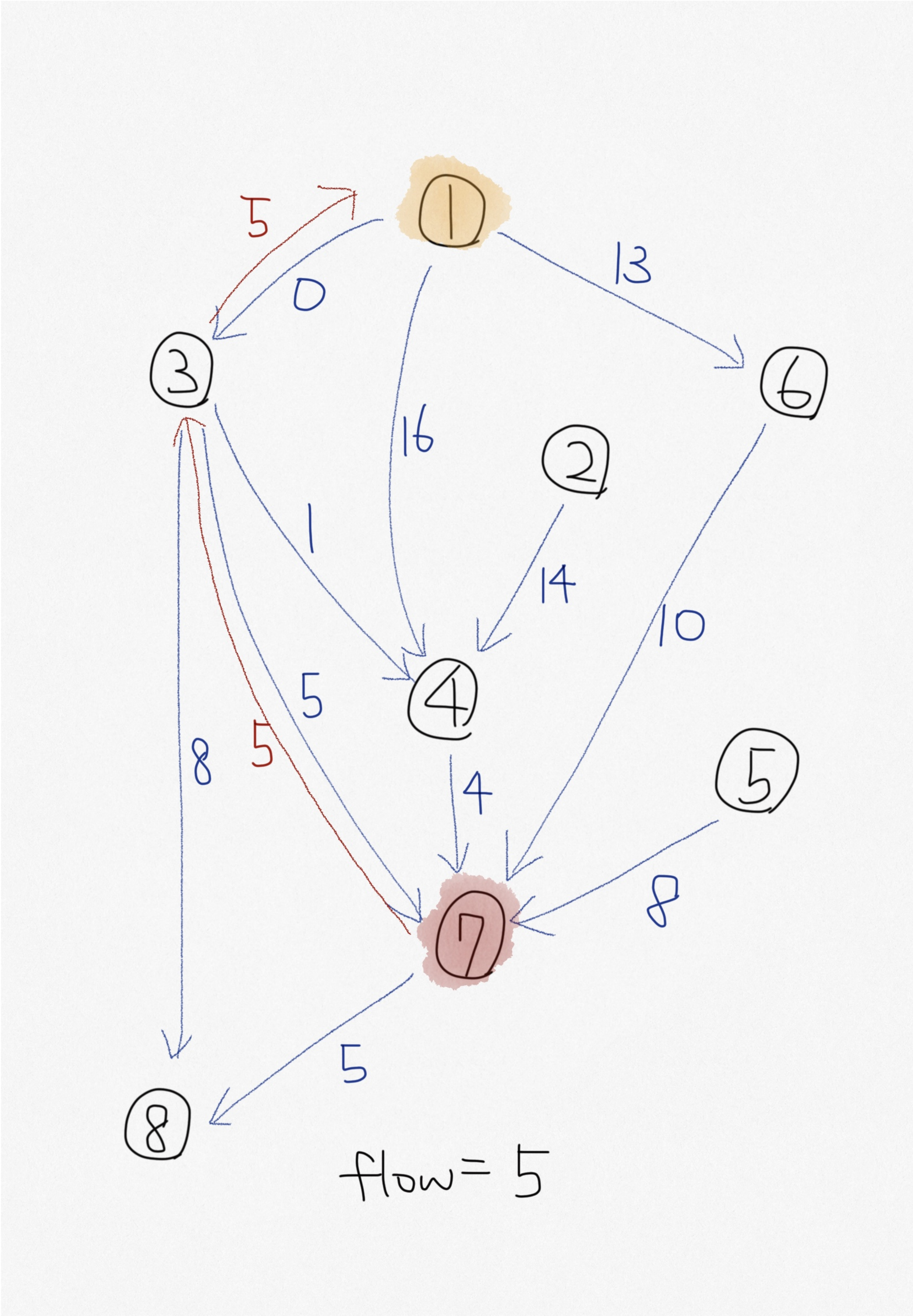
2020 Spring | 90899201Y tony20715 黃悟淳

Problem 1

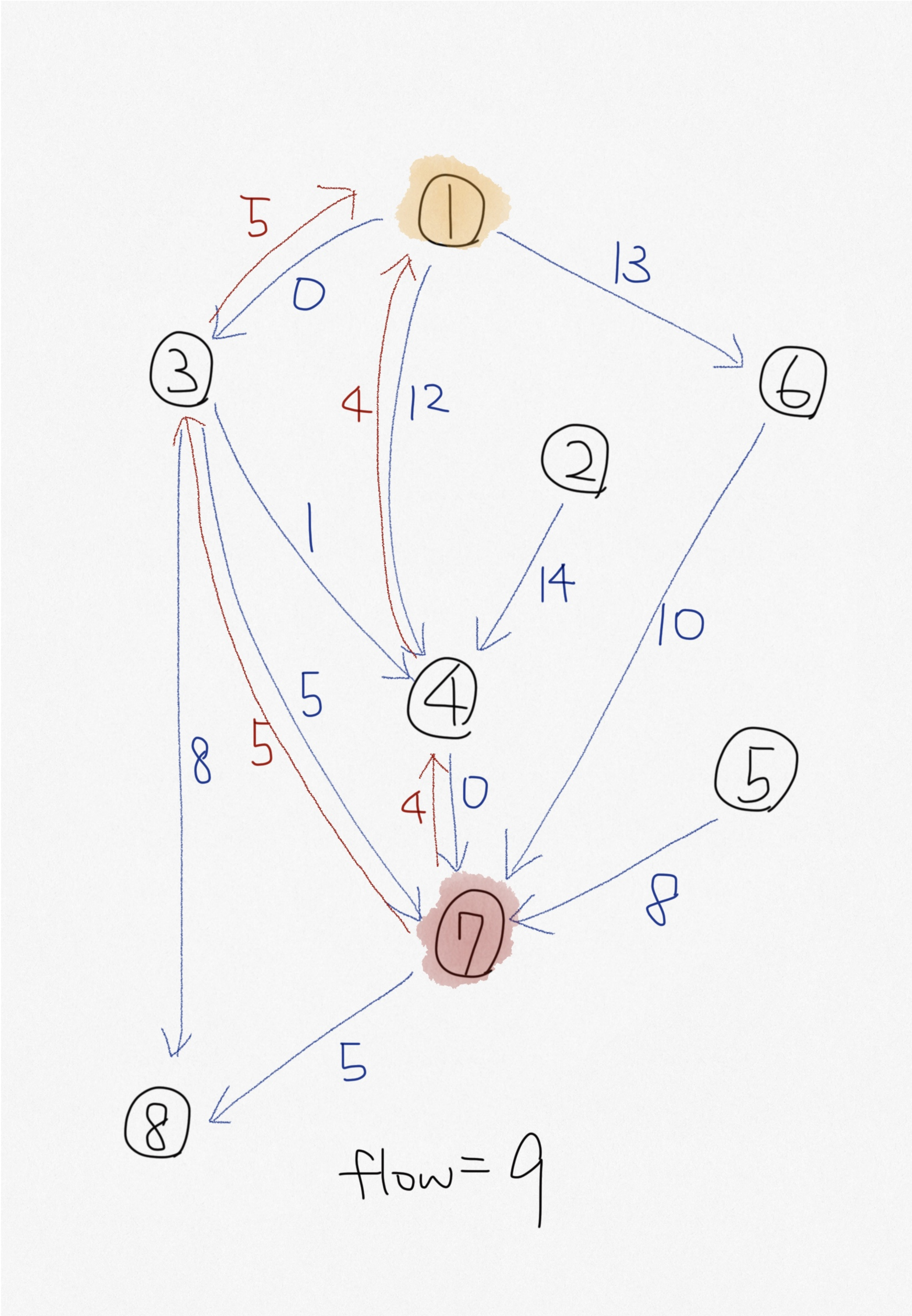
1. - 首先，把相同source、destination的多個邊，權相加整合成一個邊。此圖即為最初的Residual Network，此時流量flow=0。



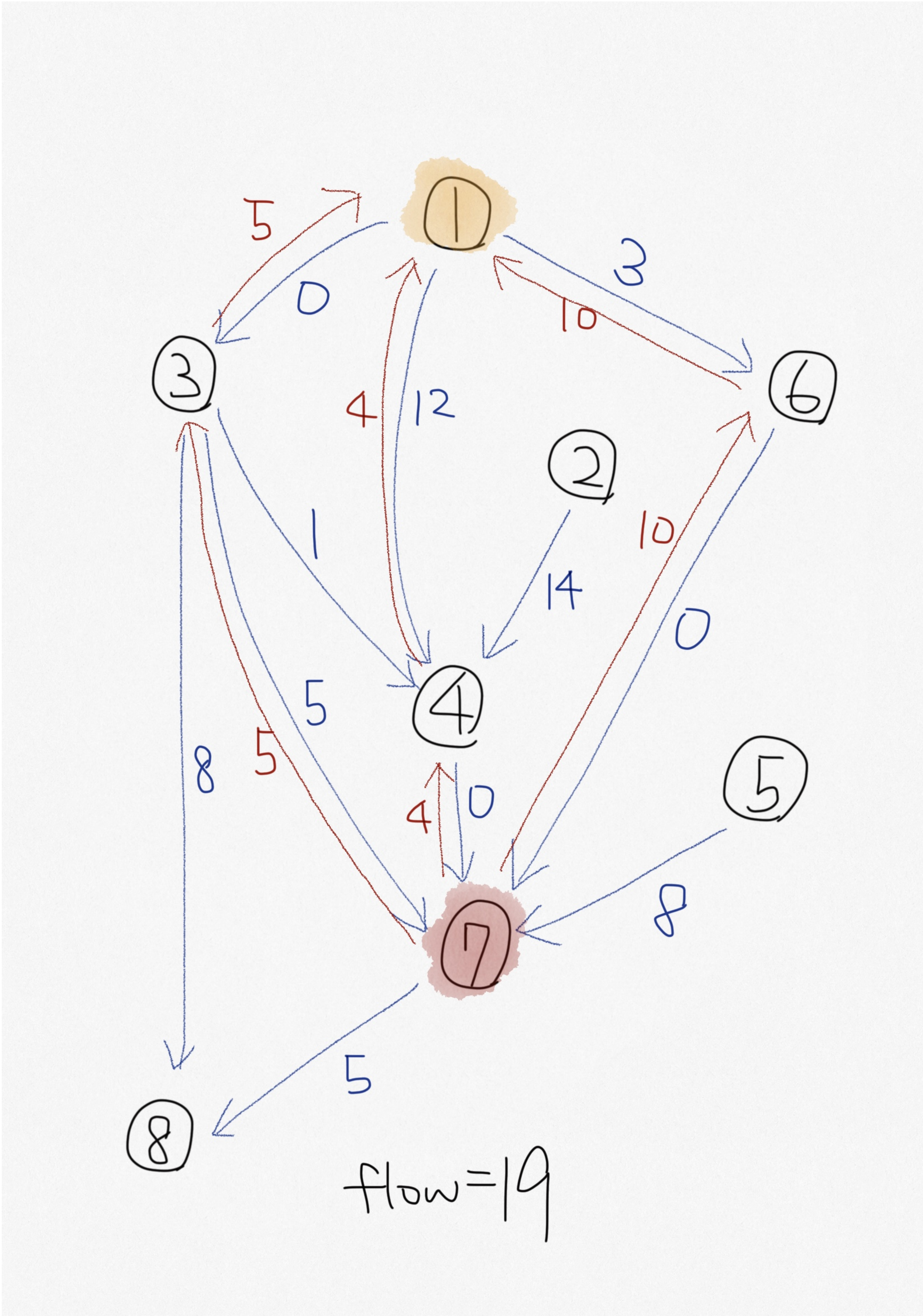
- 取1→3→7為augmenting path，其權為5。更新後的flow為5，Residual Network 如下。



- 取1→4→7作augmenting path，其權為4。更新後的flow為9，Residual Network 如下。

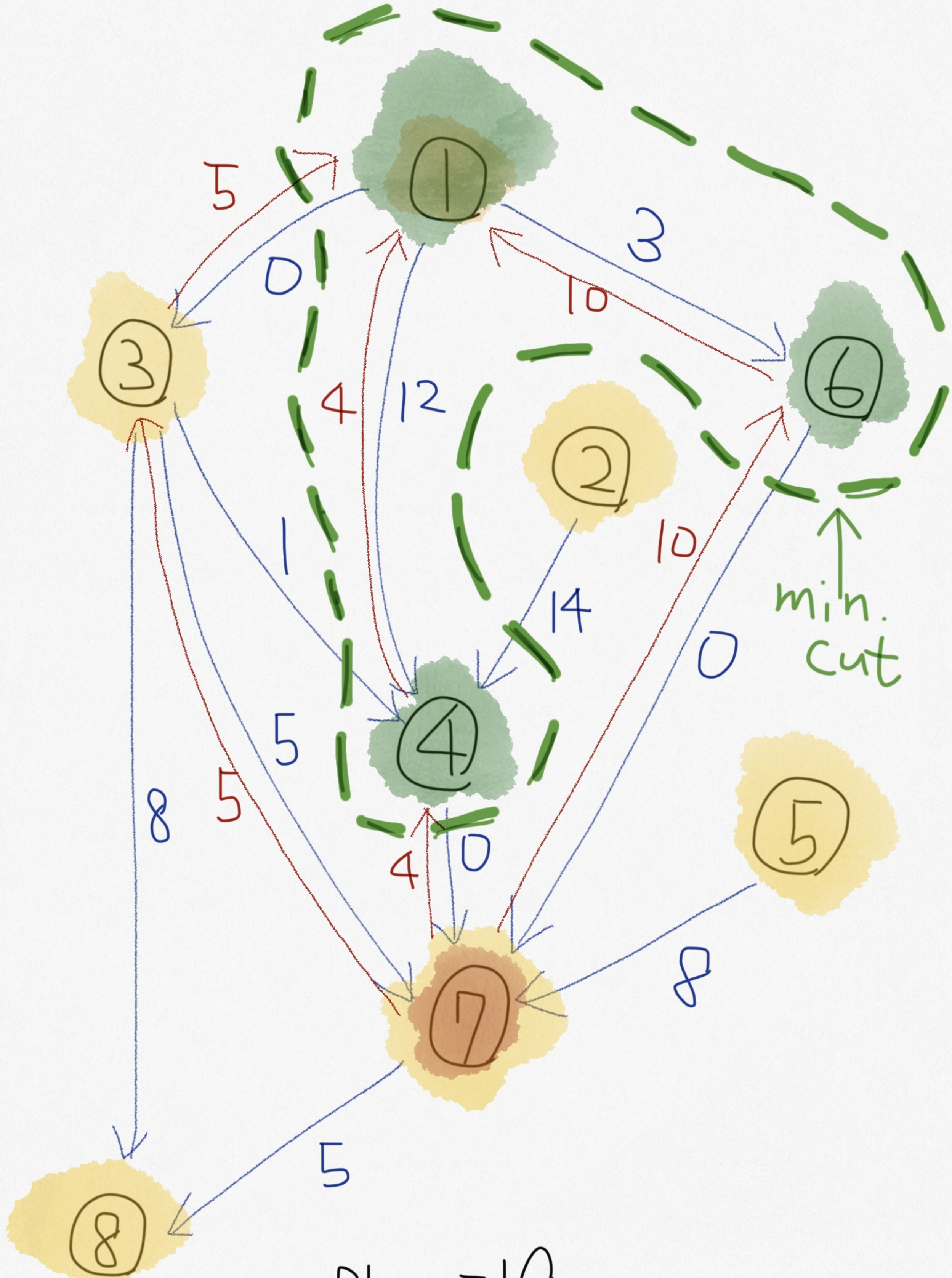


- 取1→6→7作augmenting path，其權為10。更新後的flow為19，Residual Network 如下。



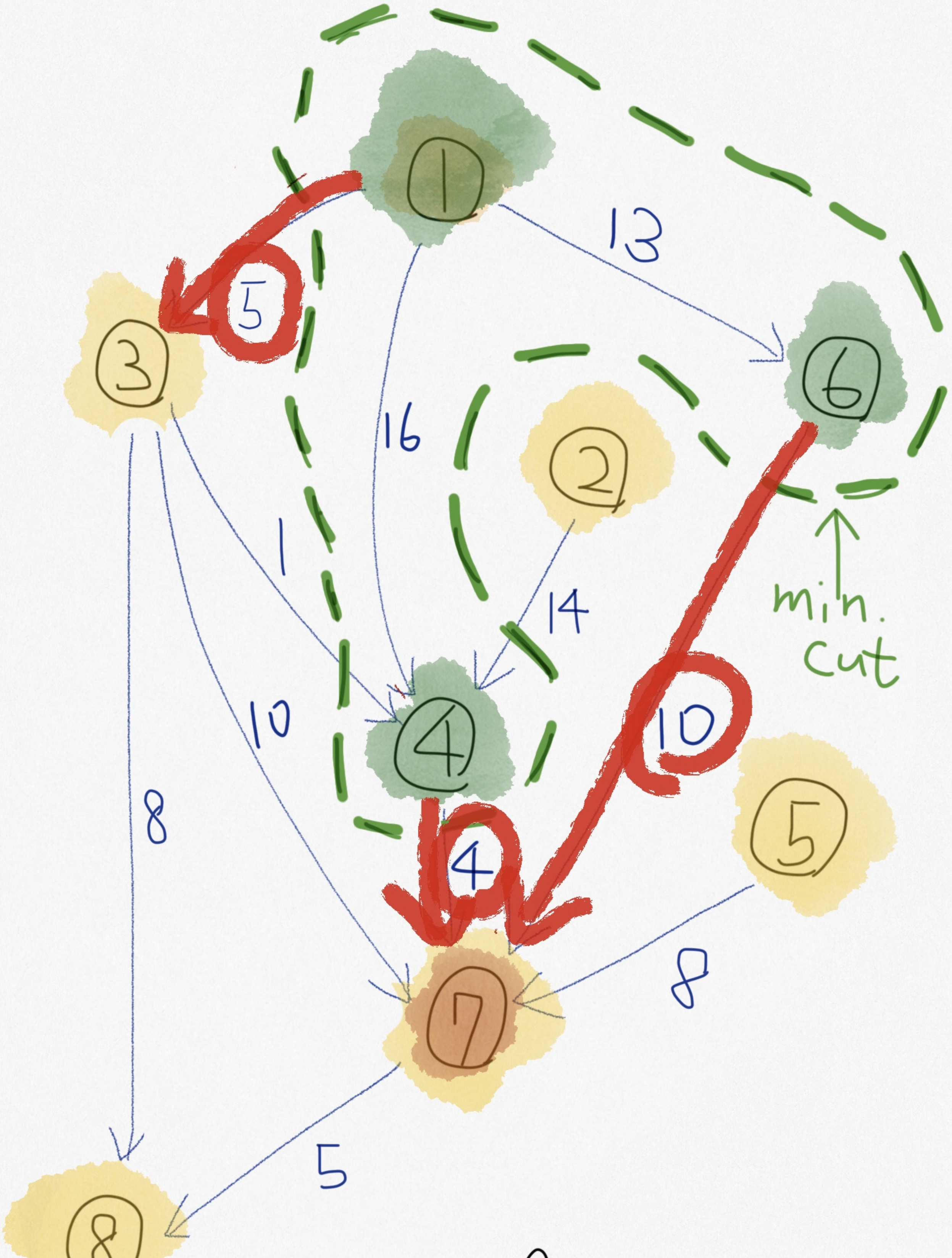
- 找不到更多的augmenting path，故上圖即為最大流，其流量flow為19。

2. 從上圖中，從節點1出發，將所有可以到達的節點分成一組(下圖中塗綠的)，直到這些節點皆不能通往其餘的(塗黃的)節點。因此，若我們決定一個cut(V, S)，使節點1,4,6屬於V，節點2,3,5,7,8屬於S，該cut即為minimum cut，如下圖。



flow=19

3. 如下圖，從V(綠色節點)流到S(黃色節點)的最大容量如圖中紅色箭頭所示，為5+4+10=19，與最大流相同。



flow=19

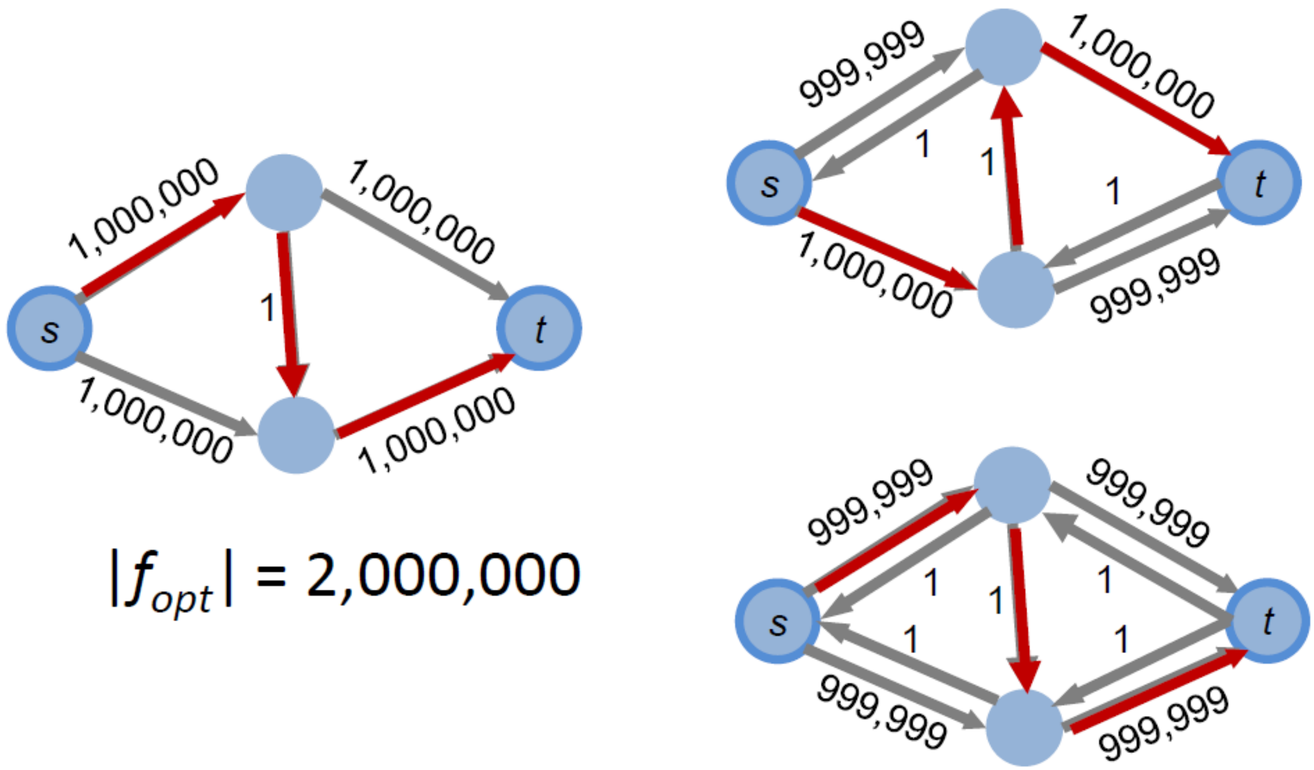
當我們用minimum cut把圖分成「包含source的節點集合」(下稱集合V)和「包含sink的節點集合」(下稱集合S)後，所有從source到sink的流都必須經過V指向S的邊，minimum cut代表這些邊是流的瓶頸(最大容量)。用盡瓶頸處(a.k.a. minimum cut)的最大容量即代表全圖最大流量，故(1)與(2)答案相同。

Problem 2

參考文件:

`http://pisces.ck.tp.edu.tw/~peng/index.php?action=showfile&file=f6cdf7ef750d7dc79c7d599b942acbaaee86a2e3e`

1. 參考教授投影片13-Flow-P2.pdf第21頁，如下圖。



   - 在最差狀況下，DFS每次都會選容量為1的邊。
   - 如果最大流量為f^*，增廣流量就要執行f^*次。
   - 每執行一次增廣流量就要執行一次DFS (題目psuodo code第二行)。
   - 使用adjacency list下執行DFS的時間複雜度為$O(V+E)$。
   - 考慮$E > V$，總時間複雜度為$O(V+E) \times O(f^*) = O(E \times |f^*|)$。
2.
   - BFS每次都選邊數最少的source to sink路徑。
   - 使用adjacency list下執行BFS的時間複雜度為$O(V+E)$。考慮$E > V$，時間複雜度為$O(E)$。
   - 計算增廣次數
     - 總共有$|E|$個有向邊，由於每個有向邊都有一個搭配的逆流邊(初始為0，隨著增廣的進行會改變)，故圖中總邊數屬於$O(2 \times |E|) = O(|E|)$。
     - 由於使用BFS，每次找到的路徑一定是越來越長(經過的邊越來越多)。每次BFS找到的路徑，一定會有一個容量最小的邊，將它命名為「瓶頸」。則這個邊在下一次增廣時若被使用，一定是使用他的逆流(圜方向已經被用光了)。

        - 每執行一次增光，就執行一次BFS。每執行一次BFS，就有一條邊會是「瓶頸」；因此若把所有邊成為「瓶頸」的次數加總，就是增廣執行的次數。
        - 由於最長的BFS路徑長度為$V-1$，一個邊最多成為$(V-1)/2=O(V)$次「瓶頸」，即一個邊最多執行$O(V)$次增廣。
        - 總共有O(E)條邊，所以總共最多執行$O(VE)$次增廣。
    - 總共$O(VE)$次增廣，每次增廣都作一次時間複雜度$O(E)$的BFS，故總時間複雜度為$O(VE\times E=)=O(VE^2)$。
3. 因為DFS和BFS會選到不同的增廣路徑。BFS保證每次都是選最短路徑當作增廣路徑，如上所述，增廣次數是與容量無關的，僅與節點數和邊數有關。相較之下，DFS的時間複雜度有最大流項$|f^*|$，使得如果最大流遠大於邊數時，DFS的時間複雜度匯巨幅上升。