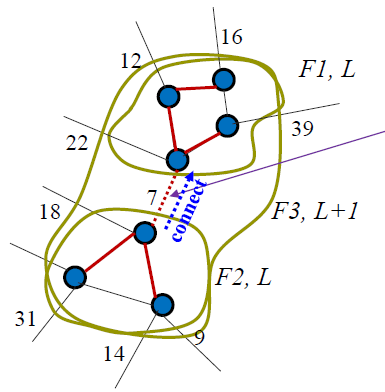
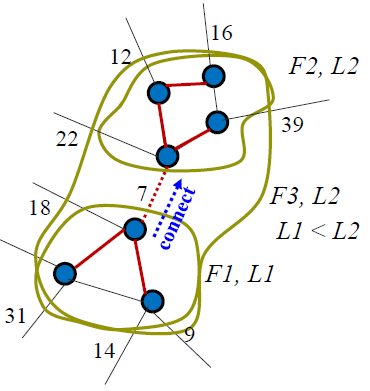
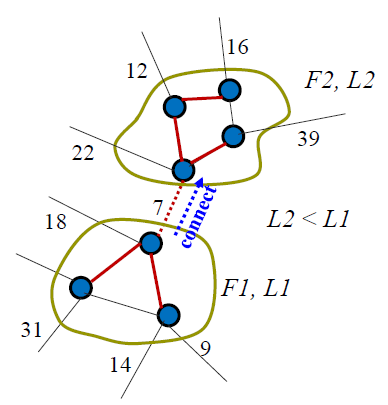
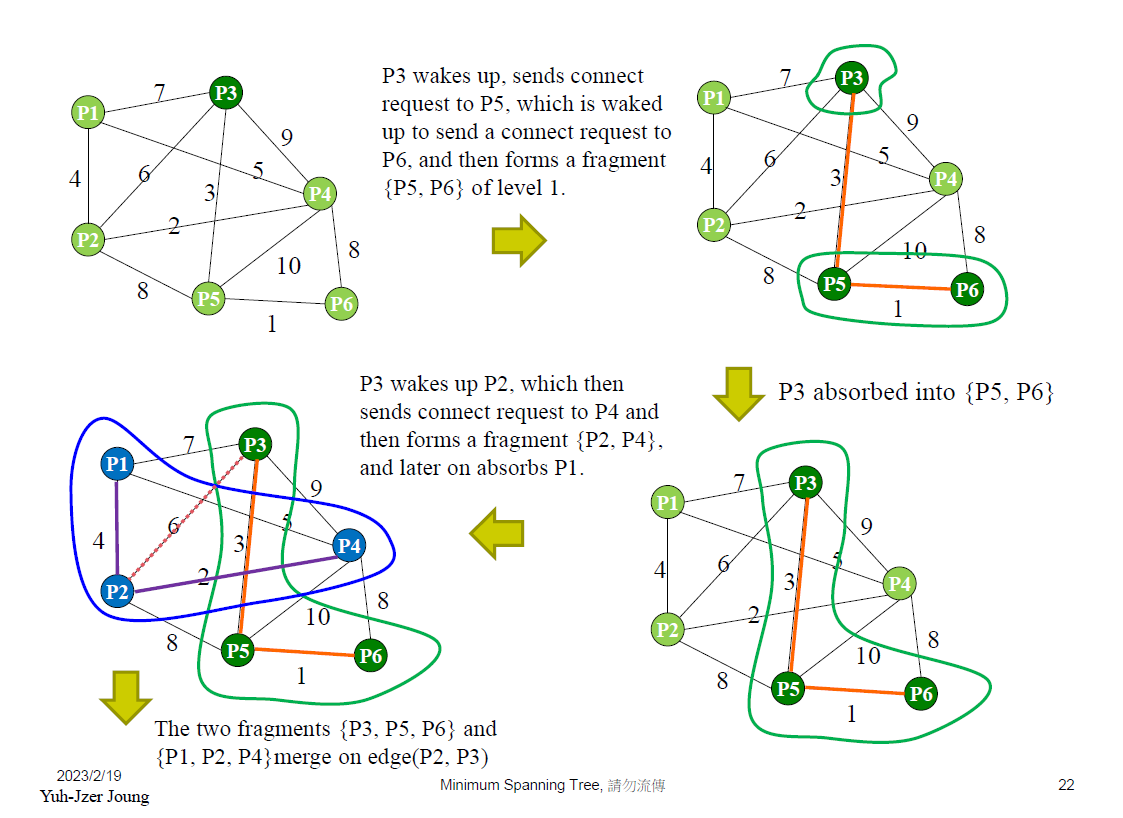
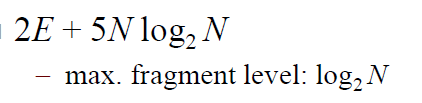
**Distributed Minimum Spanning Tree (MST)**

MST定義

* a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum total edge weight.
  + 要在一個有weight並且尚未決定方向性的graph中，找出一個graph擁有minimum total edge weight，而且沒有cycle
  + 只要在MST上就可以用最低的weight走到任何node上
* Sequential的演算法有global knowledge，所以可以從最小的找到最大的(不管是node還是edge)，最後形成MST
* Distributed的演算法沒有global knowledge，所以要修改設計方式

GHS algorithm for Distributed MST

* 假設
  + asynchronous message-passing model
  + messages are delivered in FIFO order in each edge
  + each node knows its own edges and weights (知道自己和鄰居)
  + either edge weights are distinct, or nodes have distinct IDs
    - 實際上edge weight可以一樣，但ID一定要不同，不然無法決定要選誰當min weight edge (如果ID相同weight就要不同)
  + edges of the same weight can be ordered by the ids of their adjacent nodes,
    - e.g., [6, (P2, P3)] < [6, (P4, P5)]
  + each node can spontaneously initiate the algorithm, or be awakened by neighbors upon receipt of messages (可以自發性啟動，或是通過他人喚醒)
* Lemma 1:
  + Given a fragment (subtree) of an MST, let e be a minimum-weight outgoing edge of the fragment.
  + 給一個MST的fragment，找出銓重最小的edge，就可以拿他跟相鄰的non-fragment node合併成新的fragment
    - 這時候只要把這個edge和她相鄰的non-fragment node(如果是fragment有加入的條件)加入現在這個fragment，形成新的fragment
      * 因為MST一定要相連，所以out-edge中一定要選一條，那 當然是選weight最小的
    - 這樣子一直往外長最後就會形成完整的MST
* Lemma 2:
  + 如果edge的weight都不一樣，那這個MST一定是unique
    - 因為每次找到的edge都會是一樣的(weight最低的edge)
* 以level的角度來思考
  + 一開始每個node各自都在level 0的狀態
  + Fragment合併之後，level就+1
    - 
      * F1、F2合併成F3，level從L變成L+1
      * The connecting edge becomes the“core” of the new fragment
        + 這個core是用來決定fragment要往哪裡長(決定哪條edge的weight最低)
        + 這裡的core雖然是edge，但對應的兩個node其實就是當負責決定的人 (也可以只選其中一個node)
  + 如果兩個fragment level不同?
  + Case1: level小的要求connect level大的
    - 
      * 這樣子level大的會直接合併level小的
  + Case2: level大的要求 connect level小的
    - 
    - Level小的會先不理level大的，這樣可以減少cost (因為先合併的話，之後小的每次都要先回去問跟大的合併的部分，來決定如何向外長)
      * 如果level小的長到跟level大一樣，就會回去request說要合併
      * 如果level小的還是比較小，他最終還是會發出request跟level大的合併
      * P11頁可以去看看
* Node States
  + Sleeping
    - 可以自發性啟動，或是被他人喚醒
  + Find:
    - 找minimum-weight outgoing edge (找向外長的地方)
  + Found
    - waiting for a response from the fragment (等他人回應)
* Edge State
  + Branch:
    - 成功長的edge，屬於MST的一部份
  + Rejected:
    - 拒絕長的edge，不屬於MST
  + Basic:
    - 還沒被決定屬不屬於MST
* Message Type (P14，自己看)
  + Initiate: 開始找min weight out-edges
  + connect: 找到weight最小的edge，要求連起來
  + Test: 被要求connect的 node會先去找他自己最小的 min weight out-edges，如果有更小的就先不connect
  + Reject/Accept: 如果Test找到更小的就拒絕，沒找到就接受
  + Report(w): 如果Accept，回報給 core，告知fragment要往哪裡長
  + change-core: 決定新的core
  + Example: (P)
    - 
* 如何證明其正確性?
  + Safety:
    - 不會產生非MST的連接(Ex: cycle、weight太大的edge)
    - 根據lemma 1:
      * Given a fragment (subtree) of an MST, let e be a minimum-weight outgoing edge of the fragment.
      * 因為他一定會找min weight，所以找出來的weight絕對是最小的，也沒有cycle(因為由小w找到大w，不會有大w指回小w產生cycle的情形)，符合MST
  + Liveness:
    - 最後一定會產生MST
      * 因為Fragment之間會一直找min weigh edges連線，即使connect請求暫時沒被回應，最後也會連起來，所以最終一定會形成MST
* Message complexity
  + 
* Time complexity
  + Worst case
    - O(N^2):
      * 假設有n個node，每個node都要跟其他node詢問一次可不可以合併，總共要請求n^2次
  + If all nodes are awakened initially
    - 
* Impossibility Result
  + 如果edge weight和node id不是unique (至少node id要是unique)，這個方法就不work
    - 因為這樣無法決定min weight edges