Minimum Spanning Trees

Kuan-Yu Chen (陳冠宇)

2020/12/14 @ TR-313, NTUST

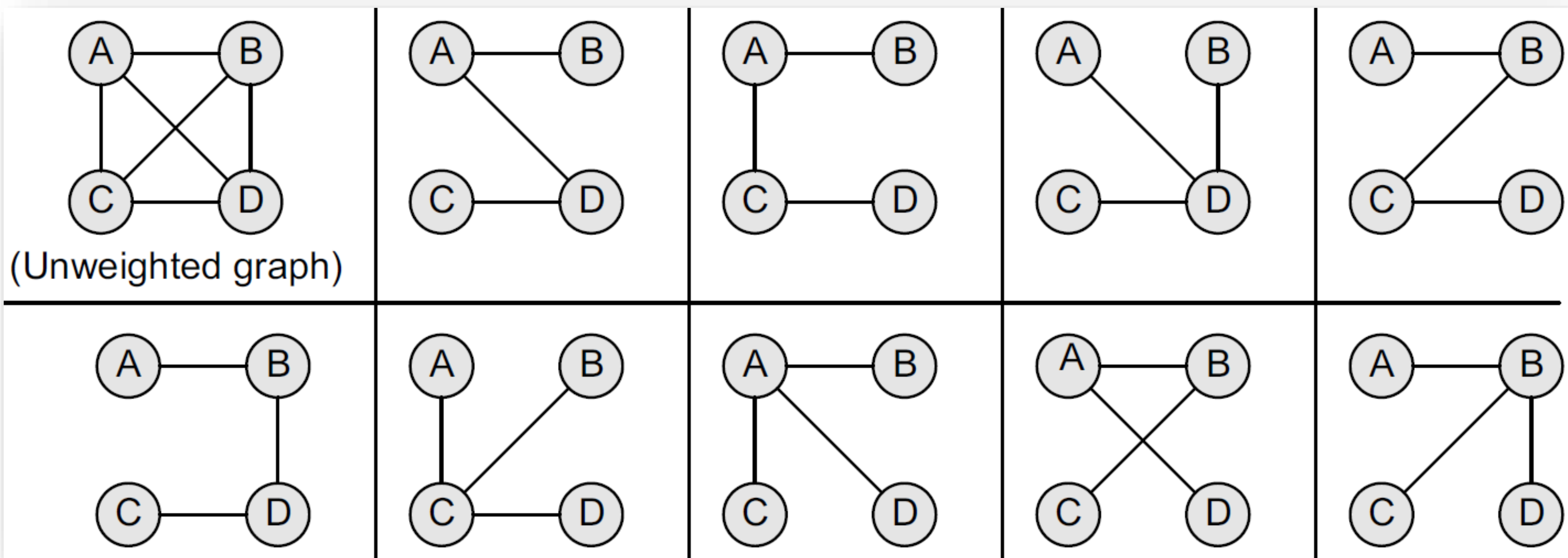
Review

- Three common ways of storing graphs
  - Sequential representation
    - adjacency matrix
  - Linked representation
    - linked list
  - Adjacency multi-list

- Search algorithms
  - BFS
  - DFS

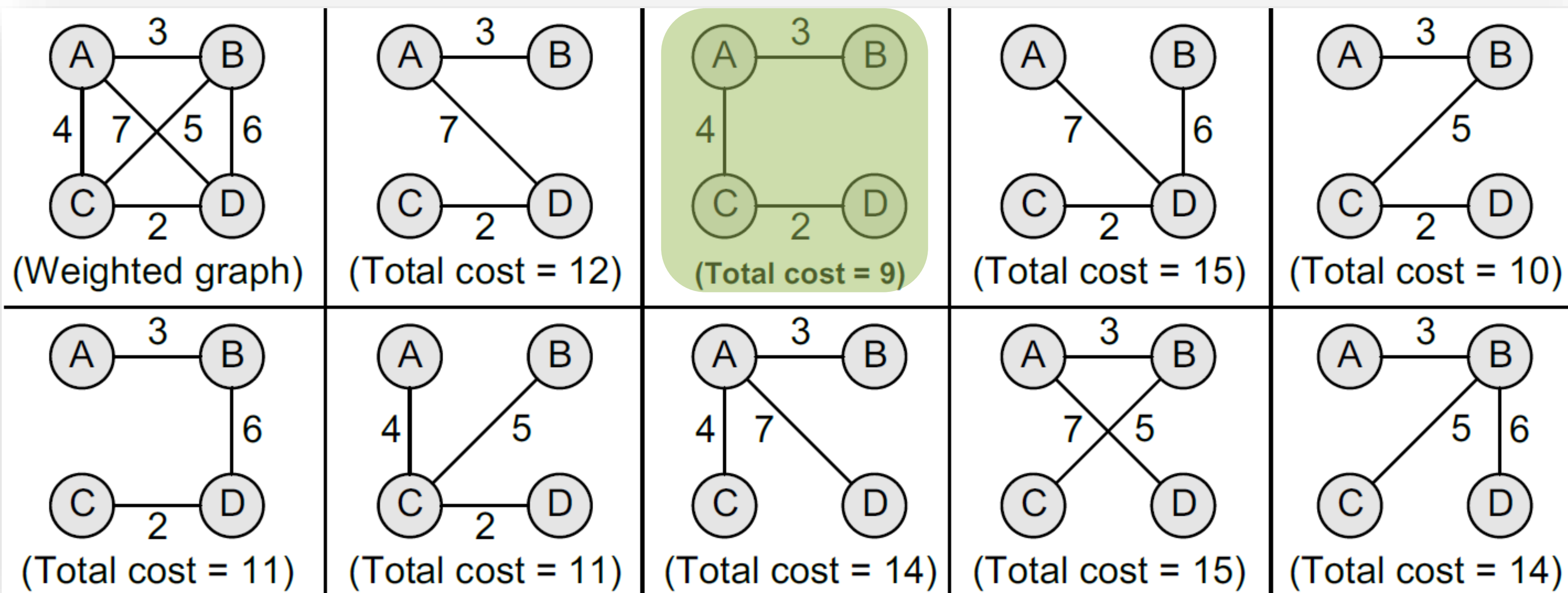
Spanning Tree

- A spanning tree of a connected and undirected graph G is a sub-graph of G which is a tree that connects all the vertices together
  - A graph G can have many different spanning trees



Minimum Spanning Tree.

- A **minimum spanning tree** (MST) is defined as a spanning tree with weight less than or equal to the weight of every other spanning tree
  - We can assign **weights** to each edge, and use it to assign a weight to a spanning tree by calculating the sum of the weights of the edges in that spanning



Minimum Spanning Tree..

- Properties
  - ***Possible multiplicity***
    - There can be multiple minimum spanning trees of the same weight
    - Particularly, if all the weights are the same, then every spanning tree will be minimum
  - ***Uniqueness***
    - When each edge in the graph is assigned a different weight, then there will be only one unique minimum spanning tree
  - ***Simplicity***
    - For an unweighted graph, any spanning tree is a minimum spanning tree

Minimum Spanning Tree…

- Minimum spanning trees can be computed quickly and easily to provide optimal solutions
  - Prim’s algorithm
  - Kruskal’s algorithm

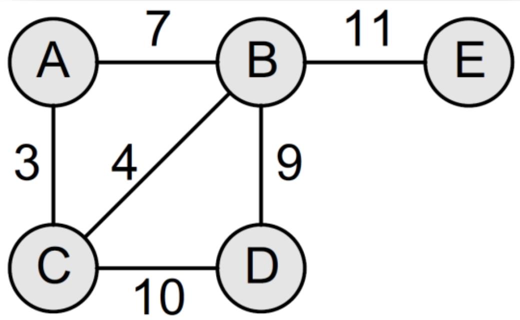
Prim’s Algorithm.

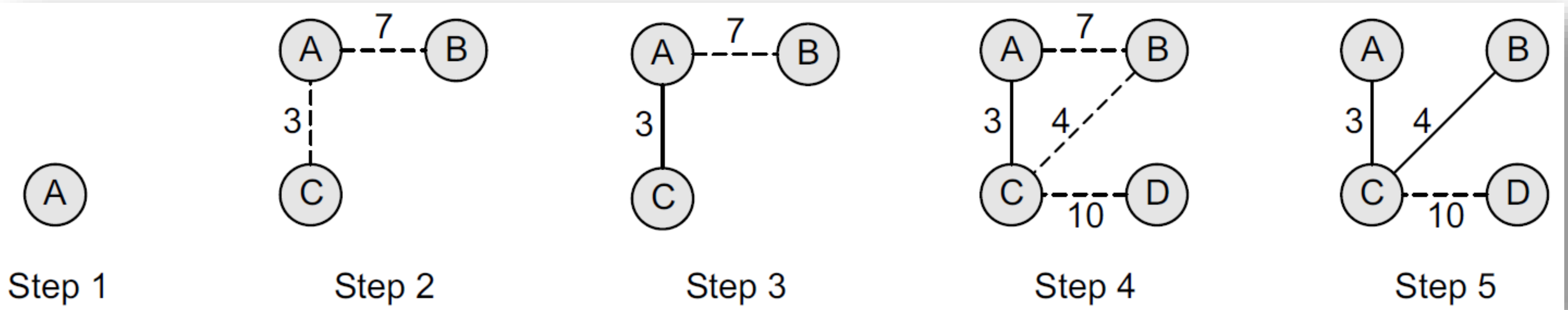
- Prim’s algorithm is a **greedy algorithm** that is used to form a minimum spanning tree for a connected weighted undirected graph
  - **Tree vertices**
    - Vertices that are a part of the minimum spanning tree *T*
  - **Fringe (Neighboring) vertices**
    - Vertices that are currently not a part of *T*, but are adjacent to some tree vertex
  - **Unseen vertices**
    - Vertices that are neither tree vertices nor fringe vertices fall under this category

```
Step 1: Select a starting vertex
Step 2: Repeat Steps 3 and 4 until there are fringe vertices
Step 3:     Select an edge e connecting the tree vertex and
            fringe vertex that has minimum weight
Step 4:     Add the selected edge and the vertex to the
            minimum spanning tree T
        [END OF LOOP]
Step 5: EXIT
```

Prim's Algorithm..

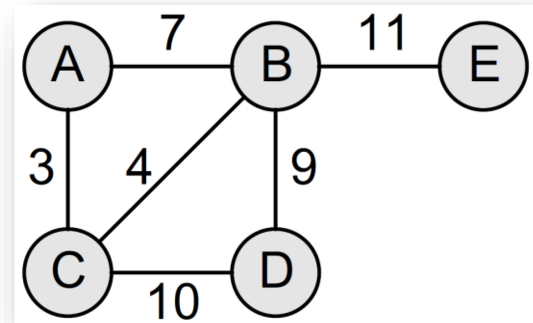
- Construct a minimum spanning tree of the graph by using Prim's algorithm
  - Step 1: Choose a starting vertex *A*
  - Step 2: Add the fringe vertices (that are adjacent to *A*)
  - Step 3: Since the edge connecting *A* and *C* has less weight, add *C* to the tree
  - Step 4: Add the fringe vertices (that are adjacent to *C*)
  - Step 5: Since the edge connecting *C* and *B* has less weight, add *B* to the tree

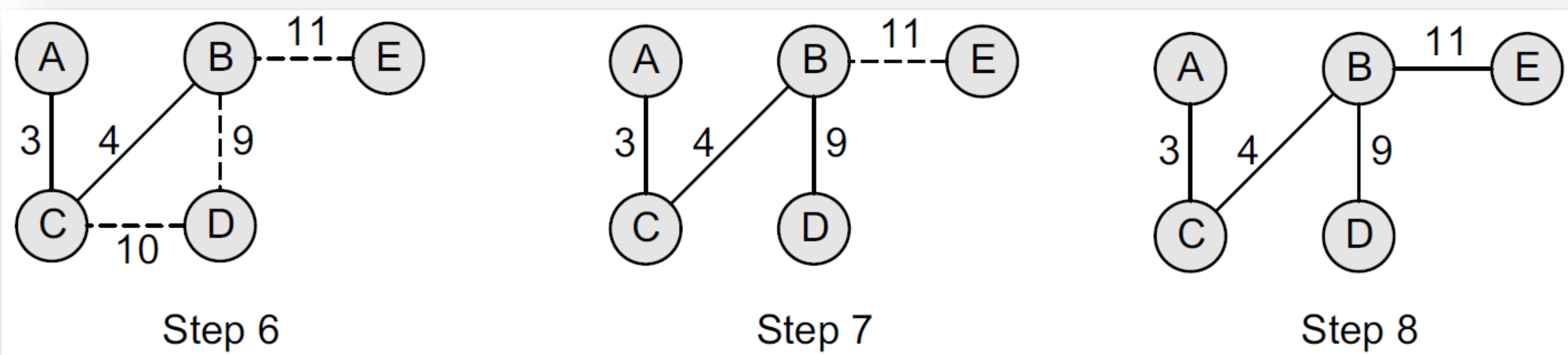




Prim's Algorithm…

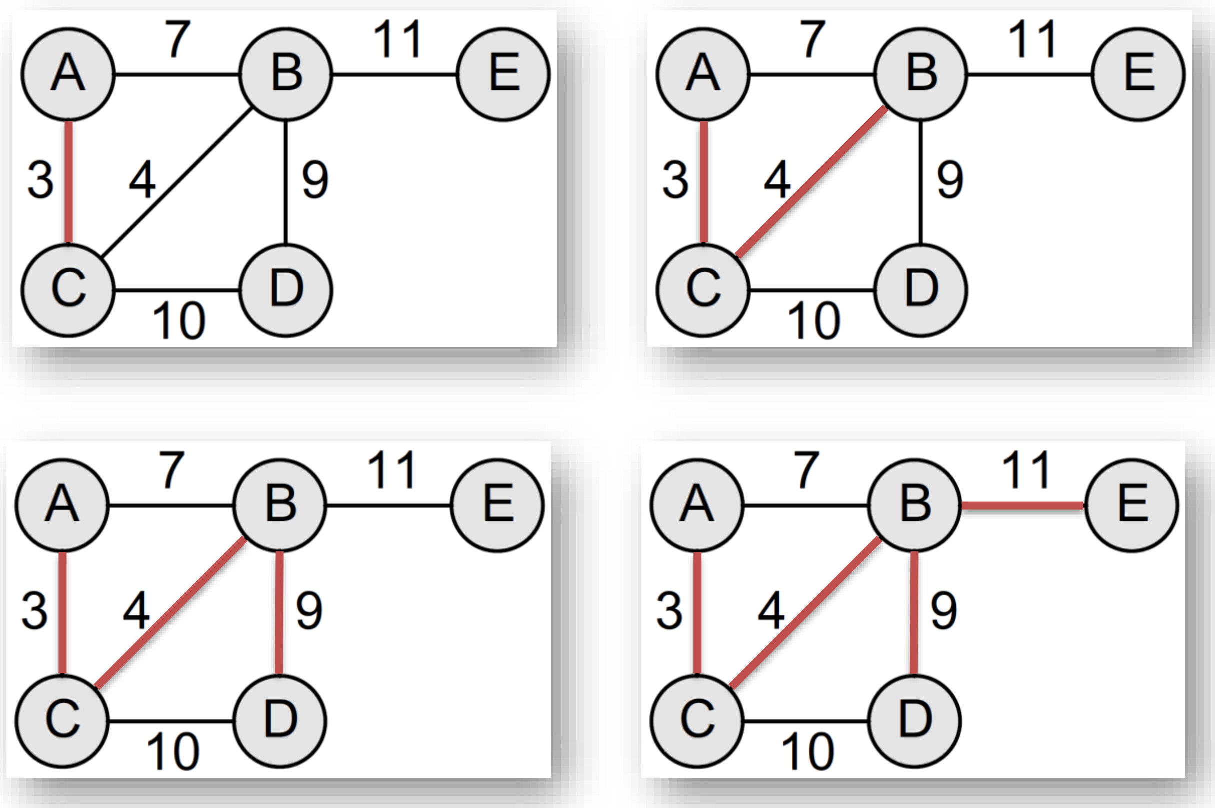
- Step 6: Add the fringe vertices (that are adjacent to B)
- Step 7: Since the edge connecting B and D has less weight, add D to the tree
- Step 8: Add E to the tree





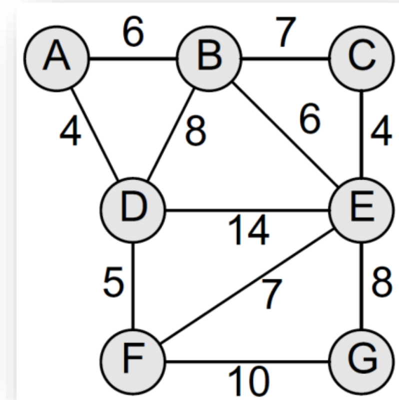
Prim's Algorithm….

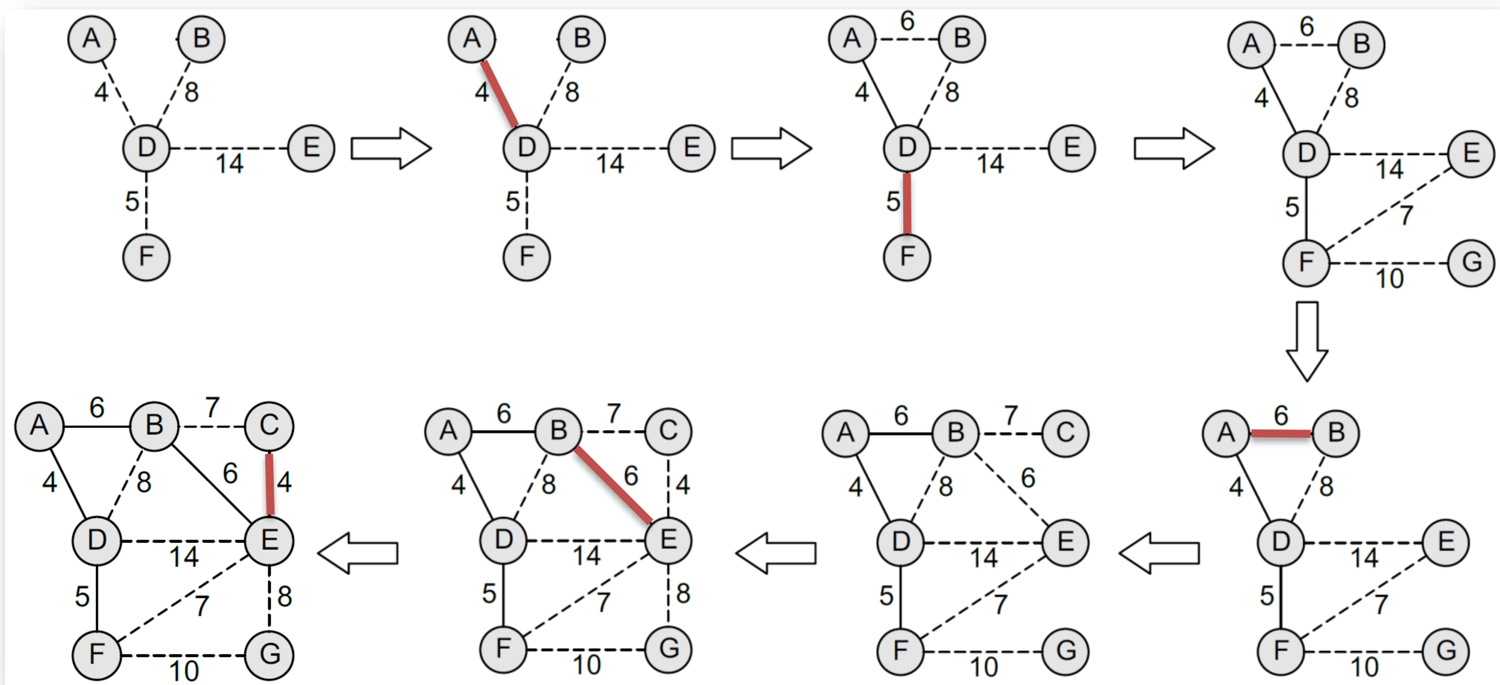
- By looking!

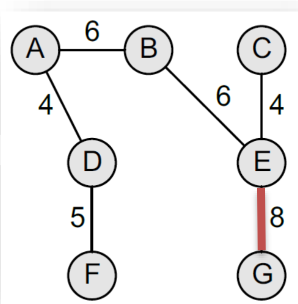


Prim's Algorithm…..

- Construct a minimum spanning tree of the graph by using Prim's algorithm from vertex *D*







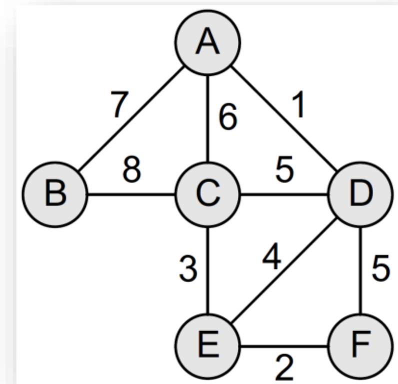
Kruskal's Algorithm.

- Kruskal's algorithm is used to find the minimum spanning tree for a connected weighted undirected graph
  - If the graph is not connected, then it finds a **minimum spanning forest**

```
Step 1: Create a forest in such a way that each graph is a separate
        tree.
Step 2: Create a priority queue Q that contains all the edges of the
        graph.
Step 3: Repeat Steps 4 and 5 while Q is NOT EMPTY
Step 4:     Remove an edge from Q
Step 5: IF the edge obtained in Step 4 connects two different trees,
        then Add it to the forest (for combining two trees into one
        tree).
        ELSE
            Discard the edge
Step 6: END
```

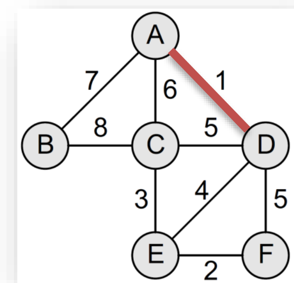
Kruskal's Algorithm..

- Apply Kruskal's algorithm on the given graph
  - Initial:
    - F = {{A}, {B}, {C}, {D}, {E}, {F}}
    - MST = {}
    - Priority Queue Q = {(A, D), (E, F), (C, E), (E, D) (C, D), (D, F), (A, C), (A, B), (B, C)}



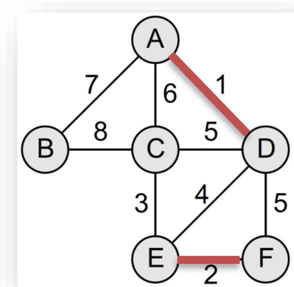
  - Step1:
    - Remove the edge (A, D) from Q

F = {{A, D}, {B}, {C}, {E}, {F}}
MST = {A, D}
Q = {(E, F), (C, E), (E, D), (C, D), (D, F), (A, C), (A, B), (B, C)}



  - Step2:
    - Remove the edge (E, F) from Q

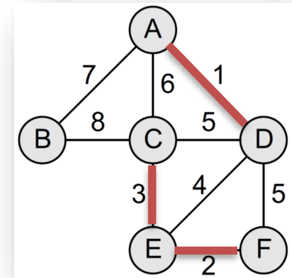
F = {{A, D}, {B}, {C}, {E, F}}
MST = {(A, D), (E, F)}
Q = {(C, E), (E, D), (C, D), (D, F), (A, C), (A, B), (B, C)}



Kruskal's Algorithm...

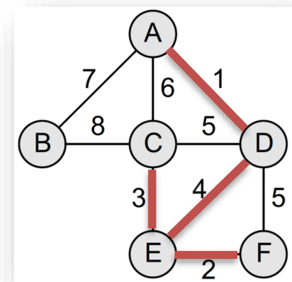
- Step3:
  - Remove the edge (C, E) from Q

F = {{A, D}, {B}, {C, E, F}}
MST = {(A, D), (C, E), (E, F)}
Q = {(E, D), (C, D), (D, F), (A, C), (A, B), (B, C)}



- Step4:
  - Remove the edge (E, D) from Q

F = {{A, C, D, E, F}, {B}}
MST = {(A, D), (C, E), (E, F), (E, D)}
Q = {(C, D), (D, F), (A, C), (A, B), (B, C)}



- Step5:
  - Remove the edge (C, D) from Q

    The edge does not connect different trees, so simply discard this edge

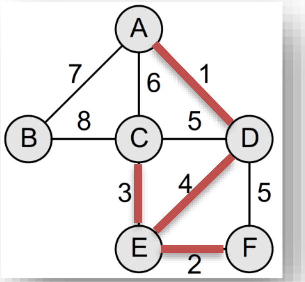
F = {{A, C, D, E, F}, {B}}
MST = {(A, D), (C, E), (E, F), (E, D)}
Q = {(D, F), (A, C), (A, B), (B, C)}

Kruskal's Algorithm….

– Step6:
  - Remove the edge (D, F) from Q

    The edge does not connect different trees, so simply discard this edge

$$
\begin{aligned}
F &= \{\{A, C, D, E, F\}, \{B\}\} \\
MST &= \{(A, D), (C, E), (E, F), (E, D)\} \\
Q &= \{(A, C), (A, B), (B, C)\}
\end{aligned}
$$



– Step7:
  - Remove the edge (A, C) from Q

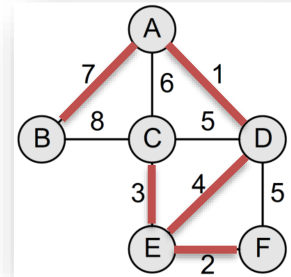
    The edge does not connect different trees, so simply discard this edge

$$
\begin{aligned}
F &= \{\{A, C, D, E, F\}, \{B\}\} \\
MST &= \{(A, D), (C, E), (E, F), (E, D)\} \\
Q &= \{(A, B), (B, C)\}
\end{aligned}
$$

Kruskal's Algorithm…..

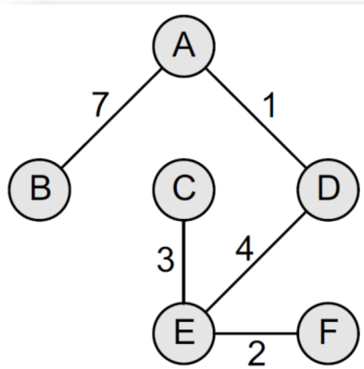
- Step8:
  - Remove the edge (A, B) from Q

F = {A, B, C, D, E, F}
MST = {(A, D), (C, E), (E, F), (E, D), (A, B)}
Q = {(B, C)}



- Step8:
  - Remove the edge (B, C) from Q

    The edge does not connect different trees, so simply discard this edge



F = {A, B, C, D, E, F}
MST = {(A, D), (C, E), (E, F), (E, D), (A, B)}
Q = {}

General Formulation

- They each use a specific rule to determine a safe edge in line 3 of GENERIC-MST

```
GENERIC-MST(G, w)
1  A = Ø
2  while A does not form a spanning tree
3      find an edge (u, v) that is safe for A
4      A = A ∪ {(u, v)}
5  return A
```

  - In Prim's algorithm
    - The set A forms a single tree
    - The safe edge added to A is always a least-weight edge **connecting the tree to a vertex not in the tree**
  - In Kruskal's algorithm
    - The set A is a forest whose vertices are all those of the given graph
    - The safe edge added to A is always a least-weight edge in the graph that **connects two distinct components (trees)**

Questions?

kychen@mail.ntust.edu.tw