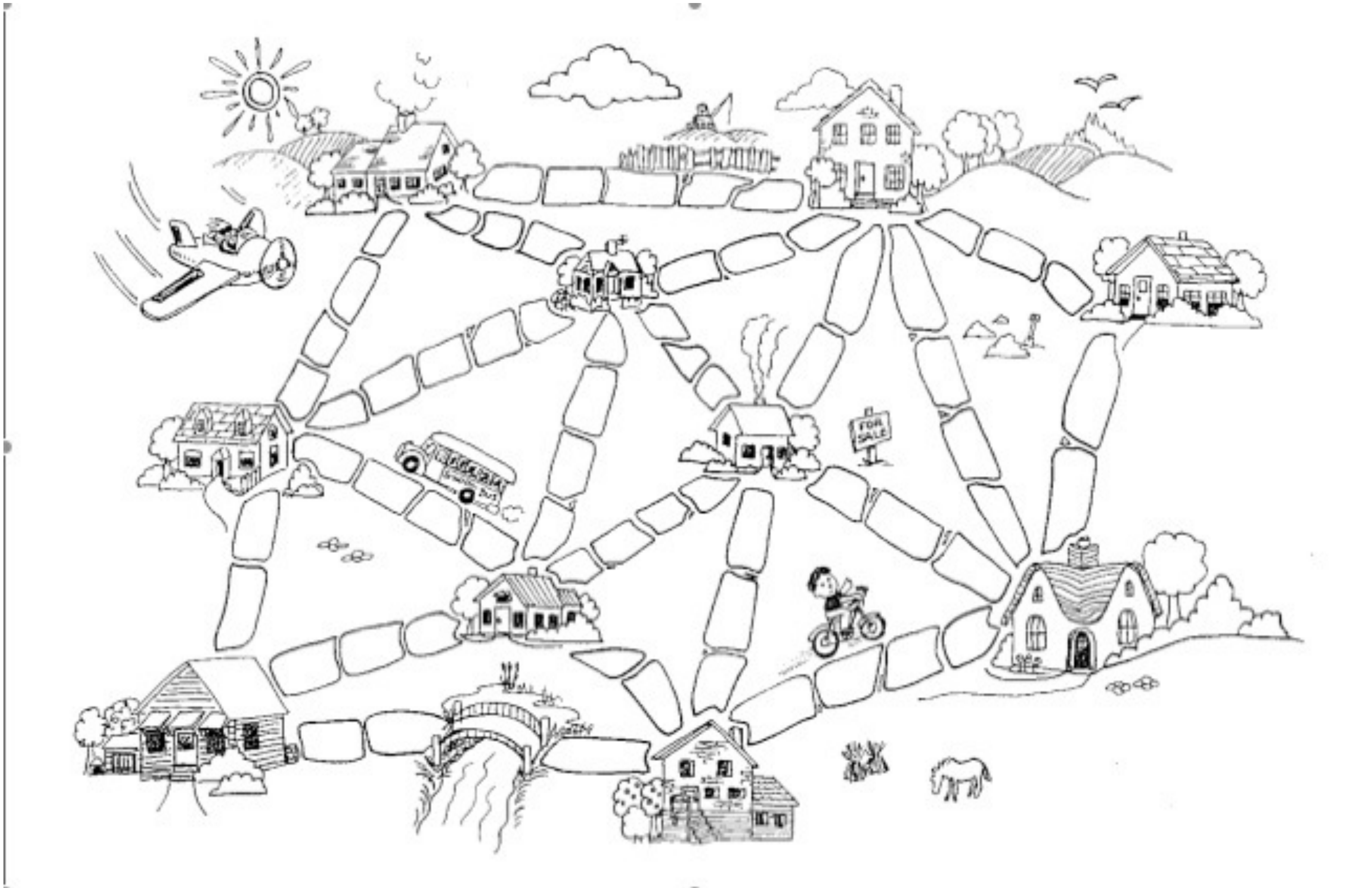
Today's announcements:

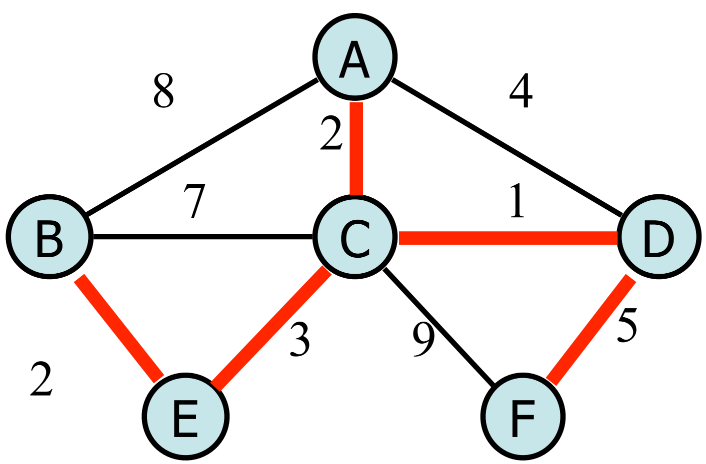
MP7 available. Due 12/8, 11:59p.

Final exam: 12/14, 7-10p, conflict: email ramais@illinois.edu

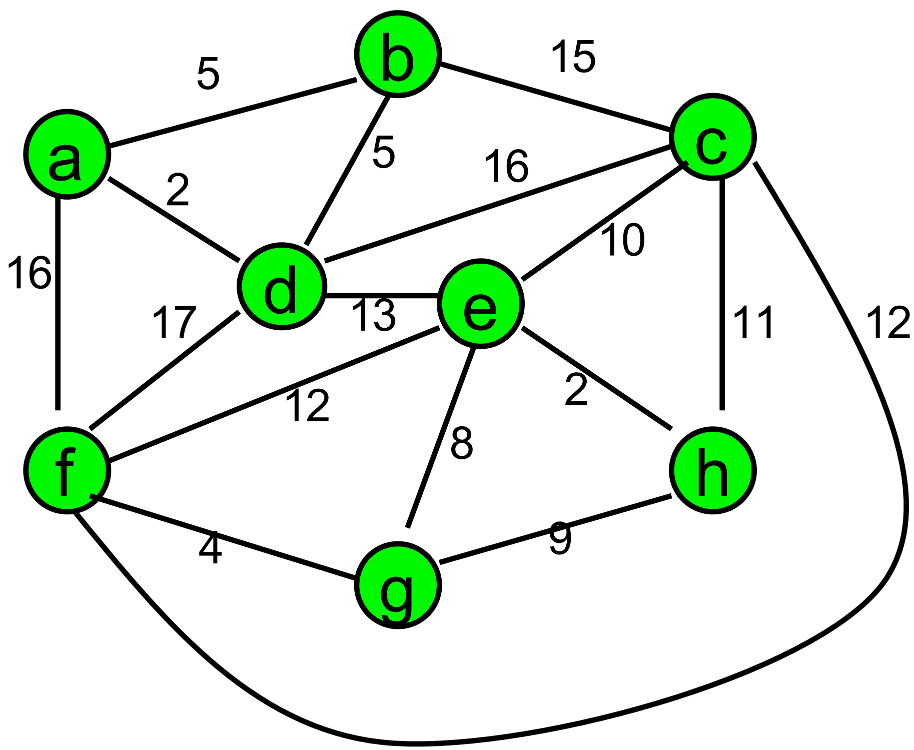


Minimum Spanning Tree Algorithms:

- Input: connected, undirected graph G with unconstrained edge weights
- Output: a graph G' with the following characteristics -
  - G' is a spanning subgraph of G
  - G' is connected and acyclic (a tree)
  - G' has minimal total weight among all such spanning trees -

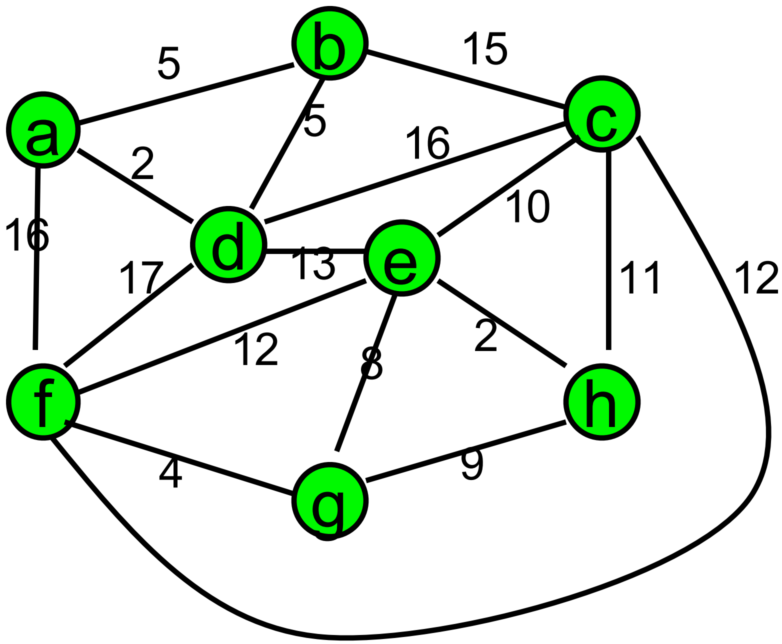


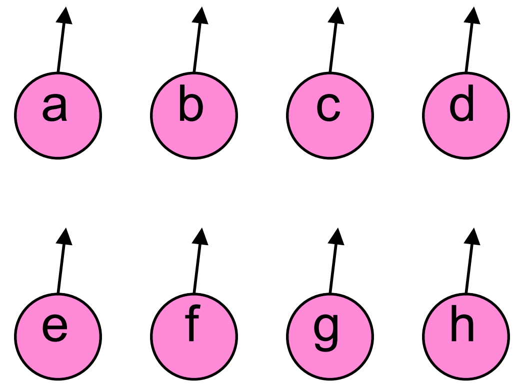
Kruskal's Algorithm



(a,d)
(e,h)
(f,g)
(a,b)
(b,d)
(g,e)
(g,h)
(e,c)
(c,h)
(e,f)
(f,c)
(d,e)
(b,c)
(c,d)
(a,f)
(d,f)

Kruskal's Algorithm (1956)





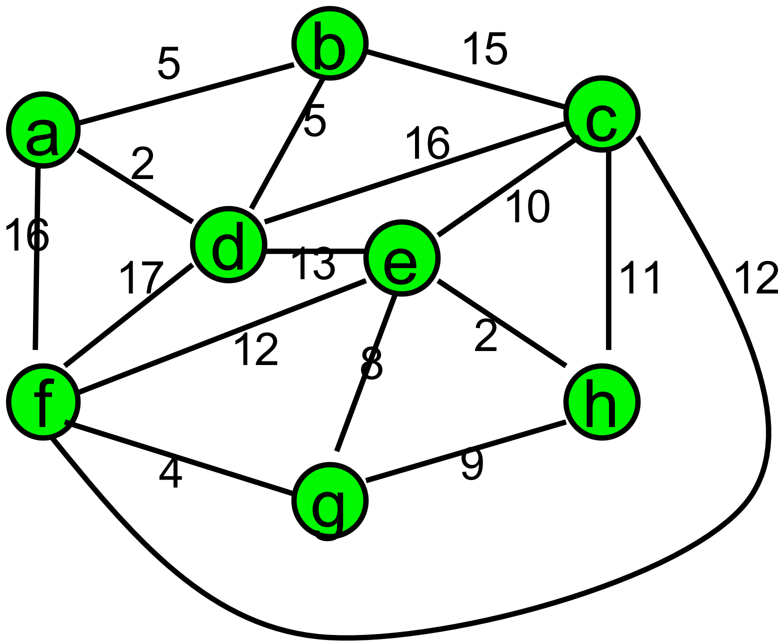
(a,d)
(e,h)
(f,g)
(a,b)
(b,d)
(g,e)
(g,h)
(e,c)
(c,h)
(e,f)
(f,c)
(d,e)
(b,c)
(c,d)
(a,f)
(d,f)

1. Initialize graph T whose purpose is to be our output. Let it consist of all n vertices and no edges.

2. Initialize a disjoint sets structure where each vertex is represented by a set.

3. RemoveMin from PQ. If that edge connects 2 vertices from different sets, add the edge to T and take Union of the vertices' two sets, otherwise do nothing. Repeat until _____ edges are added to T.

Kruskal's Algorithm - preanalysis



Algorithm ***KruskalMST*(*G*)**

disjointSets forest;
for each vertex ***v*** in ***V*** **do**
forest.makeSet(v);

priorityQueue Q;
Insert edges into ***Q,*** keyed by weights

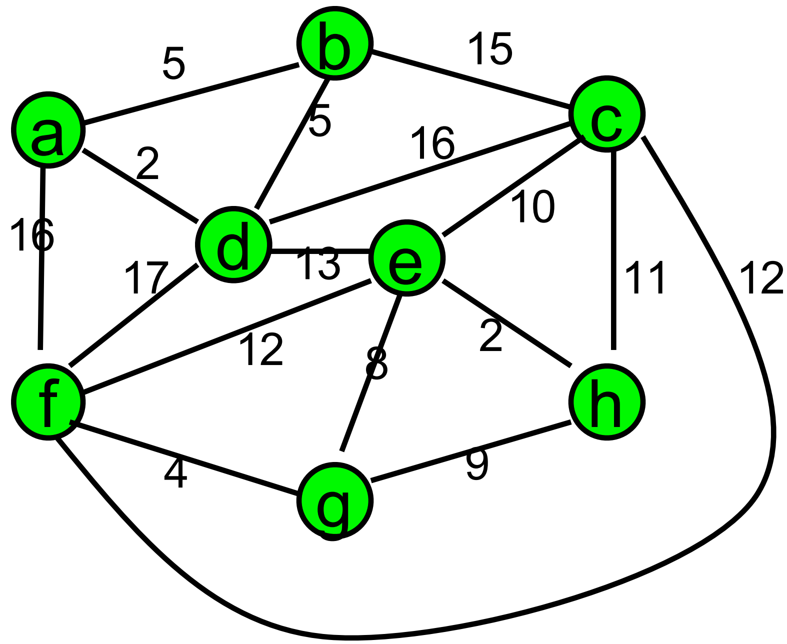
graph T = (V,E) with $E = \emptyset$;

while ***T*** has fewer than ***n***-1 edges **do**
edge ***e*** = ***Q.removeMin()***
Let ***u***, ***v*** be the endpoints of *e*
if *forest.find(v)* ≠ *forest.find(u)* **then**
Add edge ***e*** to ***E***
forest.smartUnion
(forest.find(v),forest.find(u))

return ***T***

Priority Queue:	Heap	Sorted Array
To build		
Each removeMin		

Kruskal's Algorithm - analysis



Algorithm ***KruskalMST*(*G*)**

disjointSets forest;
for each vertex ***v*** in ***V*** **do**
    forest.makeSet(v);

priorityQueue Q;
Insert edges into ***Q,*** keyed by weights

graph T = (V,E) with ***E = ∅;***

while ***T*** has fewer than ***n***-1 edges **do**
    edge ***e*** = ***Q.removeMin()***
    Let ***u***, ***v*** be the endpoints of *e*
    if ***forest.find(v) ≠ forest.find(u)*** **then**
        Add edge ***e*** to ***E***
        forest.smartUnion
            (forest.find(v),forest.find(u))

return ***T***

Priority Queue:	Total Running time:
Heap	
Sorted Array	

Prim's algorithms (1957) is based on the Partition Property:

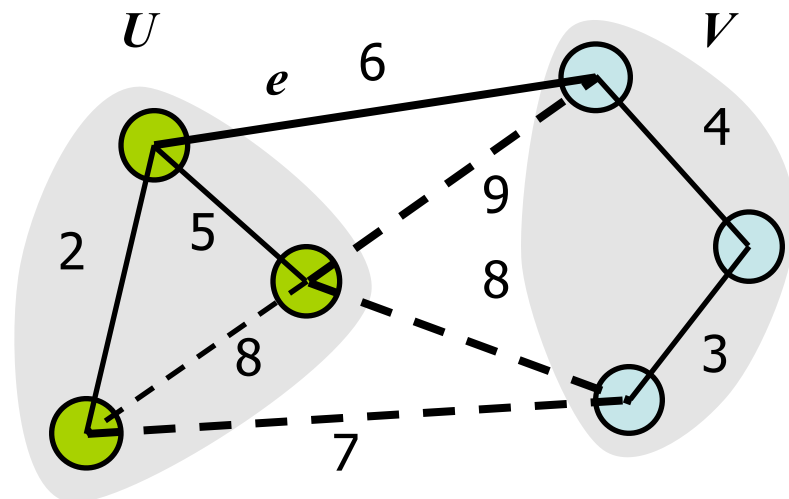
Consider a partition of the vertices of G into subsets U and V.

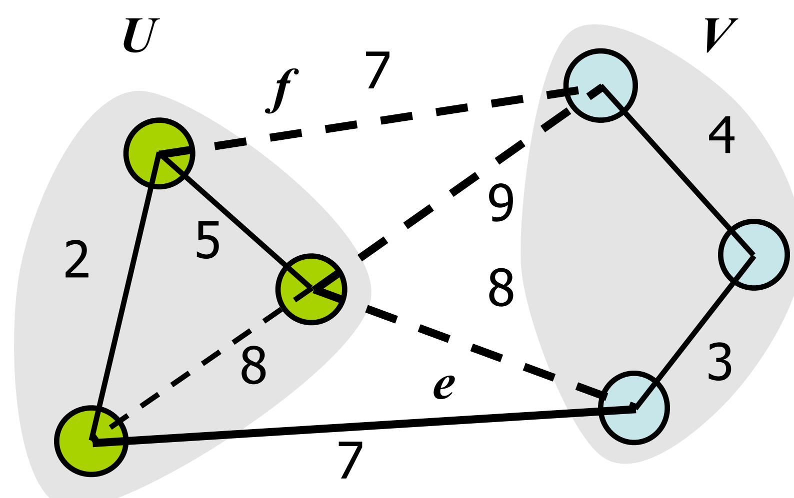
Let e be an edge of minimum weight across the partition.

Then e is part of some minimum spanning tree.

Proof:

See cs374





MST - minimum total weight spanning tree

Theorem suggests an algorithm...