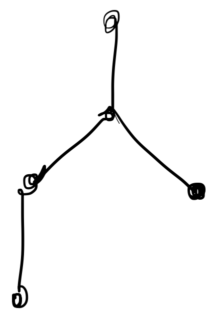
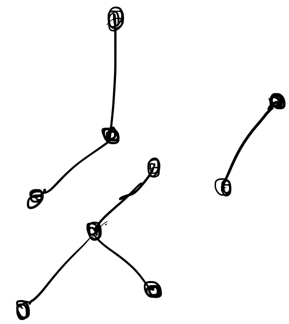
Tree and Forest.

In graph theory, a **tree** is an undirected graph in which any two vertices are connected by *exactly one* path, or equivalently a connected acyclic undirected graph.[1] A **forest** is an undirected graph in which any two vertices are connected by *at most one* path, or equivalently an acyclic undirected graph, or equivalently a disjoint union of trees.[2] [1]



Tree

Connected, Acyclic, Undireted Graph



Forest.

Acyclic, undirected Graph.

[1] Wiki: Tree

Minimum Spanning Tree.

Given a connected, directed graph $G = (V, E)$, the acyclic subset $T \subseteq E$ that connects all of the vertices and whose total weight

$$w(T) = \sum_{(u,v) \in T} w(u, v)$$

is minimized. [1]

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

This greedy strategy is captured by the following generic method, which grows the minimum spanning tree one edge at a time. The generic method manages a set of edges A, maintaining the following loop invariant:

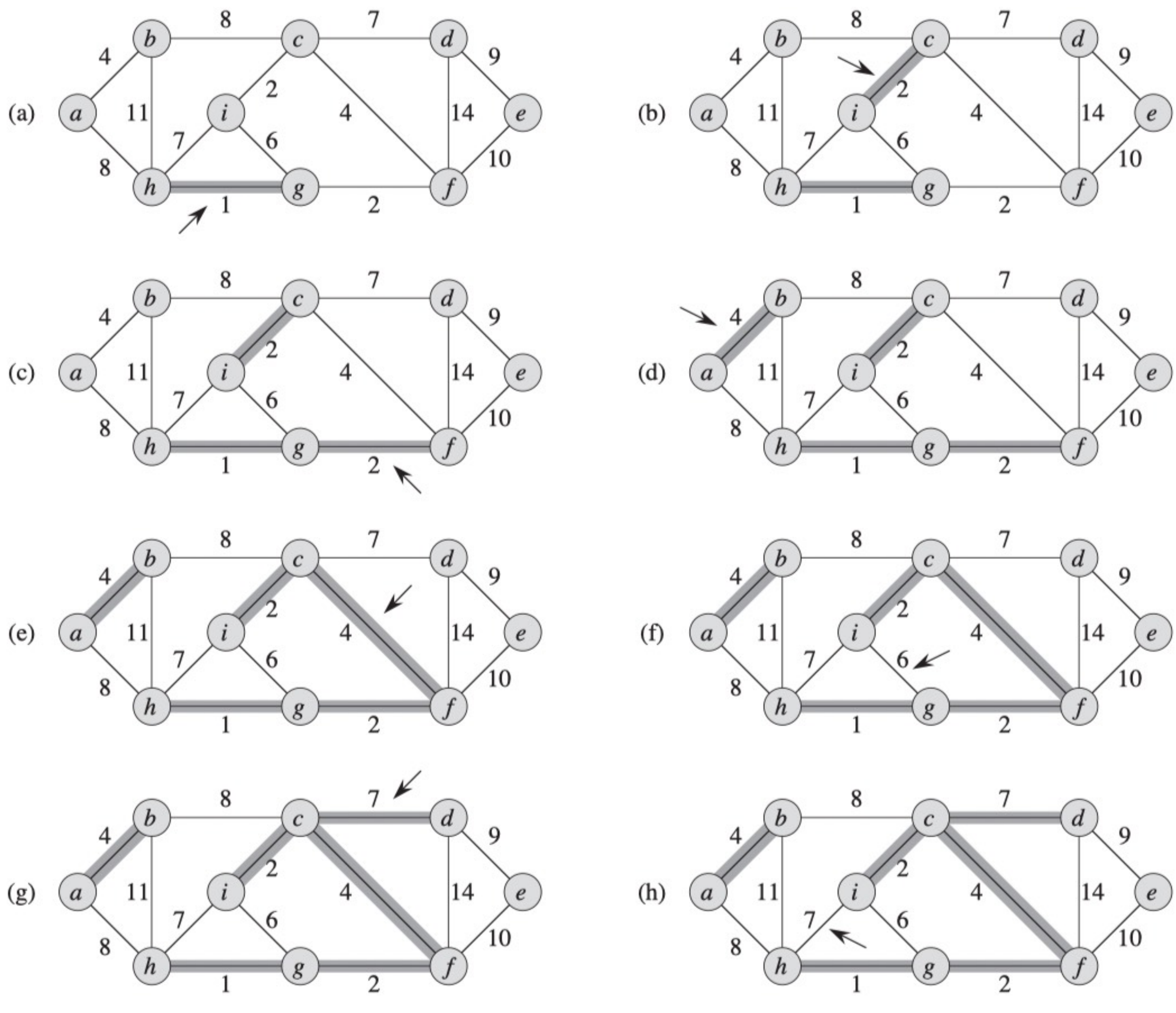
Prior to each iteration, A is a subset of some minimum spanning tree.

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.

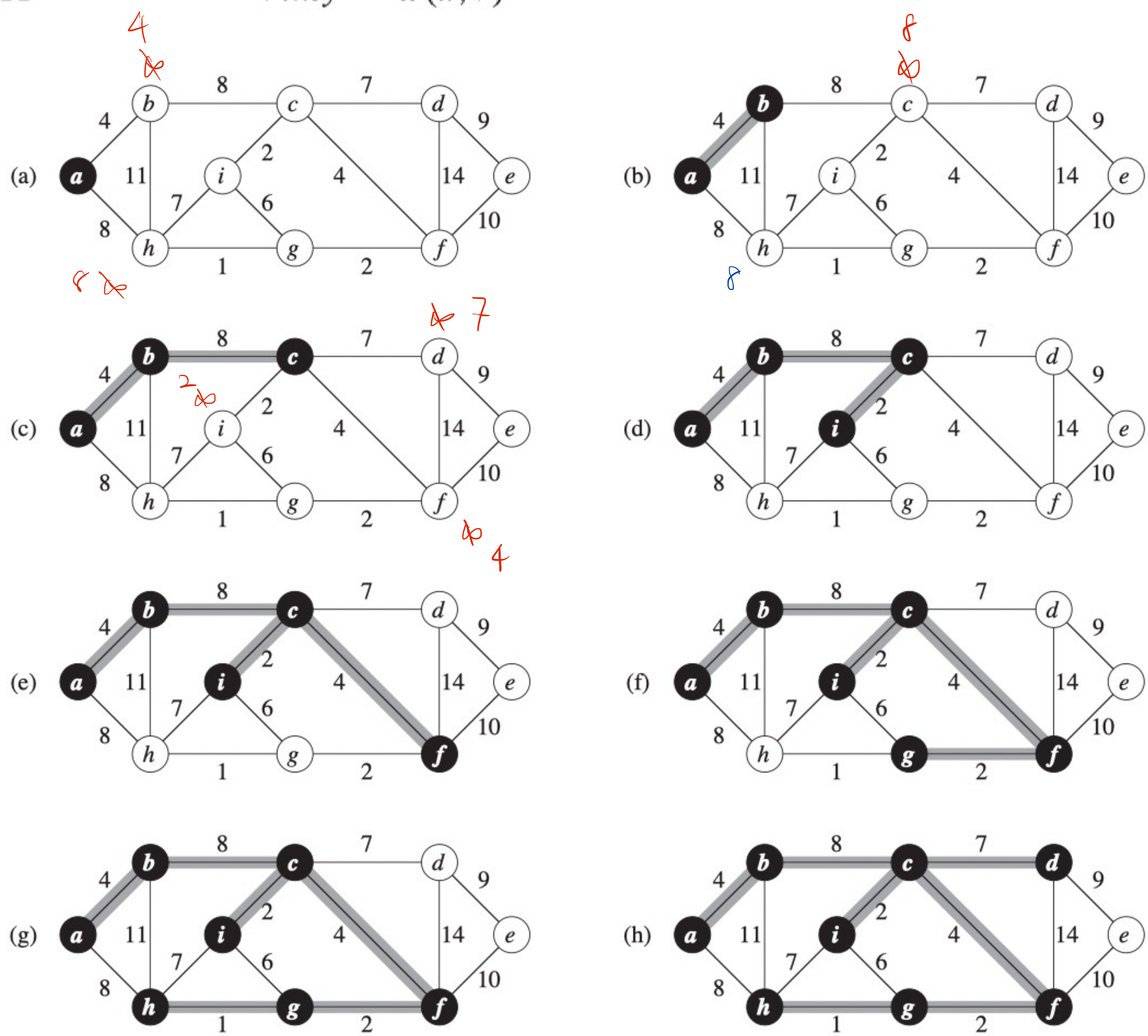
In Prim's algorithm, the set A forms a single tree. The safe edge added to A is always a least-weight edge connecting the tree to a vertex not in the tree.

[1]. Intro to algorithm., chapter 23

```
MST-KRUSKAL(G, w)
A = ∅
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```



```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```

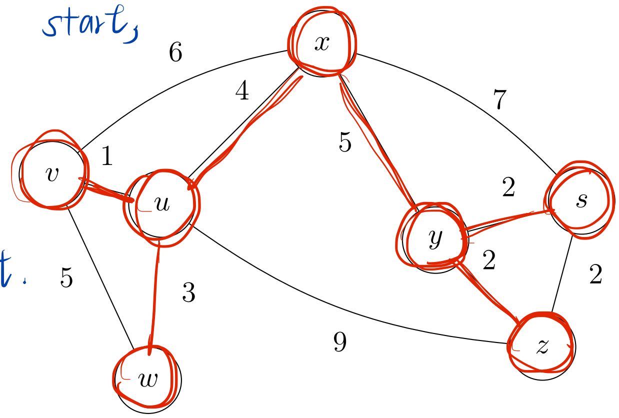


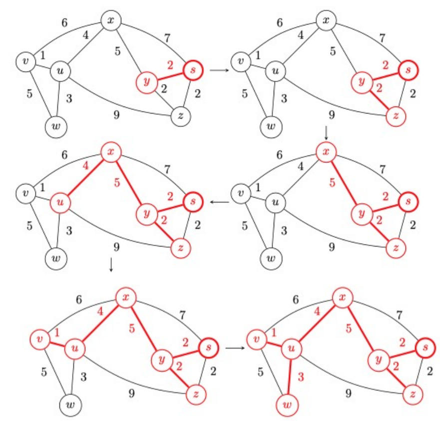
Problem 1

Prim's and Kruskal's are two algorithms for finding a Minimum Spanning Tree on a graph. In this problem, you will practice

a. Use Prim's Algorithm to construct a MST on the following graph, and state the total weight.

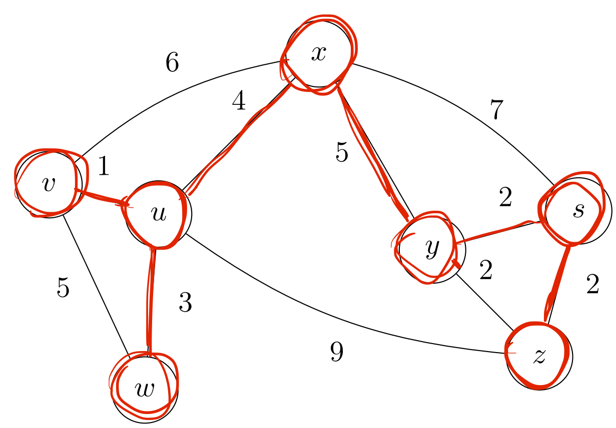
We choose y to start, but you can choose arbitrary vertex to start.

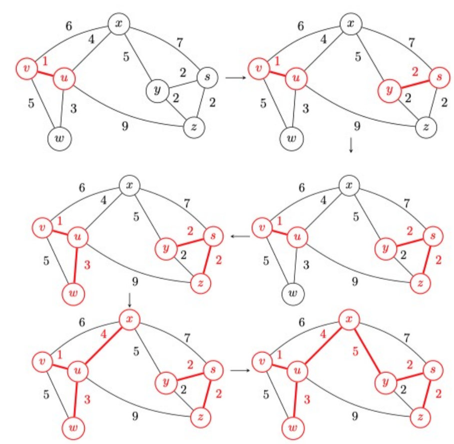




The total weight is 17.

b. Use Kruskal's Algorithm to find a MST on the same graph:

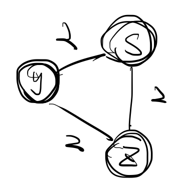




The total weight is 17.

c. Are the MSTs produced by the two algorithms the same? Why or why not? Are there any other MSTs?

They are not the same, the edges chosen from the s, y, z triangle differ. Any two of these edges can be included to make a different, valid MST.



When a graph has edges of the same weight, it can have multiple MSTs, but having multiple edges of the same weight is not a guarantee that there exist multiple MSTs (even if some of those edges are used in MSTs).

Problem 2

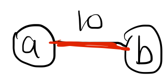
For each of the following statements about MSTs on an undirected graph $G = (V, E)$ with (not necessarily distinct) edge weights $w_e > 0$ for each $e \in E$, either prove the statement or disprove it with a counterexample.

a. Assume there is an edge e^* which has a strictly smaller weight than all other edges in E. Then e^* is in every minimum spanning tree of G.

This is true. We argue by contradiction. Assume there is a minimum spanning tree T that does not contain e∗ . Then T ∪ { e∗ } contains a cycle C, and e ∗ is not the most expensive edge in C. Remove the most expensive edge e′ in C. This new graph T ∪ { e∗ } \ { e′ } costs less than T, and is also a spanning tree. This contradicts the assumption that T was an MST.

b. Assume there is an edge e^* which has a strictly larger weight than all other edges in E. Then e^* is not in any minimum spanning tree of G.

This is false. Consider the graph with two vertices and one edge between them: this edge is the only MST, and is also trivially the largest-weight edge.

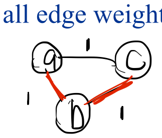


c. If an edge e is in a minimum spanning tree T of G, then there exists some cut $(S, V \setminus S)$ such that e is the (not necessarily unique) cheapest edge crossing $(S, V \setminus S)$.

This is true. Let T be a MST containing e. Remove e separates T into two connected components, which form a cut, (A, B). Assume there was some edge e′ which crossed (A, B) satisfying w(e′) < w(e) . Then swapping e′ for e would produce a spanning tree with smaller cost than T, which contradicts the assumption that T was an MST. This implies that no such edge e′ can exist.

d. For any minimum spanning tree T and pair of vertices u and v, T contains a (weighted) shortest path from u to v in G.

This is false. Consider C 3 , the cycle on three vertices a, b, and c, with all edge weights 1. The MST { (a, b), (b, c) } does nor contain the shortest path from a to c.



e. Note that in a graph with nondistinct edge weights, there may be multiple minimum-cost spanning trees. Let $T \subseteq E$ be a spanning tree. If each edge $e \in T$ belongs to *some* MST in G, then T is also a MST in G.

This is false. Consider the cycle graph C 3 , the cycle on three vertices a, b, and c, with edge weights w (a,b) = 2, w (a,c) = 2, and w (b,c) = 1. { (a, b), (b, c) } and { (a, c), (b, c) } are both MSTs with weight 3, but the spanning tree { (a, b), (a, c) } has weight 4.

Jk