* Kruskal’s Algorithm Implementation:
  + How to check if adding an edge to T would create a cycle in T?
    - Answer 1: Simple (naïve solution). Use Depth-First-Search (DFS).
      * Use DFS. If you come to a node that has already been visited, you know it is a cycle.
      * DFS itself takes O(V+E). Therefore, it takes O(V) per cycle check.
        + Can also be written as O(V+E1). The E1 is dropped because the V is bigger.
        + E1 is the number of edges in the MST, which is a subset of the input graph.
      * In the worst case, you have to check each edge in the sorted edge array for Kruskal’s Algorithm.
      * The overall time for Kruskal’s Algorithm is: O(E2\*V), where E2 is the number of edges in the original input graph.
      * O(n2)
        + Note that in this case, the number of edges and vertices in the original input graph will be roughly equal. Therefore, O(E\*V) can be expressed as O(n).
    - Answer 2: Use the union-find (UF) data structure.
      * Maintain a set for each connected component.
      * If v and w are in the same component already, then adding e(v, w) creates a cycle.
      * If not creating a cycle, to add e(v, w) to T, you merge the sets containing v and w.
      * Note that e(v,w) represents the edge between v and w.
      * Public class UF {
        + HashSet<Integer> components[];

An array of Hashsets, where each HashSet represents a set of vertices that have some connection to each other in the MST. In Tony’s lecture, he described each HashSet as being like an island.

* + - * + boolean find (int v, int w)

Checks whether v and w belong to the same HashSet.

* + - * + void unite (int v, int w)

Merges the HashSets that contain v and w.

* + - * + } // End UF class.
      * Public class Kruskal {
        + Private Set<Edge> mst = new HashSet<Edge>();
        + Public Kruskal (WeightedGraph G) {

Edge[] edges = G.edges();

Arrays.sort(edges, Edge.BY\_WEIGHT);

UF uf = new UF(G.V());

For (Edge e : edges) {

If ( ! uf.find(e.either(), e.other()) ) {

mst.add(e);

uf.unite(e.either(), e.other());

} // End class