Minimum Spanning Trees

Problem: Given a graph $G = (V, E)$, find an <u>acyclic</u> subset T (subset of E) that connects all the vertices, and whose total weight is <u>minimized</u>.

This tree "spans" the graph

Thus the 'minimum spanning tree' term ☺

We will solve this using a "Greedy" Algorithm - one that makes makes decisions based on whatever looks best at the moment

```
Kruskal's Algorithm ( Graph G)
{   Set A = new Set();// empty this will contain the min. spanning
    [redacted]                    // tree when we're done
    SetCollection allSets = new SetCollection();// add, find, remove
    foreach ( Vertex v in G.Vertices)
        allSets.add ( new Set(v));  //contains just "v" to start
    PriorityQueue pq = new PriorityQueue();// MinHeap
    foreach ( Edge e in G.Edges)
        pq.add(e);
    pq.BuildHeap();
    while(!pq.isEmpty())
    {   Edge e = pq.getMin();// assume this removes e & re-heapifies
        Set from = allSets.findSetContaining(e.start);
        Set to = allSets.findSetContaining(e.end);
        if( from != to)// in real C#, you may want !from.equals(to)
        {   A.add(e);
            allSets.remove(from); allSets.remove(to);
            allSets.add ( from.union(to));// 'to' is discarded
}   }   }
```

Implementation Details

'Set' is a <u>collection of things</u>, that has <u>no duplicates</u>
('Bag' is a collection of things, that <u>does</u> allow duplicates)
Examples of sets = {1, 10, -10} {Apple, Orange} {Edge#1, Edge#2}
" " nonsets = {1, 2, 1} {Apple, Apple, PC}

2 Strategies:
(A) No support from objects we're storing
↳ Each set can be a hash table (or AVL tree, or skip list)
so we can look up individual members quickly)
(B) Objects will include data fields to help the set out
↳ Give each object a 'ContainingSet' reference so we can (from the object) figure out which set it belongs to in O(1) time
↳ This works great.... if each object is in exactly 1 set

"Union" operation: // Merge 1 set into 'this' one

```
Set.Union(Set other)
{   for each (Object item in other)
    {   this.add(item);   }
}
```