Minimum Spanning Trees

- Spanning tree
  - Problem of connecting pins in an electronic circuit with wires
  - n pins can be connected with an arrangement of $n-1$ wires
  - We'll like to use an arrangement that minimizes the amount of wire used
  - Defined as a *connected graph* $G = \{V, E\}$ such that $|V| = |E| + 1$
- Problem instance
  - Given a complete graph $G = (V, E)$ where V is the set of pins and $E = w(u, v)$ is the set of *possible interconnections* between each pair of pins
  - $w(u, v)$ is the weight of each edge, or cost of connecting the wire between pins u and v
- Minimum spanning tree
  - Find an acyclic subset $T \subseteq E$ such that

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

    is minimized
  - T is acyclic and contains all nodes in V, hence forms a *spanning* tree
  - Problem to determine T is known as the *minimum spanning tree problem*
- Two minimum spanning tree algorithms based on greedy strategy
  1. Kruskal's algorithm
  2. Prim's algorithm
  - Both algorithms can be made to run in time $O(E \lg V)$
  - Both algorithms are based on greedy strategy
    * Make the choice that *appears to be the best* at the moment
    * In general, not guaranteed to find globally optimal solutions to the problem
    * For MST, it can be proved that greedy strategies do yield a spanning tree with minimum weight

Growing a minimum spanning tree

- Given a connected, undirected graph $G = (V, E)$ with a weight function $w : E \rightarrow R$
- Grow MST one edge at a time
- Manage a set A that is always a subset of some minimum spanning tree
- At each step, an edge (u, v) is determined such that $A \cup (u, v)$ is also a subset of a minimum spanning tree
  - (u, v) is called a *safe edge*

– (u, v) is *safe* iff it does not create a cycle with the edges selected so far in set A

- Consider the following graph:

	a	b	c	d	e	f	g	h	i
a	-	4						8	
b	4	-	8					11	
c		8	-	7		4			2
d			7	-	9	14			
e				9	-	10			
f			4	14	10	-	2		
g						2	-	1	6
h	8	11					1	-	7
i			2				6	7	-

- 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
```

  – Tricky part is to find a safe edge inside the `while` loop
  – One must exist since there is a spanning tree T such that $A \subseteq T$
  – Within the body of `while` loop, $A \subset T$ and hence, there must be an edge $(u, v) \in T$ such that $(u, v) \notin A$ and (u, v) is safe for A

- A **cut** $(S, V - S)$ of an undirected graph $G = (V, E)$ is a partition of V
- An edge $(u, v) \in E$ **crosses** the cut $(S, V - S)$ if one of its endpoints is in S while the other is in $V - S$
- **Kruskal's Algorithm**
  – MST_Kruskal (G,w)

```
A ←
for each vertex v ∈ V[G] do
   make_set(v)
sort the edges of E by nondecreasing weight w
for each edge (u,v) ∈ E, in order by nondecreasing weight, do
   if find_set(u) ≠ find_set(v) then
      A ← A ∪ {(u,v)}
      union(u,v)
return A
```

  – A simple explanation of the algorithm is:
    * Consider each node to be a tree by itself
    * Build a heap of edges, with least cost edge at the root
    * While we have less than $|V| - 1$ edges
      · Extract the least cost edge
      · If it creates a cycle with the edges selected so far, ignore it, and continue the loop
      · Add the edge to the set