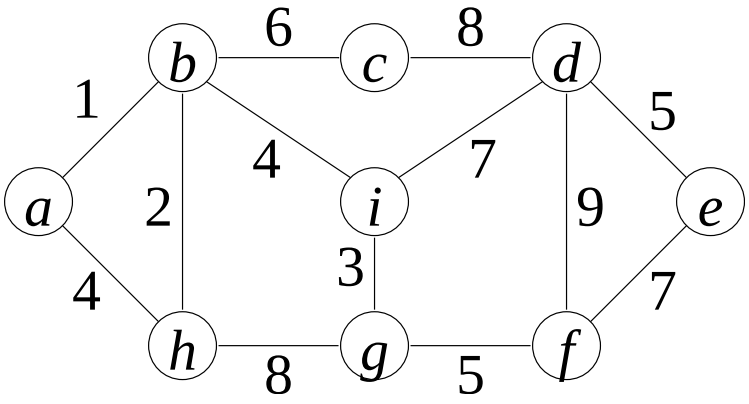
Weighted Graphs

A weighted graph is a graph in which each edge (u, v) has a weight $w(u, v)$. Each weight is a real number. Weights can represent distance, cost, time, capacity, etc.



Optimization Problems for Weighted Graphs:
Find a minimum spanning tree.
Find shortest paths between vertices.
Maximize flow from a source to a sink.
Traveling salesman problem.

Variations on Problems:
Undirected/directed graphs
Negative weights

Minimum Spanning Trees

A spanning tree T of a graph G is a subset T of the edges that form a (free) tree. If G has V vertices, then T has $V - 1$ edges.

A minimum spanning tree (MST) is a spanning tree with a minimum sum of weights.

Consider any partition of vertices V_1 and V_2.
Let E' be the edges between V_1 and V_2.

Theorem: MST has a min-wgt. edge from E'.

Proof:
Suppose T is a MST with no such edge.
Let (u, v) be a min-wgt. edge from E'.
T has a path between u and v.
Some edge (x, y) on this path is in E'.
Replacing (x, y) with (u, v) is better.

Kruskal's Algorithm

MST-KRUSKAL(G, w)
  $T \leftarrow \emptyset$
  for each vertex v in G
    do MAKE-SET(v)
  for each edge (u, v) in ascending order
    do if FIND-SET$(u) \neq$ FIND-SET(v)
        then $T \leftarrow T \cup \{(u, v)\}$
            UNION(u, v)
            exit loop if all vertices are unioned
  return T

Proof of correctness: (u, v) is a minimum weight edge between u's set and other vertices.

Use disjoint-set forest for union-find, almost $O(E)$.
Use $O(E \lg E)$ sorting algorithm or binary heap.
MST-KRUSKAL is $O(E \lg E) = O(E \lg V)$.

Prim's Algorithm

MST-PRIM(G, w, r)
  for each vertex v in G **do** $key[v] \leftarrow \infty$
  $key[r] \leftarrow 0$, $p[r] \leftarrow$ NIL
  $Q \leftarrow$ vertices of G
  while Q is not empty
    do $u \leftarrow$ EXTRACT-MIN(Q)
      for each v adjacent from u
        do if $v \in Q$ and $w(u, v) < key[v]$
          then $p[v] \leftarrow u$
            DECREASE-KEY$(Q, v, w(u, v))$

Proof of correctness: $(p[u], u)$ is a minimum weight edge between Q and $V - Q$.

Use binary heap for priority queue, $O(V)$ elts.
$O(V)$ EXTRACT-MINs is $O(V \lg V)$.
$O(E)$ DECREASE-KEYs is $O(E \lg V)$.
MST-PRIM is $O(E \lg V) = O(E \lg E)$.
Using Fibonocci heaps makes it $O(E + V \lg V)$.