Announcement

- Hw3 will be due this Thursday.
Topics

- Shortest paths
  - (Dijkstra’s algorithm)
- Minimum Spanning Tree
  - Prim-Jarnik Algorithm
  - Kruskal Algorithm

Shortest Path Problem

Given a weighted graph and two vertices \( u \) and \( v \), we want to find a path of minimum total weight between \( u \) and \( v \).
- Length of a path is the sum of the weights of its edges.

Example:
- Shortest path between Providence and Honolulu

Applications
- Internet packet routing
- Flight reservations
- Driving directions
Dijkstra's Algorithm

- The distance of a vertex $v$ from a vertex $s$ is the length of a shortest path between $s$ and $v$.
- Dijkstra's algorithm computes the distances of all the vertices from a given start vertex $s$.
- Assumptions:
  - the graph is connected
  - the edges are undirected
  - the edge weights are nonnegative
- We grow a "cloud" of vertices, beginning with $s$ and eventually covering all the vertices.
- We store with each vertex $v$ a label $d(v)$ representing the distance of $v$ from $s$ in the subgraph consisting of the cloud and its adjacent vertices.
- At each step:
  - We add to the cloud the vertex $u$ outside the cloud with the smallest distance label, $d(u)$.
  - We update the labels of the vertices adjacent to $u$.

Example
Example (cont.)

Minimum Spanning Tree

- Spanning subgraph
  - Subgraph of a graph $G$ containing all the vertices of $G$

- Spanning tree
  - Spanning subgraph that is itself a (free) tree

- Minimum spanning tree (MST)
  - Spanning tree of a weighted graph with minimum total edge weight

Applications
- Communications networks
- Transportation networks
Partition Property

Partition Property:
- Consider a partition of the vertices of G into subsets U and V.
- Let e be an edge of minimum weight across the partition.
- There is a minimum spanning tree of G containing edge e.

Proof:
- Let T be an MST of G.
- If T does not contain e, consider the cycle C formed by e with T and let f be an edge of C across the partition.
- By the cycle property, weight(f) ≥ weight(e).
- Thus, weight(f) = weight(e).
- We obtain another MST by replacing f with e.

Prim-Jarnik's Algorithm

- Similar to Dijkstra's algorithm (for a connected graph).
- We pick an arbitrary vertex s and we grow the MST as a cloud of vertices, starting from s.
- We store with each vertex v a label d(v) = the smallest weight of an edge connecting v to a vertex in the cloud.

At each step:
- We add to the cloud the vertex u outside the cloud with the smallest distance label.
- We update the labels of the vertices adjacent to u.
At each step:
- We add to the cloud the vertex not in the cloud with the smallest distance label.
- We update the labels of the vertices adjacent to a.
Kruskal’s Algorithm

- A priority queue stores the edges outside the cloud
  - Key: weight
  - Element: edge
- At the end of the algorithm
  - We are left with one cloud that encompasses the MST
  - A tree $T$ which is our MST

Algorithm $\text{KruskalMST}(G)$

for each vertex $v$ in $G$ do
  define $\text{Cloud}(v)$ of $\in\langle v \rangle$
  let $Q$ be a priority queue, insert all edges into $Q$ using their weights as the key
  $T \subseteq Q$
  while $T$ has fewer than $n-1$ edges do
    edge $e = T.\text{removeMin}()$
    let $u, v$ be the endpoints of $e$
    if $\text{Cloud}(u) = \text{Cloud}(v)$ then
      $T$ is not a tree
      return $T$
    else
      Add edge $e$ to $T$
      Merge Cloud($v$) and Cloud($w$)
  return $T$