Graph Optimization Problems and Greedy Algorithms

- Greedy Algorithms
  - **// Make the best choice now!**
- Optimization Problems
  - Minimizing Cost or Maximizing Benefits
  - **Minimum Spanning Tree**
    - Minimum cost for connecting all vertices
  - **Single-Source Shortest Paths**
    - Shortest Path between two vertices

TECH
Computer Science

Greedy Algorithms: Make the best choice now!

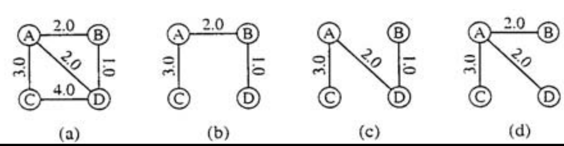
- Making choices in sequence such that
  - **each individual choice is best**
    - according to some limited "short-term" criterion,
    - that is not too expensive to evaluate
  - **once a choice is made, it cannot be undone!**
    - even if it becomes evident later that it was a poor choice
- Make progress by choosing an action that
  - **incurs the minimum short-term cost,**
  - **in the hope that a lot of small short-term costs add up to small overall cost.**
- Possible drawback:
  - **actions with a small short-term cost may lead to a situation, where further large costs are unavoidable.**

Optimization Problems

- Minimizing the total cost or Maximizing the total benefits
  - **Analyze all possible outcomes and find the best, or**
  - **Make a series of choices whose overall effect is to achieve the optimal.**
- Some optimization problems can be solved *exactly* by greedy algorithms
  - **Minimum cost for connecting all vertices**
    - Minimum Spanning Tree Algorithm
  - **Shortest Path between two vertices**
    - Single-Source Shortest Paths Algorithm

Minimum Spanning Tree

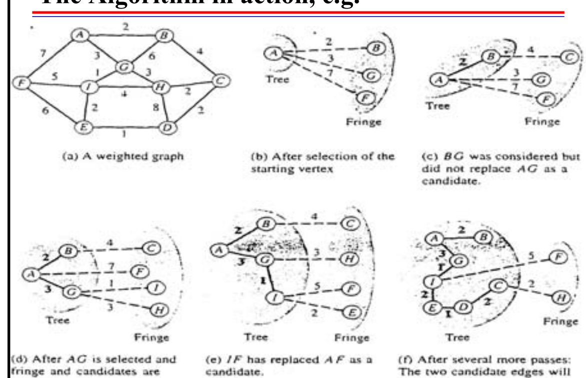
- **A *spanning tree* for a connected, undirected graph, G=(V,E) is**
  - a subgraph of G that is
  - an undirected tree and contains
  - all the vertices of G.
- **In a weighted graph G=(V,E,W), the** weight of a subgraph **is**
  - the sum of the weights of the edges in the subgraph.
- **A *minimum spanning tree* for a weighted graph is**
  - a spanning tree with the minimum weight.



Prim's Minimum Spanning Tree Algorithm

- Select an arbitrary starting vertex, (the root)
- branches out from the tree constructed so far by
  - **choosing an edge at each iteration**
  - **attach the edge to the tree**
    - that edge has minimum weight among all edges that can be attached
  - **add to the tree the vertex associated with the edge**
- During the course of the algorithm, vertices are divided into three disjoint categories:
  - **Tree vertices: in the tree constructed so far,**
  - **Fringe vertices: not in the tree, but adjacent to some vertex in the tree,**
  - **Unseen vertices: all others**

The Algorithm in action, e.g.



Prim's Minimum Spanning Trees: Outline

primMST(G, n) // *OUTLINE*

Initialize all vertices as *unseen*.

Select an arbitrary vertex s to start the tree; reclassify it as *tree*.

Reclassify all vertices adjacent to s as *fringe*.

While there are fringe vertices:

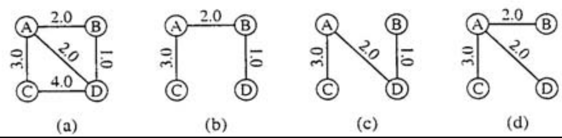
Select an edge of minimum weight between a tree vertex t and a fringe vertex v;

Reclassify v as *tree*; add edge tv to the tree;

Reclassify all *unseen* vertices adjacent to v as *fringe*.

Properties of Minimum Spanning Trees

- Definition: Minimum spanning tree property
  - Let a connected,weighted graph G=(V,E,W) be given,and let T be any spanning tree of G.
  - Suppose that for every edge vw of G that is **not** in T,
  - **if** uv is added to T, **then** it creates a cycle
  - **such that** uv is a maximum-weight edge on that cycle.
  - The the tree T is said to have the *minimum spanning tree property*.



Properties of Minimum Spanning Trees …

- Lemma:
  - **In a connected, weighted graph G = (V, E, W),**
  - **if T1 and T2 are two spanning trees that have the MST property,**
  - **then they have the same total weight.**
- **Theorem:**
  - **In a connected,weighted graph G=(V,E,W)**
  - **a tree T is a minimum spanning tree *if and only if***
  - **T has the MST property.**

Correctness of Prim's MST Algorithm

- Lemma:
  - **Let G = (V, E, W) be a connected, weighted graph with n = |V|;**
  - **let T_k be the tree with k vertices constructed by Prim's algorithm, for k = 1, …, n; and**
  - **let G_k be the subgraph of G induced by the vertices of T_k (i.e., uv is an edge in G_k if it is an edge in G and both u and v are in T_k).**
  - **Then T_k has the MST property in G_k.**
- **Theorem:**
  - Prim's algorithm outputs a minimum spanning tree.

Problem: Single-Source Shortest Paths

- Problem:
  - **Finding a minimum-weight path between two specified vertices**
  - **It turns out that, in the worst case, it is *no* easier to find a minimum-weight path between a specified pair of nodes s and t *than***
  - **it is to find minimum-weight path between s and every vertex reachable from s. (single-source shortest paths)**

Shortest-Path

- **Definition: shortest path**
  - Let P be a nonempty path
  - in a weighted graph G=(V,E,W)
  - consisting of k edges $xv_1, v_1v_2, \dots v_{k-1}y$ (possibly v_1=y).
  - The *weight* of P, denoted as W(P) is
  - the sum of the weights, $W(xv_1), W(v_1v_2), \dots W(v_{k-1}y)$.
  - If x=y, the empty path is considered to be a path from x to y. The weight of the empty path is zero.
  - If no path between x and y has weight less than W(P),
  - then P is called a ***shortest path***,or minimum-weight path.

Properties of Shortest Paths

- Lemma: Shortest path property
  - **In a weighted graph G,**
  - **suppose that a shortest path from x to z consist of**
  - **path P from x to y followed by**
  - **path Q from y to z.**
  - ***Then* P is a shortest path from x to y, and**
  - **Q is a shortest path form y to z.**

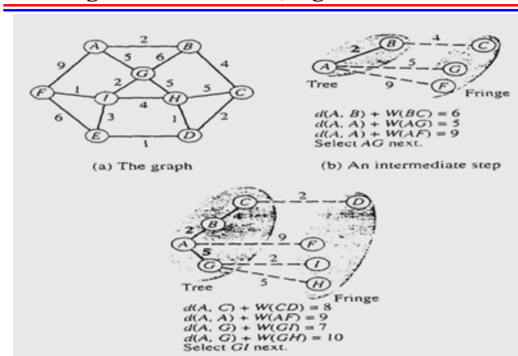
Dijkstra's Shortest-Path Algorithm

- Greedy Algorithm
- weights are nonnegative

dijkstraSSSP(G, n) // *OUTLINE*

Initialize all vertices as *unseen*.
Start the tree with the specified source vertex s; reclassify it as *tree*;
define $d(s, s) = 0$.
Reclassify all vertices adjacent to s as *fringe*.
While there are fringe vertices:
  Select an edge between a tree vertex t and a fringe vertex v such that $(d(s,t) + W(tv))$ is minimum;
  Reclassify v as *tree*; add edge tv to the tree;
  define $d(s, v) = (d(s,t) + W(tv))$.
  Reclassify all *unseen* vertices adjacent to v as *fringe*.

The Algorithm in action, e.g.



(a) The graph
(b) An intermediate step
(c) An intermediate step: CH was considered, but not chosen, to replace GH as a candidate.

Correctness of Dijkstra's Shortest-Path Algorithm

- **Theorem:**
  - Let G=(V,E,W) be a weighted graph with nonnegative weights.
  - Let V' be a subset of V and
  - let s be a member of V'.
  - Assume that d(s,y) is the shortest distance in G from s to y, for each $y \in V'$.
  - **If** edge yz is chosen to minimize d(s,y)+W(yz) over **all edges** with one vertex y in V' and one vertex z in V-V',
  - **then** the path consisting of a shortest path from s to y followed by the edge yz is a shortest path from s to z.
- **Theorem:**
  - Given a directed weighted graph G with a nonnegative weights and a source vertex s, Dijkstra's algorithm computes the shortest distance from s to each vertex of G that is reachable from s.