Greedy Technique

Constructs a solution to an *optimization problem* piece by piece through a sequence of choices that are:

- *feasible*
- *locally optimal*
- *irrevocable*

For some problems, yields an optimal solution for every instance.
For most, does not but can be useful for fast approximations.

Applications of the Greedy Strategy

- **Optimal solutions:**
  - change making for "normal" coin denominations
  - minimum spanning tree (MST)
  - single-source shortest paths
  - simple scheduling problems
  - Huffman codes

- **Approximations:**
  - traveling salesman problem (TSP)
  - knapsack problem
  - other combinatorial optimization problems

Change-Making Problem

Given unlimited amounts of coins of denominations $d_1 > \ldots > d_m$, give change for amount n with the least number of coins

Example: $d_1 = 25c$, $d_2 = 10c$, $d_3 = 5c$, $d_4 = 1c$ and $n = 48c$

Greedy solution:

Greedy solution is

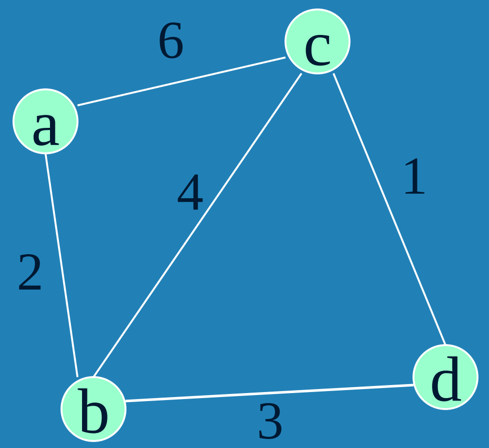
- **optimal for any amount and "normal" set of denominations**
- **may not be optimal for arbitrary coin denominations**

Minimum Spanning Tree (MST)

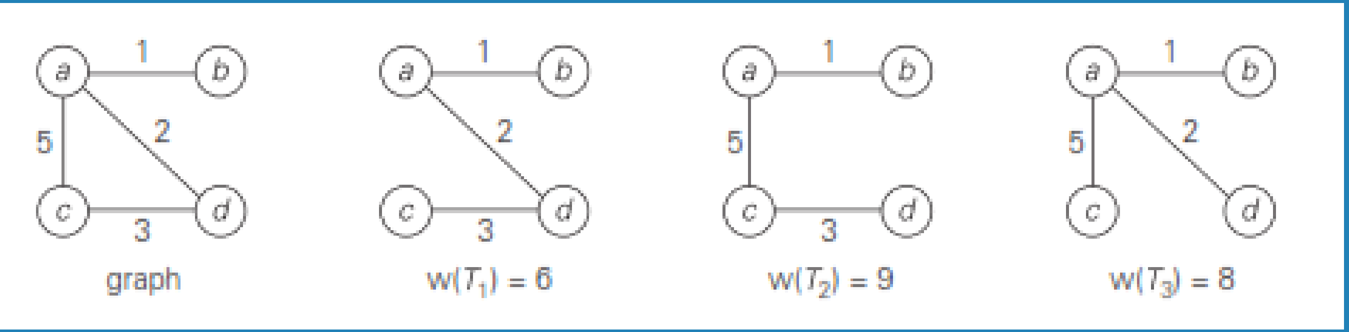
- ***Spanning tree*** of a connected graph *G*: a connected acyclic subgraph of *G* that includes all of *G*'s vertices

- ***Minimum spanning tree*** of a weighted, connected graph *G*: a spanning tree of *G* of minimum total weight

Example:



MST



- **Robert Prim rediscovered the algorithm published 27 years earlier by the Czech mathematician Vojtech Jarnik in a Czech journal.**

- DEFINITION Aspanning tree of an undirected connected graph is its connected acyclic subgraph (i.e., a tree) that contains all the vertices of the graph. If such a graph has weights assigned to its edges, a minimum spanning tree is its spanning tree of the smallest weight, where the weight of a tree is defined as the sum of the weights on all its edges. The minimum spanning tree problem is the problem of finding a minimum spanning tree for a given weighted connected graph.

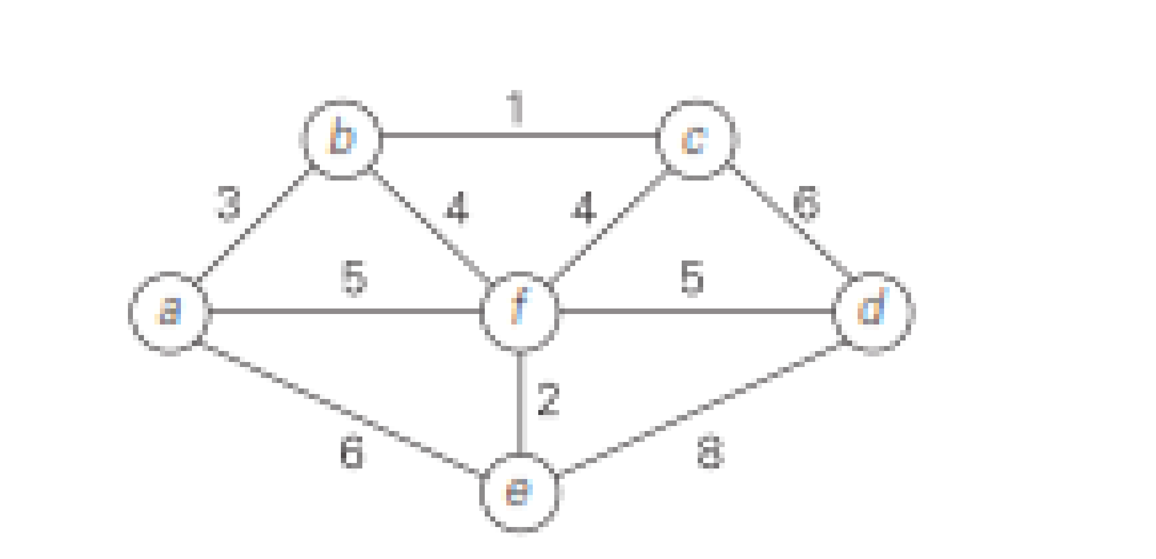
ALGORITHM *Prim(G)*

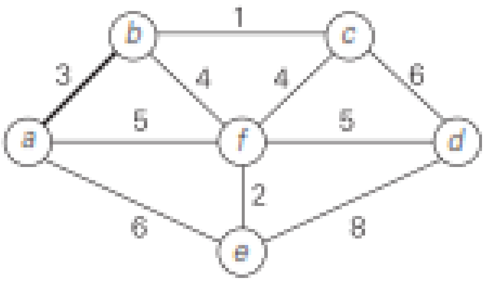
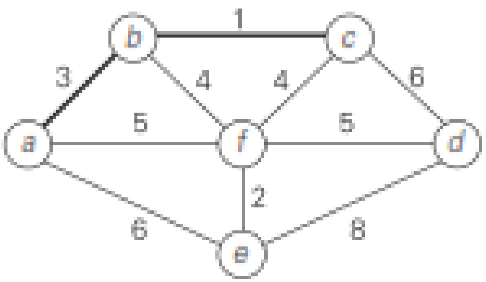
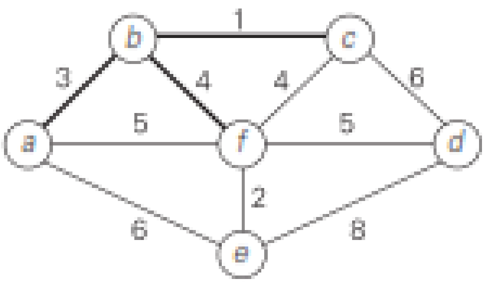
//Prim's algorithm for constructing a minimum spanning tree
//Input: A weighted connected graph $G = \langle V, E \rangle$
//Output: E_T, the set of edges composing a minimum spanning tree of G

$V_T \leftarrow \{v_0\}$ //the set of tree vertices can be initialized with any vertex
$E_T \leftarrow \varnothing$
for $i \leftarrow 1$ **to** $|V| - 1$ **do**
    find a minimum-weight edge $e^* = (v^*, u^*)$ among all the edges (v, u)
    such that v is in V_T and u is in $V - V_T$
    $V_T \leftarrow V_T \cup \{u^*\}$
    $E_T \leftarrow E_T \cup \{e^*\}$
return E_T

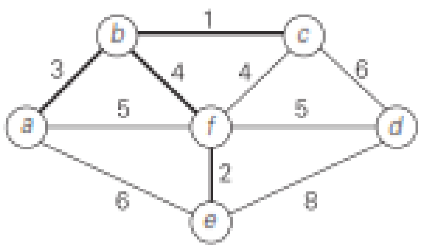
Prim's MST algorithm

- Start with tree T_1 consisting of one (any) vertex and "grow" tree one vertex at a time to produce MST through a series of expanding subtrees $T_1, T_2, \ldots, T_n$
- On each iteration, construct T_{i+1} from T_i by adding vertex not in T_i that is closest to those already in T_i (this is a "greedy" step!)
- Stop when all vertices are included

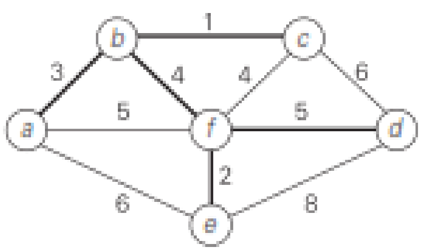

b
1
c
3
4
4
6
5
5
a
f
d
2
6
8
e

Tree vertices	Remaining vertices	Illustration
a(−, −)	**b(a, 3)** c(−, ∞) d(−, ∞) e(a, 6) f(a, 5)	
b(a, 3)	**c(b, 1)** d(−, ∞) e(a, 6) f(b, 4)	
c(b, 1)	d(c, 6) e(a, 6) **f(b, 4)**	

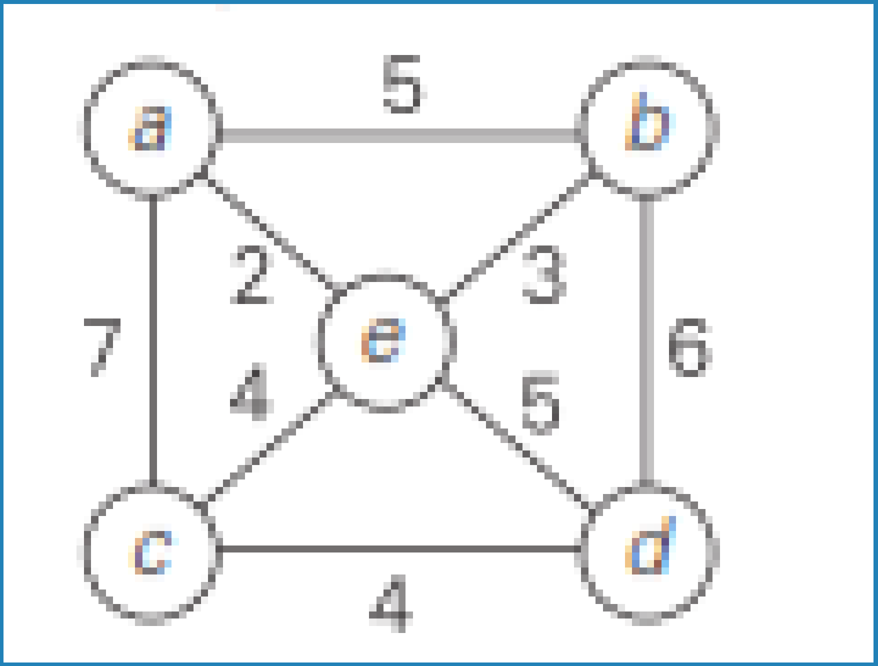
f(b, 4) d(f, 5) **e(f, 2)**



e(f, 2) **d(f, 5)**

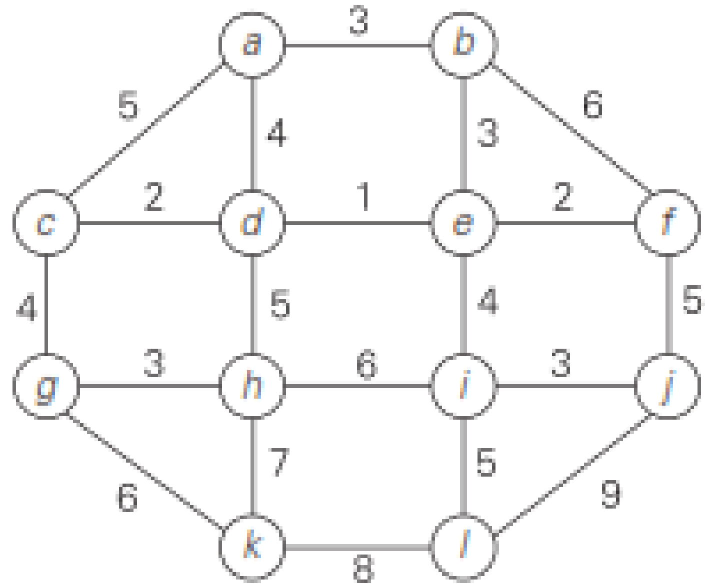


d(f, 5)

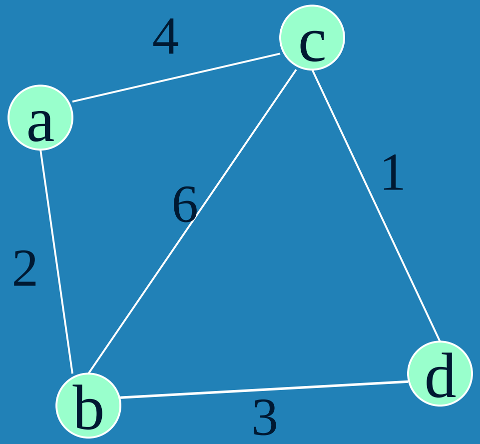

a
b
c
d
e
5
2
3
7
6
4
5
4

Tree vertices	Priority queue of remaining vertices
a(-,-)	b(a,5) c(a,7) d(a,∞) **e(a,2)**
e(a,2)	**b(e,3)** c(e,4) d(e,5)
b(e,3)	**c(e,4)** d(e,5)
c(e,4)	**d(c,4)**
d(c,4)	

The minimum spanning tree found by the algorithm comprises the edges *ae*, *eb*, *ec*, and *cd*.


a
b
c
d
e
f
g
h
i
j
k
l
3
5
4
3
6
2
1
2
4
5
4
5
3
6
3
6
7
5
9
8

Example



Notes about Prim's algorithm

- Proof by induction that this construction actually yields MST
- Needs priority queue for locating closest fringe vertex
- Efficiency
  - O(n^2) for weight matrix representation of graph and array implementation of priority queue
  - O(m log n) for adjacency list representation of graph with n vertices and m edges and min-heap implementation of priority queue

Another greedy algorithm for MST: Kruskal's

- Sort the edges in nondecreasing order of lengths
- "Grow" tree one edge at a time to produce MST through a series of expanding forests F_1, F_2, …, F_{n-1}
- On each iteration, add the next edge on the sorted list unless this would create a cycle. (If it would, skip the edge.)