11.5 *Minimum Spanning Trees*

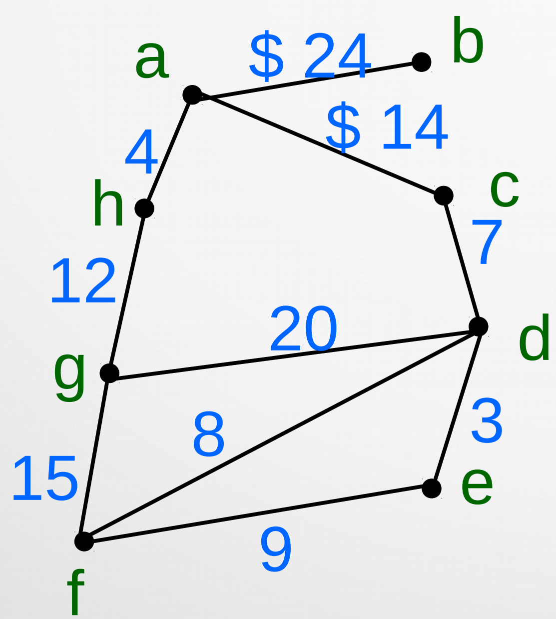
Let's consider graphs with weighted edges.

[Def] A *minimum spanning tree* in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

11.5 *Minimum Spanning Trees*

Let's consider graphs with weighted edges.

[Def] A *minimum spanning tree* in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

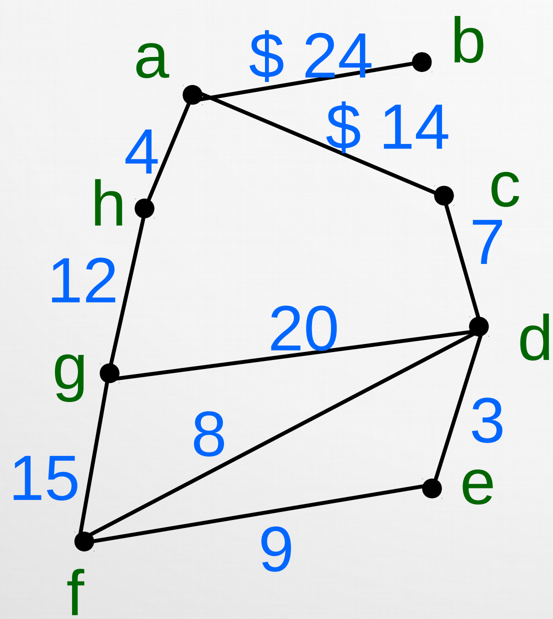


What is the cheapest path from city f to city a?

11.5 *Minimum Spanning Trees*

Let's consider graphs with weighted edges.

[Def] A *minimum spanning tree* in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.



What is the cheapest path from city f to city a?

$15+$12+$4 = $31

$9+$3+$7+$14 = $33

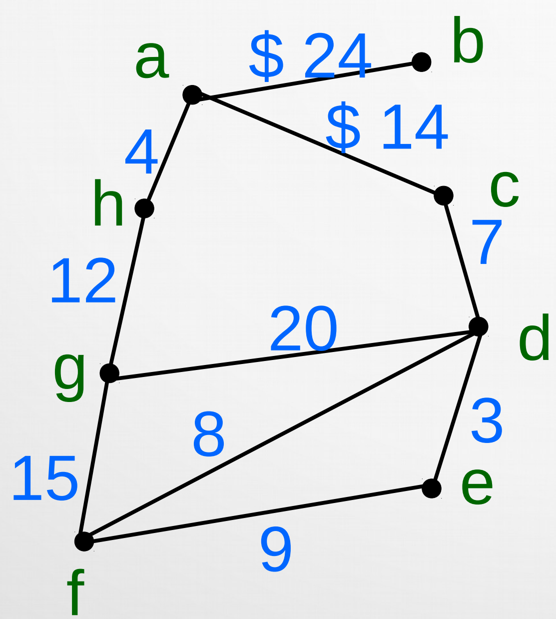
$8+$7+$14 = $29

...

11.5 *Minimum Spanning Trees*

Let's consider graphs with weighted edges.

[Def] A *minimum spanning tree* in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.



What is the cheapest path from city f to city a?

$15+$12+$4 = $31

$9+$3+$7+$14 = $33

$8+$7+$14 = $29

f → d → c → a

...

11.5 *Minimum Spanning Trees*

Prim's algorithm

procedure *Prim*(*G*: weighted connected undirected graph with *n* vertices)
T := a minimum weight edge
for *i* := 1 to *n-2*
    e := an edge of minimum weight incident to a vertex in T and not forming a simple circuit in T if added to T
    T := T with added edge *e*
return T

We will assume that the edges are ordered when we need to choose between two or more edges with the same weights.

11.5 *Minimum Spanning Trees*

Prim's algorithm

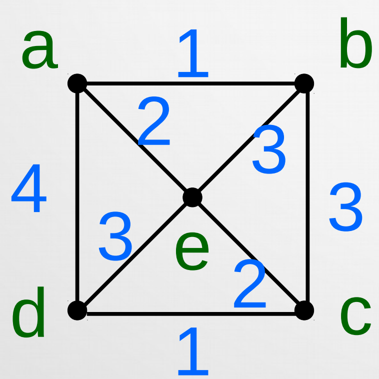
procedure *Prim*(*G*: weighted connected undirected graph with *n* vertices)
T := a minimum weight edge
for *i* := 1 to *n-2*
    e := an edge of minimum weight incident to a vertex in T and not forming a simple circuit in T if added to T
    T := T with added edge *e*
return T

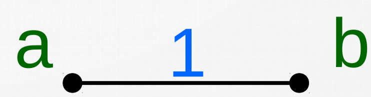
We will assume that the edges are ordered when we need to choose between two or more edges with the same weights.

11.5 *Minimum Spanning Trees*

Prim's algorithm

procedure *Prim*(*G*: weighted conn. undir.graph with *n* vertices)
T := a minimum weight edge
for *i* := 1 to *n-2*
    e := an edge of minimum weight incident to a vertex
    in T and not forming a simple circuit in T if added to T
    T := T with added edge *e*
return T

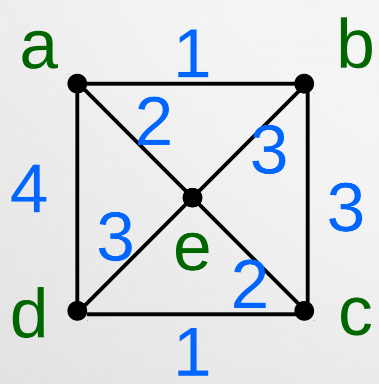


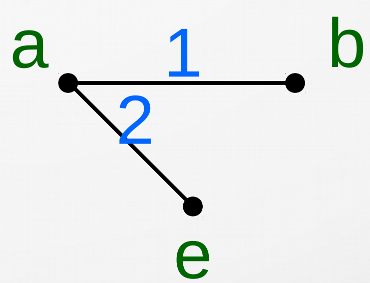


11.5 *Minimum Spanning Trees*

Prim's algorithm

procedure *Prim*(*G*: weighted conn. undir.graph with *n* vertices)
T := a minimum weight edge
for *i* := 1 to *n-2*   *i = 1*
→ *e* := an edge of minimum weight incident to a vertex
in T and not forming a simple circuit in T if added to T
T := T with added edge *e*
return T

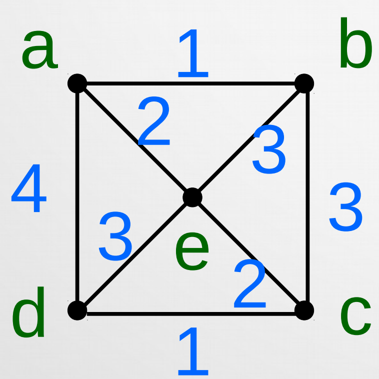


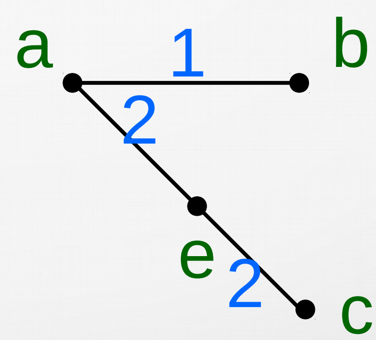


11.5 *Minimum Spanning Trees*

Prim's algorithm

procedure *Prim*(*G*: weighted conn. undir.graph with *n* vertices)
T := a minimum weight edge
for *i* := 1 to *n-2* *i = 2*
→ *e* := an edge of minimum weight incident to a vertex
in T and not forming a simple circuit in T if added to T
T := T with added edge *e*
return T

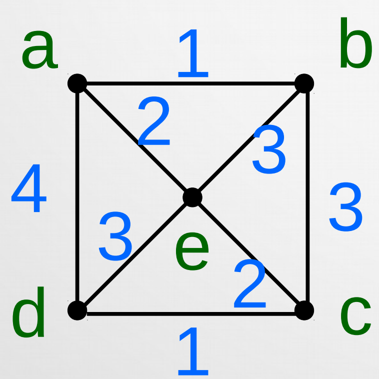


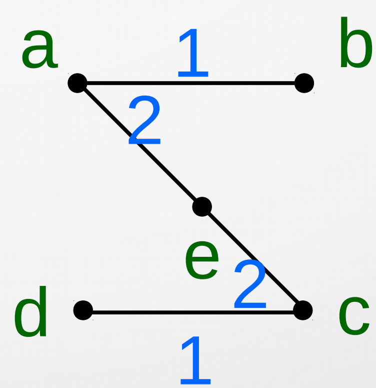


11.5 *Minimum Spanning Trees*

Prim's algorithm

procedure *Prim*(*G*: weighted conn. undir.graph with *n* vertices)
T := a minimum weight edge
for *i* := 1 to *n-2*   *i = 3 = 5-2, last iteration*
→ *e* := an edge of minimum weight incident to a vertex
in T and not forming a simple circuit in T if added to T
T := T with added edge *e*
return T

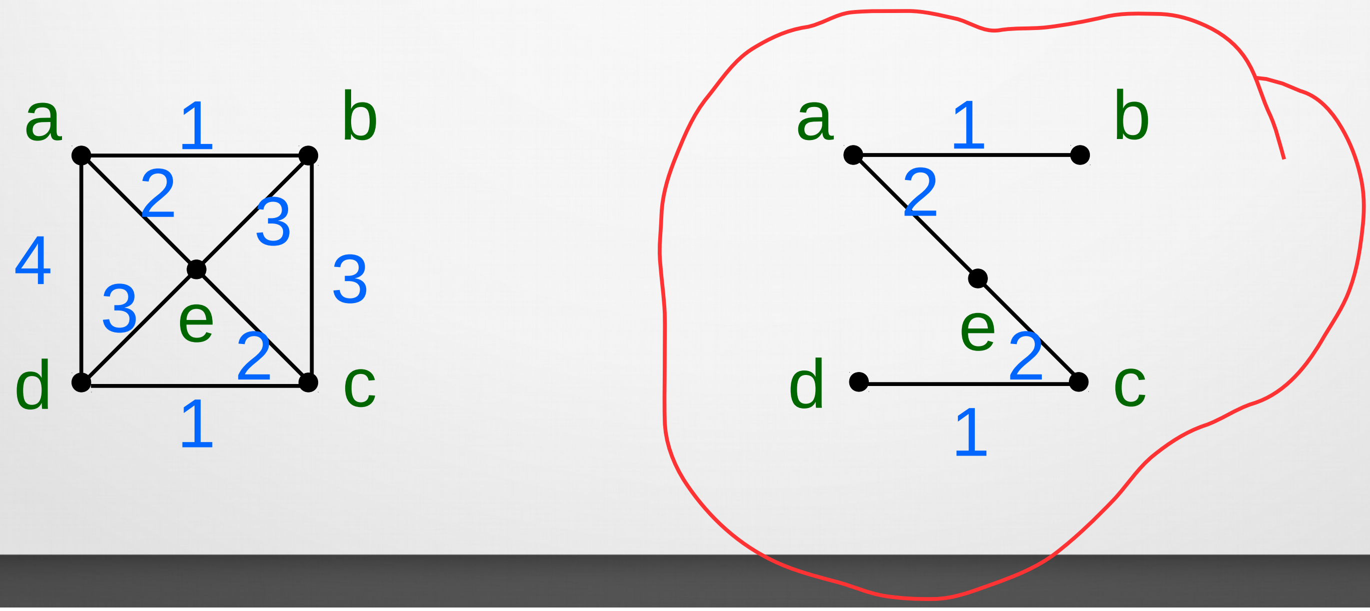




11.5 *Minimum Spanning Trees*

Prim's algorithm

procedure *Prim*(*G*: weighted conn. undir.graph with *n* vertices)
T := a minimum weight edge
for *i* := 1 to *n-2*   *i = 3*
    e := an edge of minimum weight incident to a vertex
    in T and not forming a simple circuit in T if added to T
    T := T with added edge *e*
return T

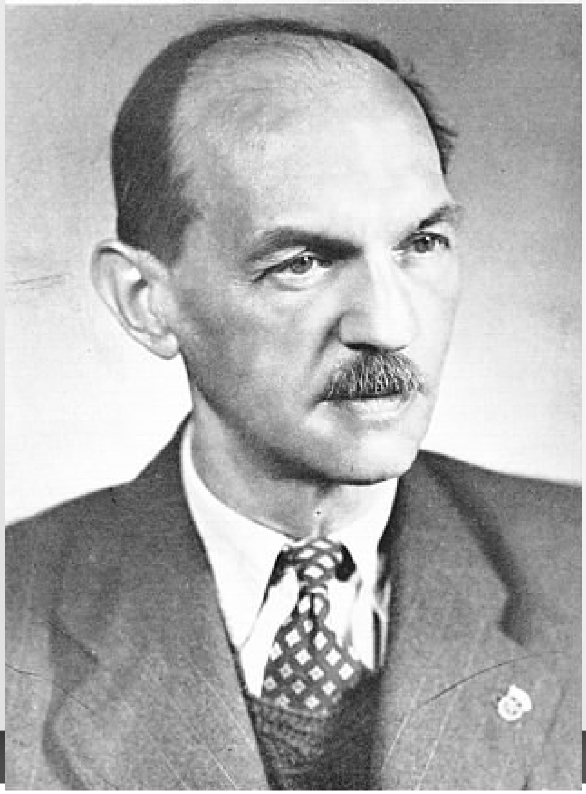


11.5 *Minimum Spanning Trees*

Prim's algorithm

Historical information:

- originally discovered by the Czech mathematician Vojtěch Jarník in 1930,
- was rediscovered in 1957 by Robert Prim (independent work).



11.5 *Minimum Spanning Trees*

Prim's algorithm

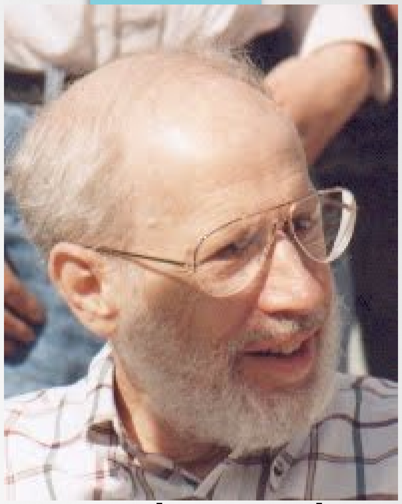
Historical information:

- originally discovered by the Czech mathematician Vojtěch Jarník in 1930,
- was rediscovered in 1957 by Robert Prim (independent work).

During his career at Bell Laboratories, Robert Prim along with coworker Joseph Kruskal developed two different algorithms for finding a minimum spanning tree in a weighted graph

11.5 *Minimum Spanning Trees*

Kruskal's algorithm



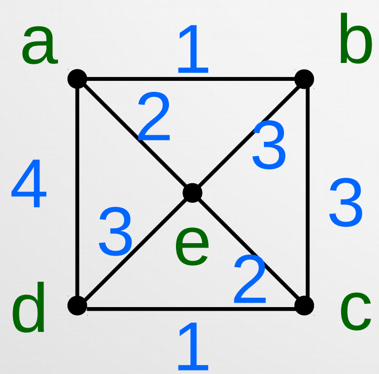
The second algorithm up for discussion was discovered by Joseph Kruskal in 1956, although the basic ideas it uses were described much earlier.

procedure *Kruskal*(*G*: weighted connected undirected graph
with *n* vertices)
T:= empty graph
for *i* := *1* to *n-1*
    e:= any edge in *G* with smallest weight that does not form a
        simple circuit when added to T
    T := T with added *e*
return T

11.5 *Minimum Spanning Trees*

Kruskal's algorithm

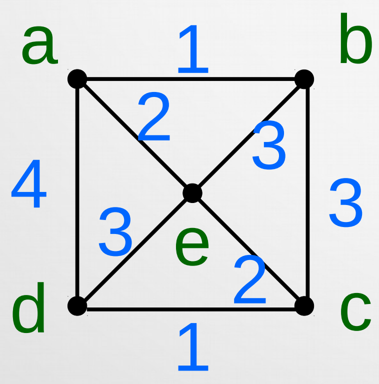
procedure *Kruskal*(*G*: weighted conn. undir.graph with *n* vertices)
T:= empty graph
for *i* := *1* to *n-1*
    e:= any edge in *G* with smallest weight that does not form a simple circuit when added to T
    T := T with added *e*
return T



11.5 *Minimum Spanning Trees*

Kruskal's algorithm

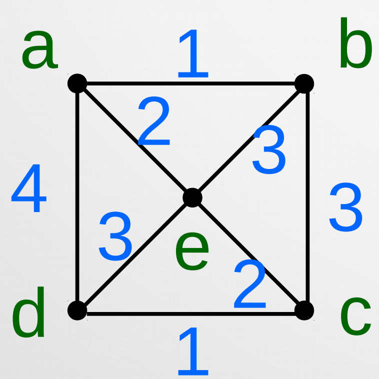
procedure *Kruskal*(*G*: weighted conn. undir.graph with *n* vertices)
T:= empty graph
for *i* := *1* to *n-1*
    e:= any edge in *G* with smallest weight that does not form a
        simple circuit when added to T
    T := T with added *e*
return T

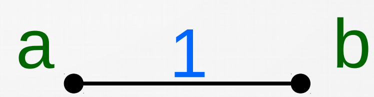


11.5 *Minimum Spanning Trees*

Kruskal's algorithm

procedure *Kruskal*(*G*: weighted conn. undir.graph with *n* vertices)
T:= empty graph
for *i* := *1* to *n-1*   *i := 1*
→ *e*:= any edge in *G* with smallest weight that does not form a simple circuit when added to T
T := T with added *e*
return T

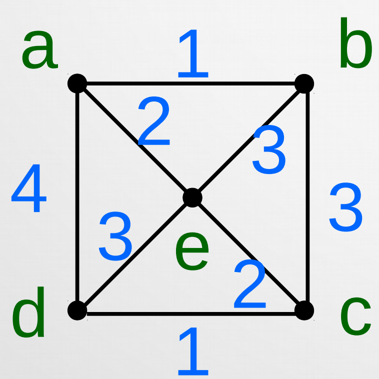


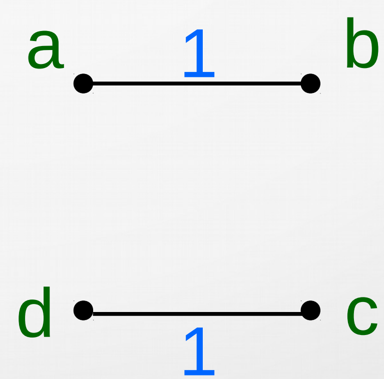


11.5 *Minimum Spanning Trees*

Kruskal's algorithm

procedure *Kruskal*(*G*: weighted conn. undir.graph with *n* vertices)
T:= empty graph
for *i* := *1* to *n-1*   *i := 2*
→ *e*:= any edge in *G* with smallest weight that does not form a simple circuit when added to T
T := T with added *e*
return T

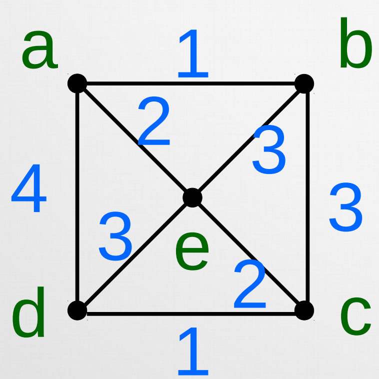


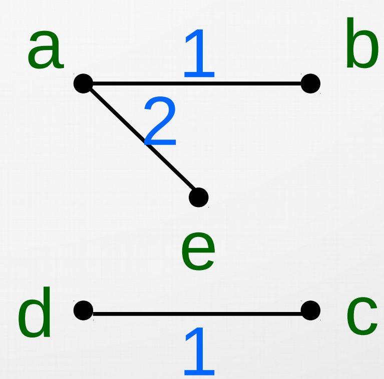


11.5 *Minimum Spanning Trees*

Kruskal's algorithm

procedure *Kruskal*(*G*: weighted conn. undir.graph with *n* vertices)
T:= empty graph
for *i* := *1* to *n-1*   *i := 3*
→ *e*:= any edge in *G* with smallest weight that does not form a simple circuit when added to T
T := T with added *e*
return T

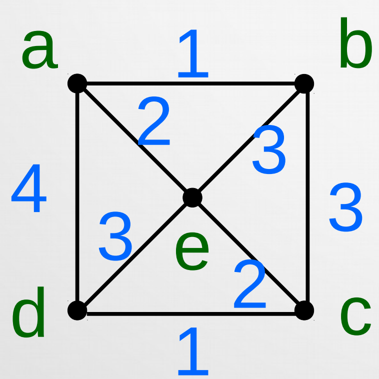


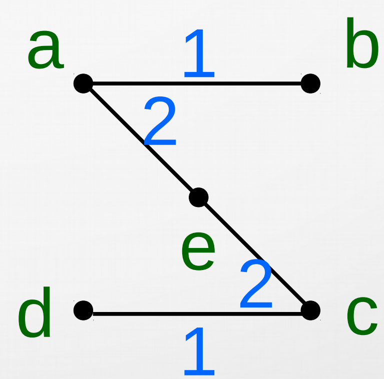


11.5 *Minimum Spanning Trees*

Kruskal's algorithm

procedure *Kruskal*(*G*: weighted conn. undir.graph with *n* vertices)
T:= empty graph
for *i* := *1* to *n-1*   *i := 4*
→ *e*:= any edge in *G* with smallest weight that does not form a simple circuit when added to T
T := T with added *e*
return T





11.5 *Minimum Spanning Trees*

Kruskal's algorithm

procedure *Kruskal*(*G*: weighted conn. undir.graph with *n* vertices)
T:= empty graph
for *i* := *1* to *n-1*   *i := 4*
    e:= any edge in *G* with smallest weight that does not form a
        simple circuit when added to T
    T := T with added *e*
return T