Algorithms

(Greedy Technique)

Pramod Ganapathi
Department of Computer Science
State University of New York at Stony Brook

March 31, 2022

Contents

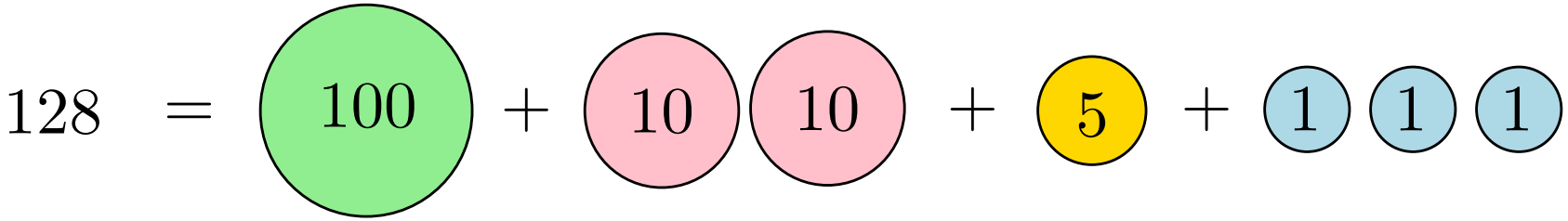
What is greedy technique?

- Domain.
  This technique works on optimization problems.
- Core idea.
  Choose the best option in every step.
- Assumption.
  Local optimality at every step leads to global optimality.
  (This idea is not true in general but works for some problems.)

Cashier's Algorithm HOME

Coin change problem

- [Link] A cashier has an infinite supply of coins of denominations $1, 5, 10, 100$. Devise a method to pay any amount n to customer using fewest number of coins.
- Coin denominations $= \{1, 5, 10, 100\}$, $n = 128$, and mincoins $= 7$, because,



Cashier's greedy algorithm

CASHIER'SGREEDYALGORITHM(n)

Input: Value n

Output: Minimum #coins to make change for n using denominations $\{1, 5, 10, 100\}$.

1. Sort $C[1..4]$ in decreasing order
2. $C[1..4] \leftarrow [100, 10, 5, 1]$
3. $A[1..4] \leftarrow [0, 0, 0, 0]$
4. **for** $i \leftarrow 1$ **to** 4 **do**
5.   **if** $n = 0$ **then break**
6. $A[i] \leftarrow n/C[i]$
7. $n \leftarrow n - A[i]$
8. $mincoins \leftarrow A[1] + A[2] + A[3] + A[4]$
9. **return** $A[1..4]$ and $mincoins$

Non-optimality of cashier's greedy algorithm

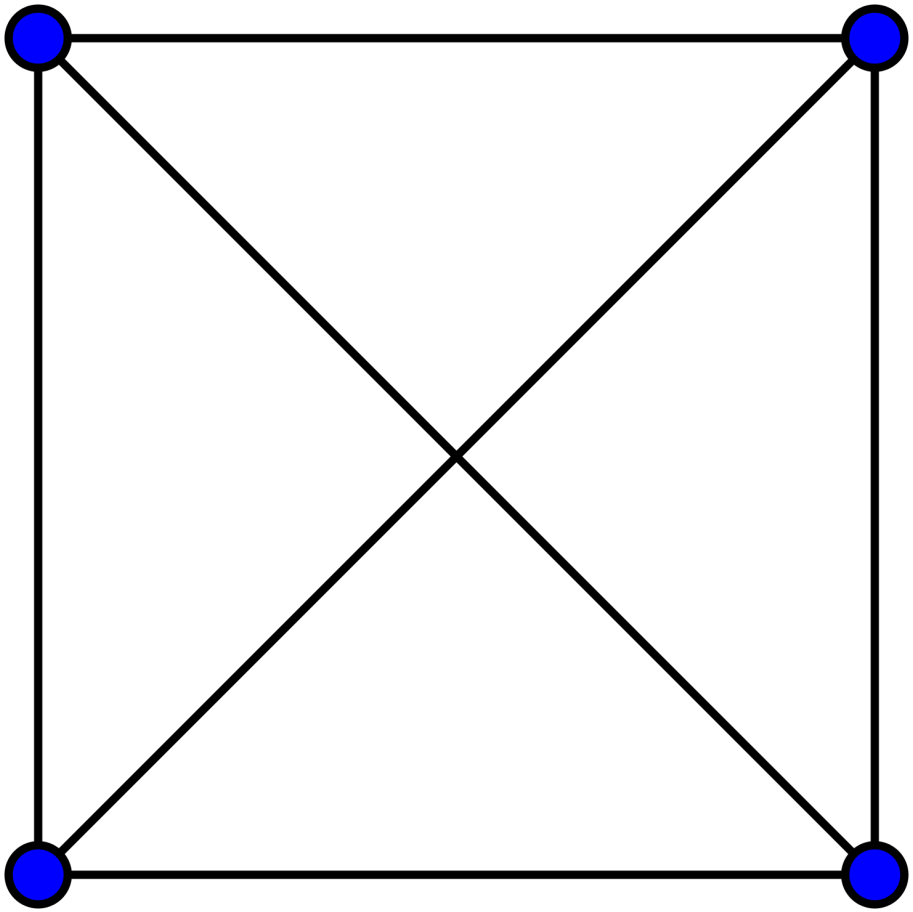
Denominations	Optimal?	Counterexample
$\{1, 5, 10, 100\}$	✓	–
$\{1, k_1, (k_1 k_2), .., (k_1..k_m)\}$	✓	–
$\{1, 3, 4\}$	✗	Cashier: $6 = 4 + 1 + 1$
		Optimal: $6 = 3 + 3$
$\{2, 3, 5\}$	Infeasible	Cashier: $6 = 5 +?$
		Optimal: $6 = 3 + 3$

Minimum spanning tree (MST) problem

Problem

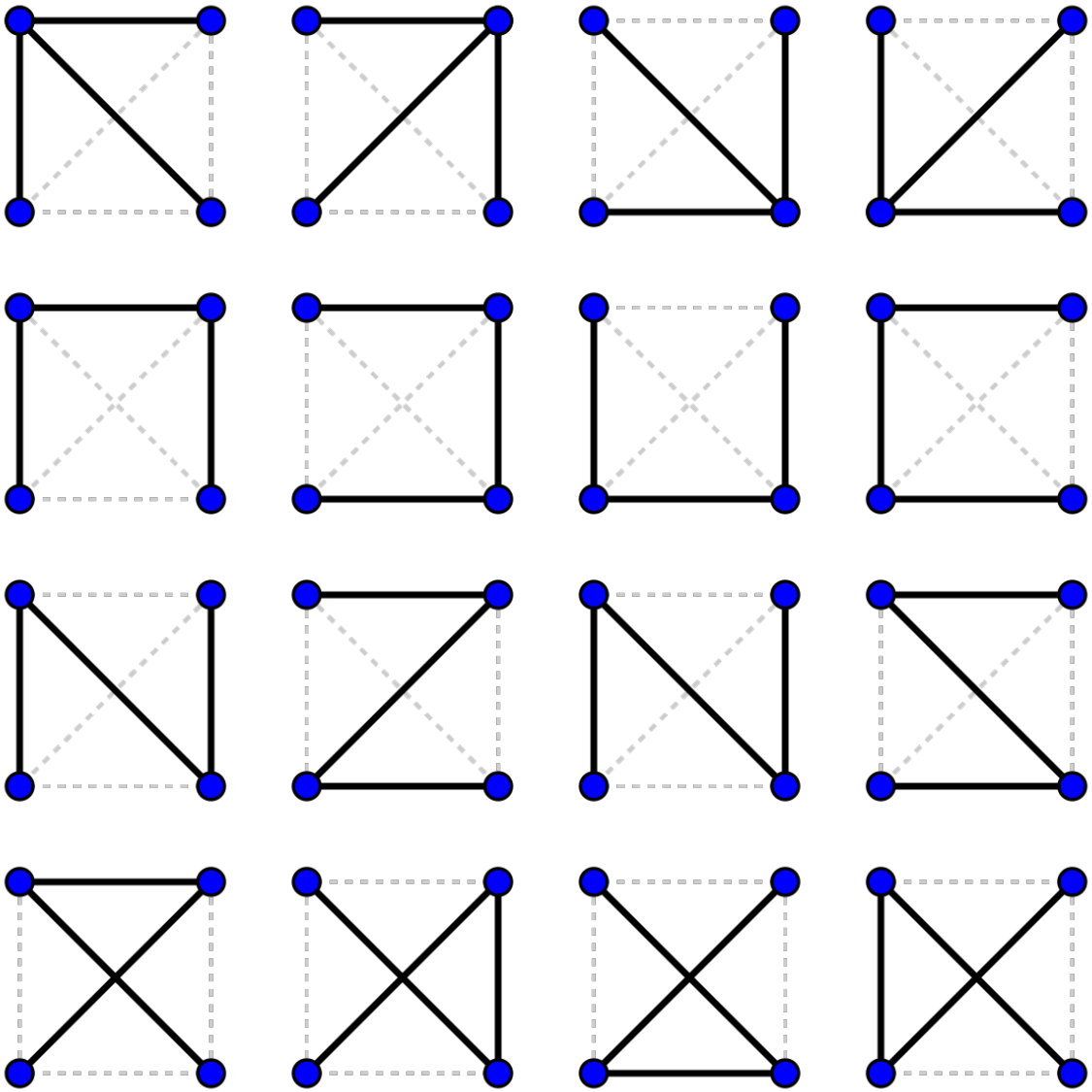
- Given n points, connect them in the cheapest possible way so that there will be a path between every pair of points.

Find the spanning trees of this graph



Source: (https://commons.wikimedia.org/wiki/Category:Complete_graph_K4#/media/File:Complete_graph_K4.svg)

All spanning trees



Source: (https://de.wikipedia.org/wiki/Datei:Spanning_Trees_qtl1.svg)

Boruvka's Algorithm HOME

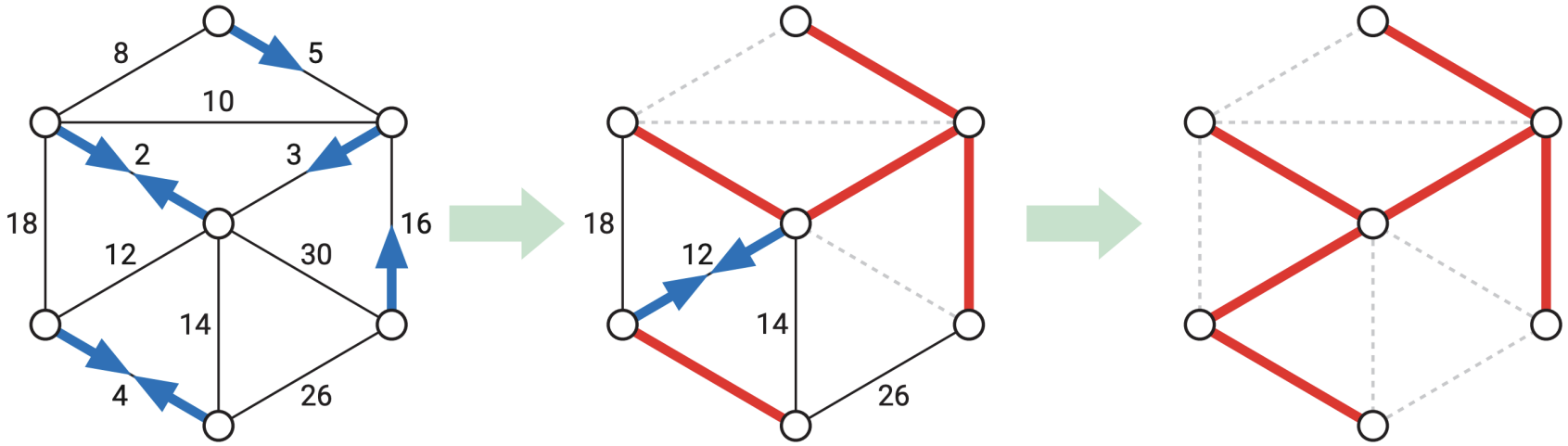
Algorithm

MST-Boruvka(G)

Input: Graph G
Output: Minimum spanning tree T of G

1. forest T consists of all vertices, each vertex is a component
2. **while** forest T has more than one component **do**
3. add to forest T the shortest edge out of each component
4. **return** T

Example



Source: Jeff Erickson's Algorithms book. (`https://jeffe.cs.illinois.edu/teaching/algorithms/`)

Jarnik's Algorithm HOME

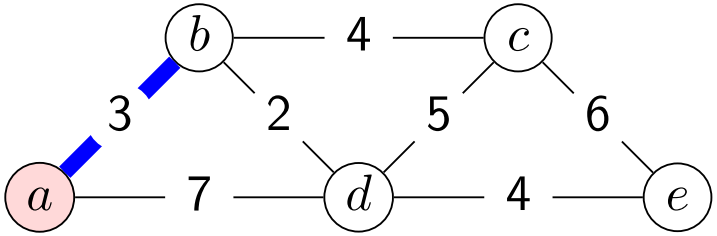
Algorithm

MST-JARNIK(G)

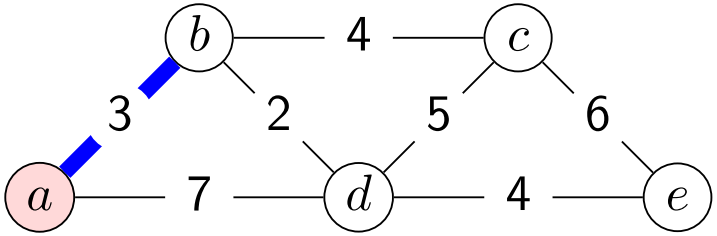
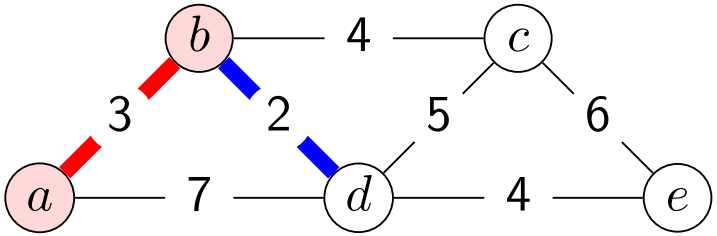
Input: Graph G
Output: Minimum spanning tree T of G

1. $T \leftarrow$ an arbitrary vertex
2. **for** $(|V| - 1)$ times **do**
3. add to T a shortest edge that has exactly one end point in T
4. **return** T

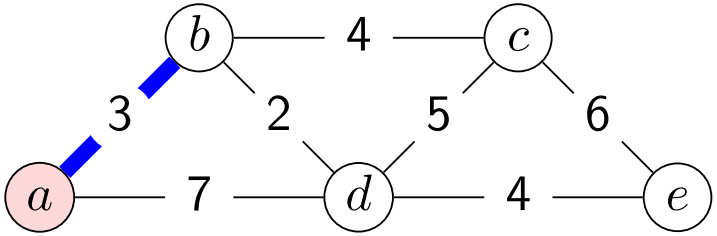
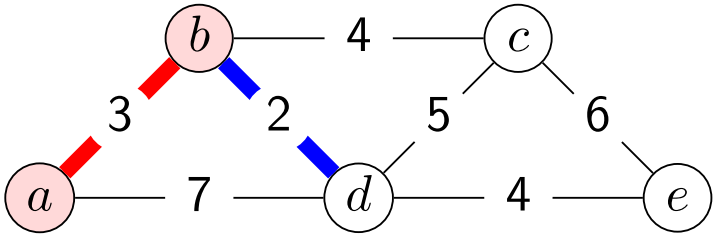
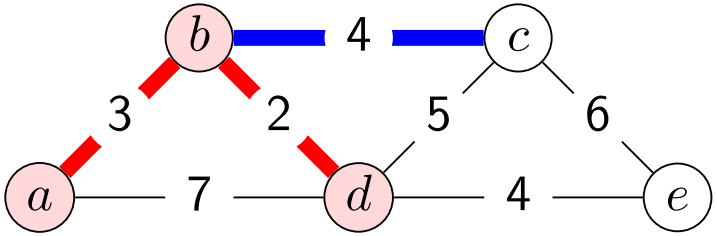
Example 1

Tree	Remaining vertices	Evolving tree
$a(0,-)$	$b(3,a)$, $c(\infty,-)$, $d(7,a)$, $e(\infty,-)$	

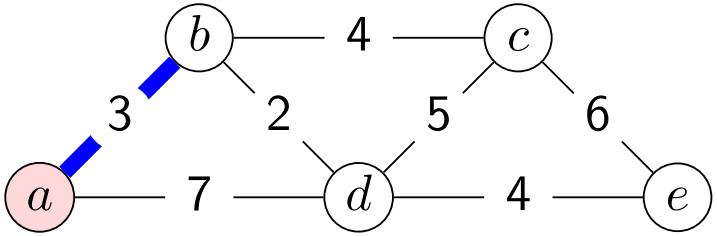
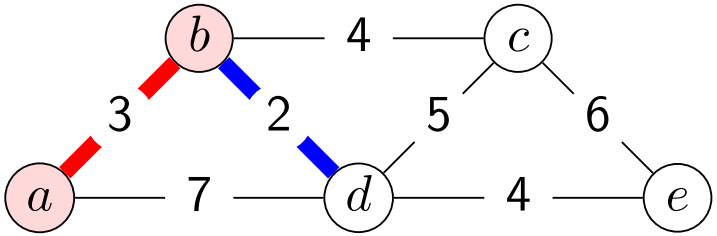
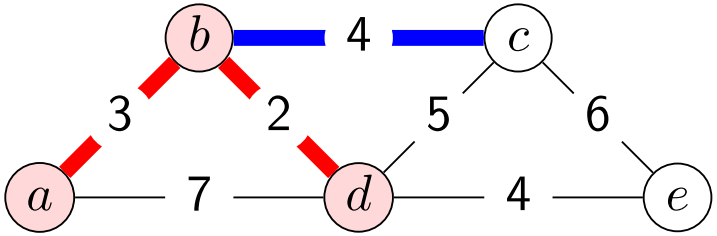
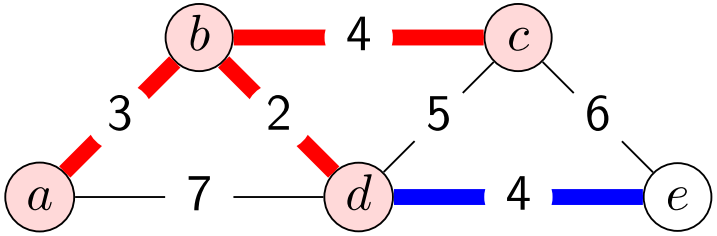
Example 1

Tree	Remaining vertices	Evolving tree
$a(0,-)$	$b(3,a)$, $c(\infty,-)$, $d(7,a)$, $e(\infty,-)$	
$b(3,a)$	$c(4,b)$, $d(2,b)$, $e(\infty,-)$	

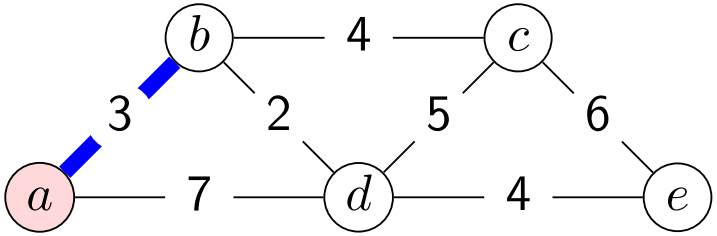
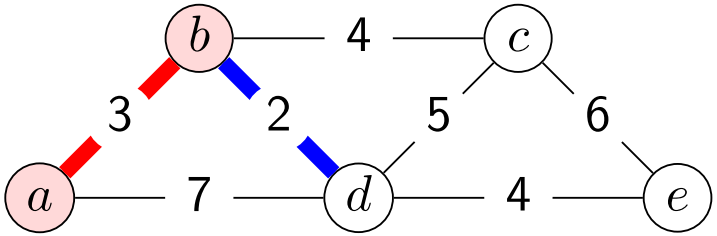
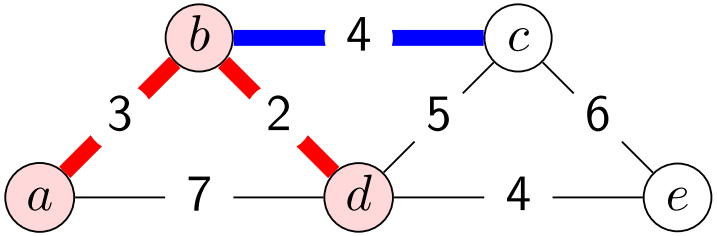
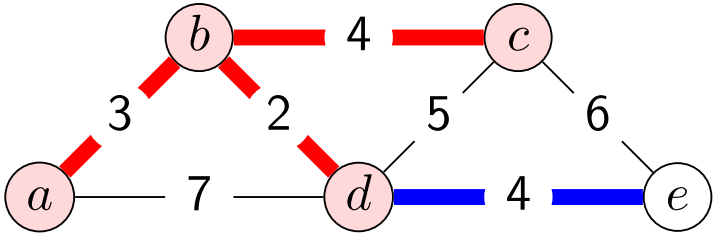
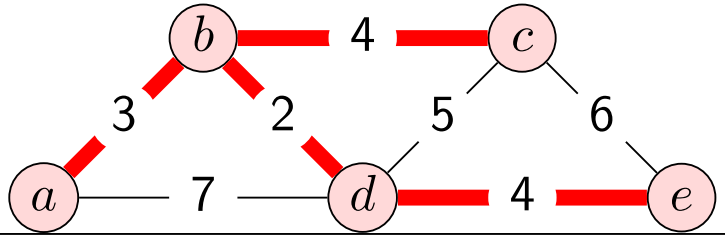
Example 1

Tree	Remaining vertices	Evolving tree
$a(0,-)$	$b(3,a)$, $c(\infty,-)$, $d(7,a)$, $e(\infty,-)$	
$b(3,a)$	$c(4,b)$, $d(2,b)$, $e(\infty,-)$	
$d(2,b)$	$c(4,b)$, $e(4,d)$	

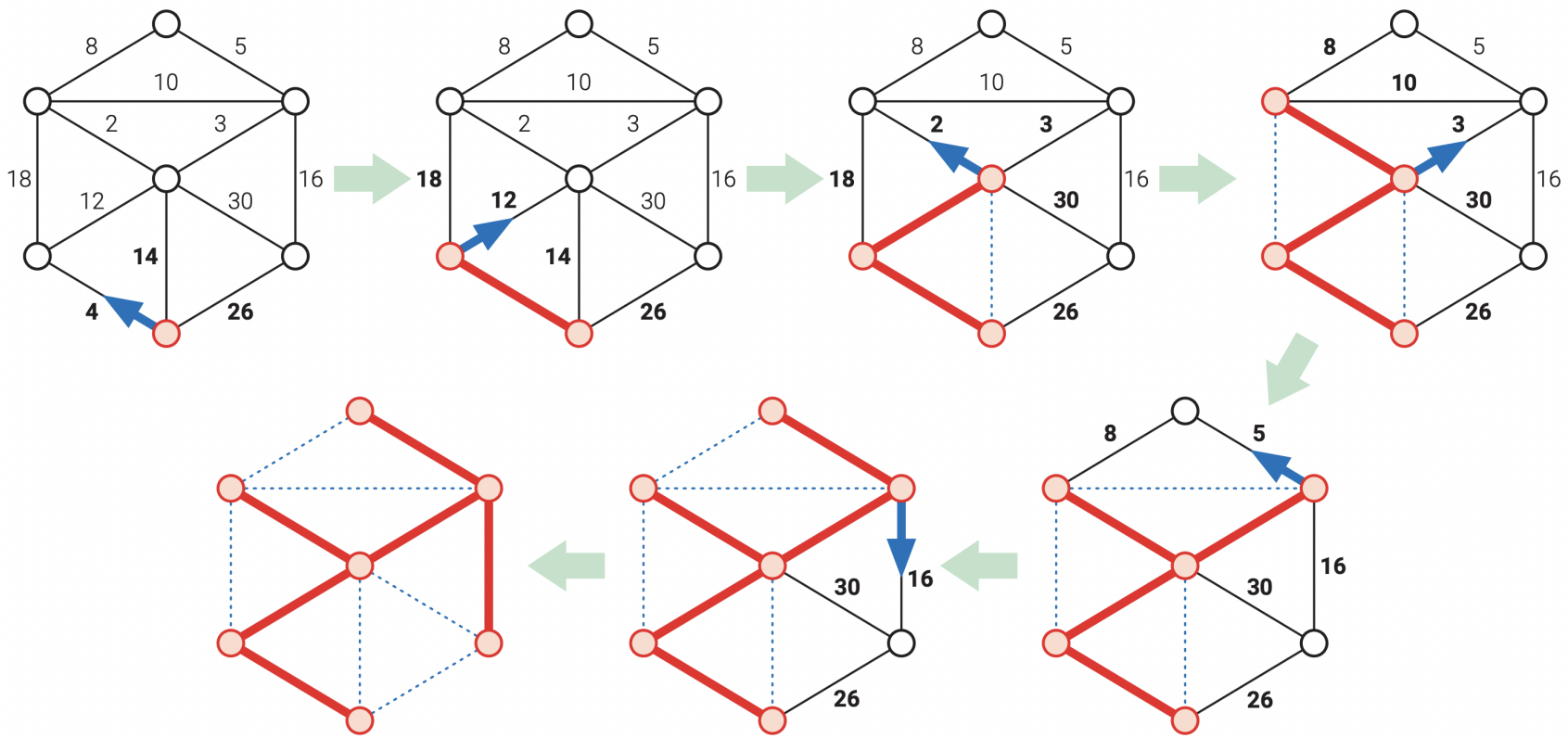
Example 1

Tree	Remaining vertices	Evolving tree
$a(0,-)$	$b(3,a)$, $c(\infty,-)$, $d(7,a)$, $e(\infty,-)$	
$b(3,a)$	$c(4,b)$, $d(2,b)$, $e(\infty,-)$	
$d(2,b)$	$c(4,b)$, $e(4,d)$	
$c(4,b)$	$e(4,d)$	

Example 1

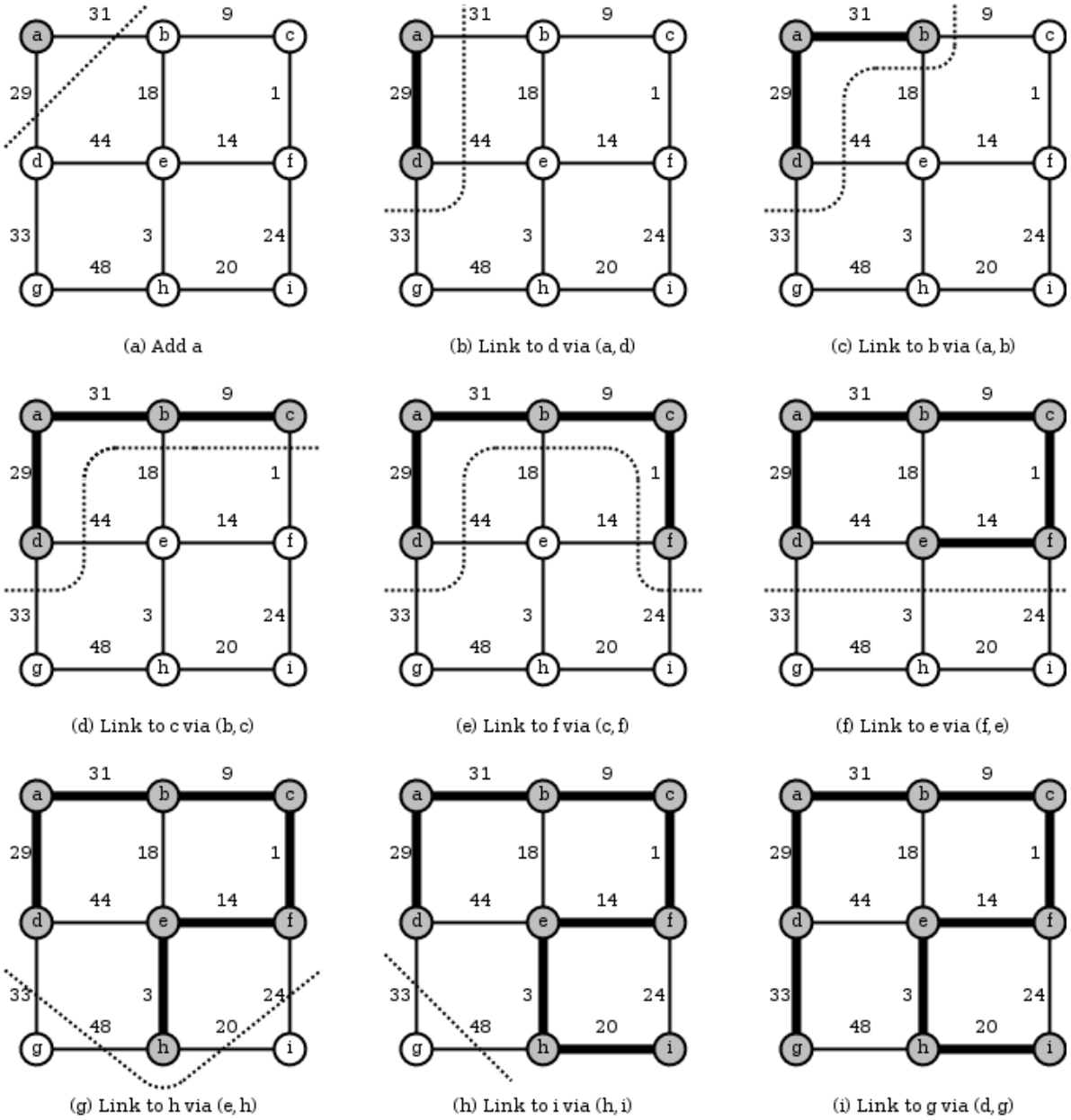
Tree	Remaining vertices	Evolving tree
$a(0,-)$	$b(3,a)$, $c(\infty,-)$, $d(7,a)$, $e(\infty,-)$	
$b(3,a)$	$c(4,b)$, $d(2,b)$, $e(\infty,-)$	
$d(2,b)$	$c(4,b)$, $e(4,d)$	
$c(4,b)$	$e(4,d)$	
$e(4,d)$		

Example 2



Source: Jeff Erickson's Algorithms book. (`https://jeffe.cs.illinois.edu/teaching/algorithms/`)

Example 3



(a) Add a

(b) Link to d via (a, d)

(c) Link to b via (a, b)

(d) Link to c via (b, c)

(e) Link to f via (c, f)

(f) Link to e via (f, e)

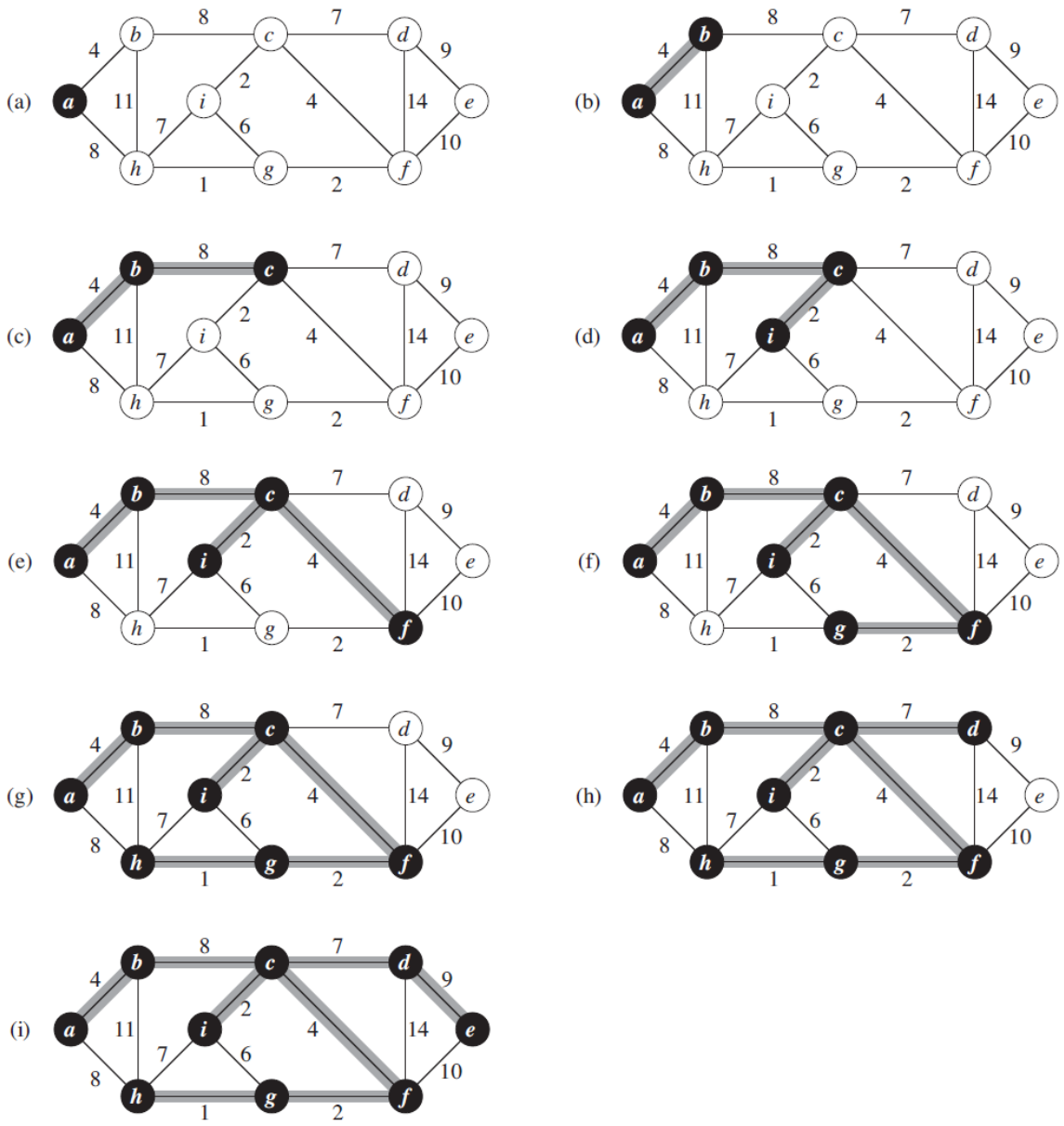
(g) Link to h via (e, h)

(h) Link to i via (h, i)

(i) Link to g via (d, g)

Source: (http://users.cecs.anu.edu.au/~Alistair.Rendell)

Example 4



Source: CLRS' Introduction to Algorithms textbook

Algorithm

MST-JARNIK(G)

Input: Graph $G = (V, E)$
Output: Set of edges E_T composing a minimum spanning tree of G

1. $V_T \leftarrow \{v_0\}$
2. $E_T \leftarrow \phi$
3. **for** $i \leftarrow 1$ to $|V| - 1$ **do**
4. 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 $V - V_T$
5. $V_T \leftarrow V_T \cup \{u^*\}$
6. $E_T \leftarrow E_T \cup \{e^*\}$
7. **return** E_T

Min-heap data structure

- What efficient data structure can we use to find the minimum-weight edge between V_T and $V - V_T$?
  Min-heap
  (we could also use balanced BST. However, min-heap is lighter data structure compared with balanced BST.)
- What are the items in the min-heap?
  We store $(key, value)$ pair for each vertex in $V - V_T$, where,
  $key =$ distance to the nearest vertex in V_T
  $value =$ nearest vertex (or parent) in V_T

Algorithm

$\text{MST-JARNIK}(G, weight, treeroot)$

Input: Graph $G = (V, E)$
Output: Minimum spanning tree of G

```
for each vertex u ∈ V do
  (u.distance, u.parent) ← (∞, null)                ▷ (key, value) pair
treeroot.distance ← 0
heap ← V                                            ▷ heap represents V − V_T
while heap is nonempty do
  u ← heap.DeleteMin()   ▷ u ∈ V − V_T is the nearest vertex to V_T
                ▷ Now update the distances of the fringe vertices in V − V_T
  for each vertex v adjacent to vertex u do
    if v ∈ heap and weight(u, v) < v.distance then
      (v.distance, v.parent) ← (weight(u, v), u)
return set V with parent information
```

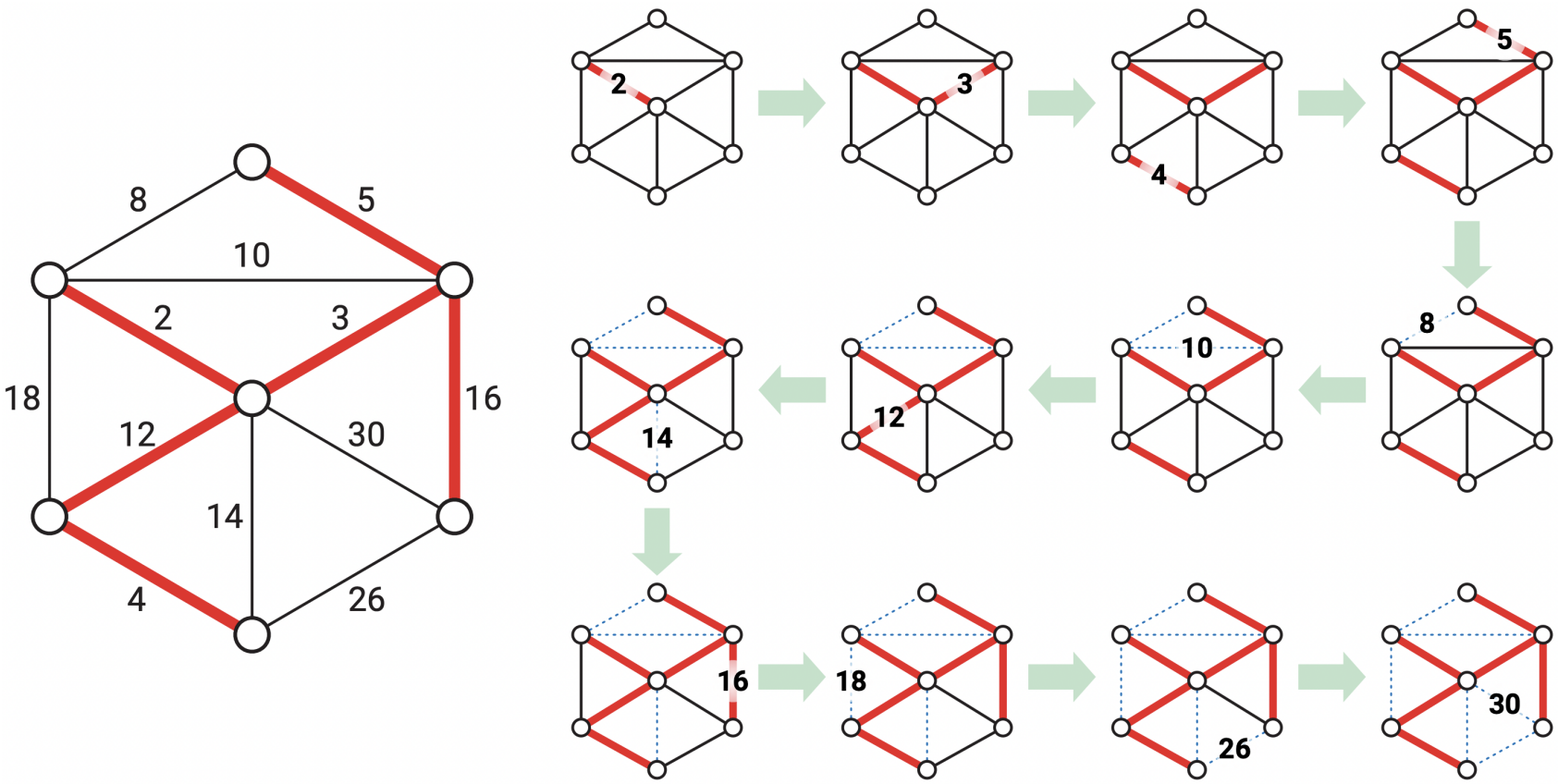
Kruskal's Algorithm HOME

Algorithm

MST-KRUSKAL(G)

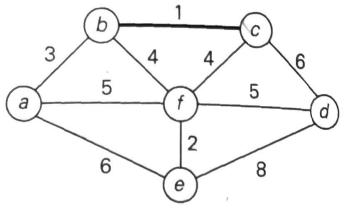
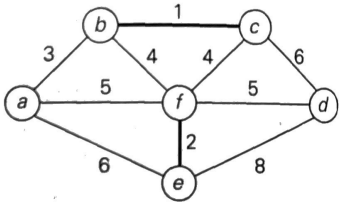
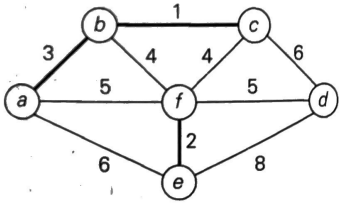
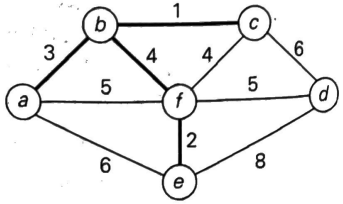
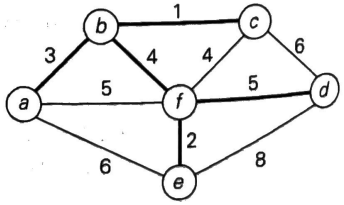
Input: Graph G
Output: Minimum spanning tree T of G
1. forest $T \leftarrow \phi$
2. **for** $(|V| - 1)$ times **do**
3. add to forest T a shortest edge that does not introduce a cycle in T
4. **return** T

Example



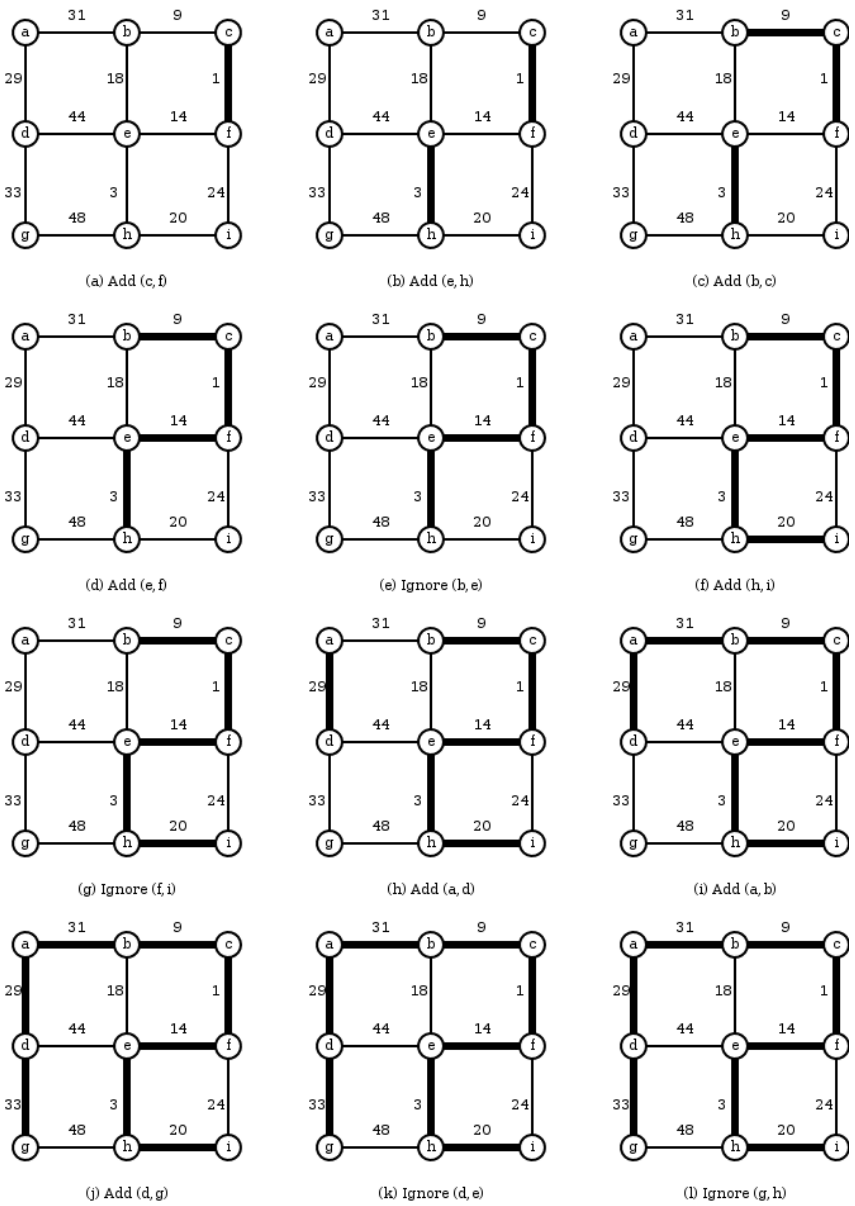
Source: Jeff Erickson's Algorithms book. (`https://jeffe.cs.illinois.edu/teaching/algorithms/`)

Example

Tree edges	Sorted list of edges	Illustration
	bc ef ab bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
bc 1	bc **ef** ab bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
ef 2	bc ef **ab** bf cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
ab 3	bc ef ab **bf** cf af df ae cd de 1 2 3 4 4 5 5 6 6 8	
bf 4	bc ef ab bf cf af **df** ae cd de 1 2 3 4 4 5 5 6 6 8	
df 5		

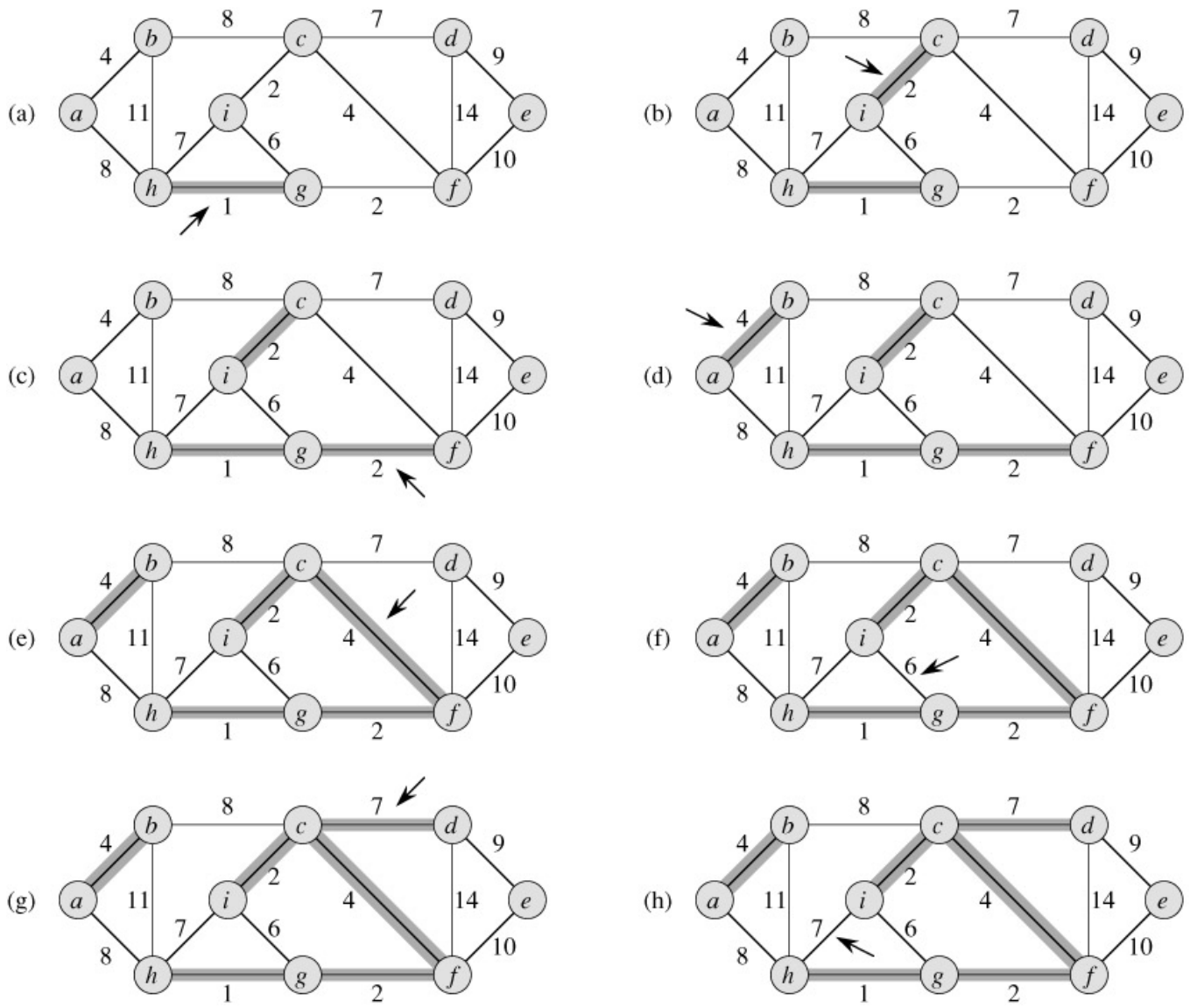
Source: Anany Levitin's Introduction to Design and Analysis of Algorithms textbook

Example



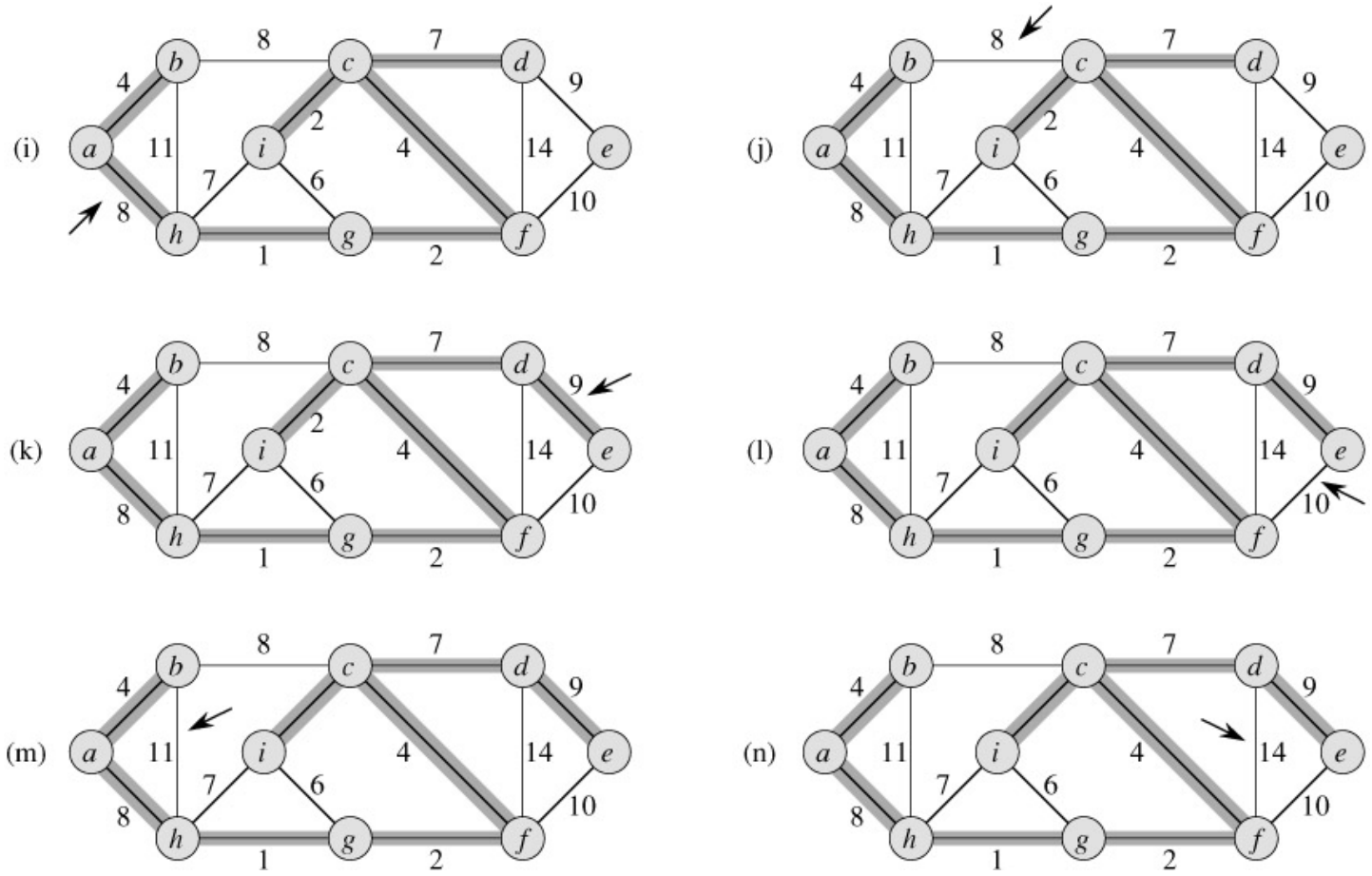
Source: (`http://users.cecs.anu.edu.au/~Alistair.Rendell`)

Example



Source: CLRS' Introduction to Algorithms textbook

Example



Source: CLRS' Introduction to Algorithms textbook

Algorithm

MST-KRUSKAL$(G, weight)$

Input: Graph G
Output: Set of edges E_T composing a minimum spanning tree T of G

1. sort E in nondecreasing order of the edge weights
   suppose $weight(e_1) \leq weight(e_2) \leq \cdots \leq weight(e_{|E|})$
2. forest T is defined by its edges E_T
3. $E_T \leftarrow \phi$; $ecount \leftarrow 0$ ▷ initialize the set of forest edges and its size
4. $k \leftarrow 0$ ▷ initialize the number of processed edges
5. **while** $count < |V| - 1$ **do**
6. $\quad k \leftarrow k + 1$
7. $\quad$ **if** $E_T \cup \{e_k\}$ is acyclic **then**
8. $\quad\quad E_T \leftarrow E_T \cup \{e_k\}$
9. $\quad\quad ecount \leftarrow ecount + 1$
10. **return** E_T

Union-find ADT

- Kruskal's algorithm requires dynamic partitioning of n-element set S into a collection of disjoint subsets $S_1, S_2, \ldots, S_k$.
  What is a good ADT for solving this problem?
  Union-find ADT
- What are the operations in a union-find ADT?

Method	Functionality
MakeSet(x)	Creates one-element set $\{x\}$.
FindSet(x)	Returns a subset containing x.
UnionSet(x,y)	Constructs the union of subsets S_x and S_y containing x and y, respectively, and adds it to the collection to replace S_x and S_y, which are deleted from it.

Operations using union-find ADT

- Let set $S = \{1, 2, 3, 4, 5, 6\}$.

Method	Return value	Union-find contents
$\textsc{MakeSet}(1)$	-	$\{1\}$
$\textsc{MakeSet}(2)$	-	$\{1\}, \{2\}$
$\ldots$	$\ldots$	$\ldots$
$\textsc{MakeSet}(6)$	-	$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}$
$\textsc{FindSet}(3)$	$\{3\}$	$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}$
$\textsc{UnionSet}(1, 4)$	-	$\{1, 4\}, \{2\}, \{3\}, \{5\}, \{6\}$
$\textsc{UnionSet}(5, 2)$	-	$\{1, 4\}, \{5, 2\}, \{3\}, \{6\}$
$\textsc{UnionSet}(4, 5)$	-	$\{1, 4, 5, 2\}, \{3\}, \{6\}$
$\textsc{FindSet}(2)$	$\{1, 4, 5, 2\}$	$\{1, 4, 5, 2\}, \{3\}, \{6\}$
$\textsc{UnionSet}(3, 6)$	-	$\{1, 4, 5, 2\}, \{3, 6\}$
$\textsc{FindSet}(6)$	$\{3, 6\}$	$\{1, 4, 5, 2\}, \{3, 6\}$
$\textsc{UnionSet}(6, 2)$	-	$\{1, 4, 5, 2, 3, 6\}$

Algorithm

$\text{MST-KRUSKAL}(G, weight)$

Input: Graph $G = (V, E)$
Output: Minimum spanning tree T of G

forest $T \leftarrow \phi$
for each vertex $v \in V$ **do**
  $\text{MAKESET}(v)$
sort the edges E in nondecreasing order of the edge weights
suppose $weight(e_1) \leq weight(e_2) \leq \cdots \leq weight(e_{|E|})$
for each edge $(u, v) \in E$ taken in nondecreasing order by weight **do**
  if $\text{FINDSET}(u) \neq \text{FINDSET}(v)$ **then** $\triangleright$ $T \cup \{(u, v)\}$ is acyclic
    $T \leftarrow T \cup \{(u, v)\}$
    $\text{UNIONSET}(u, v)$
return T

Shortest path problem

Problem

- Given a weighted digraph and a source node, find the shorest path costs to all other nodes in the graph.

Dijkstra's Algorithm HOME

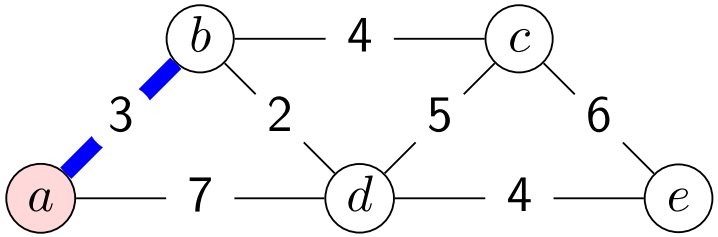
Algorithm

Dijkstra(G, s)

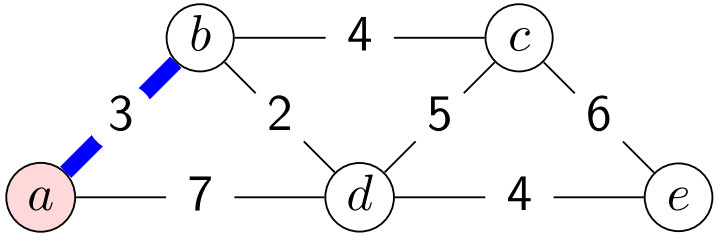
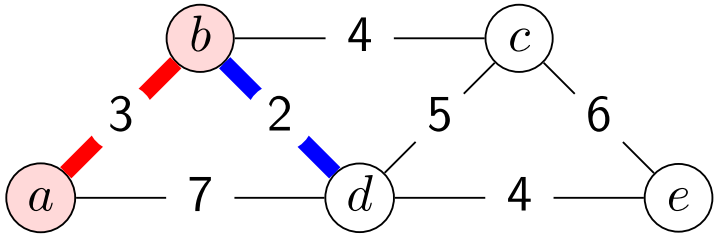
Input: Weighted digraph G with nonnegative weights and source vertex s
Output: Shortest path cost d_v for every vertex v in V

```
T ← s                                          ▷ shortest path tree
for i ← 1 to (|V| − 1) do
  add to T the ith nearest vertex to source s
return T
```

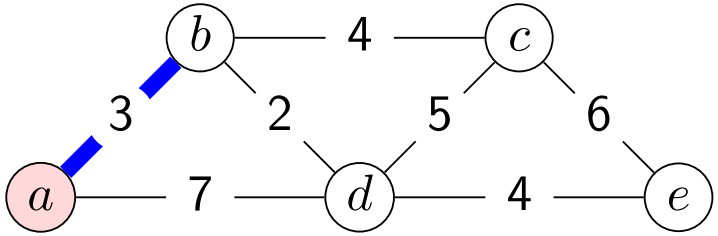
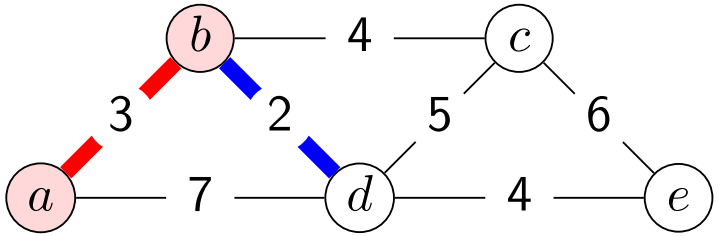
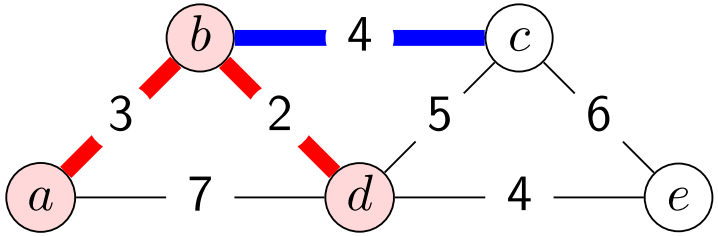
Example 1

Tree	Remaining vertices	Evolving tree
$a(0,-)$	$b(3,a)$, $c(\infty,-)$, $d(7,a)$, $e(\infty,-)$	

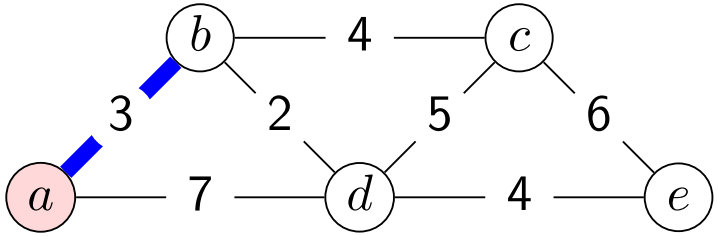
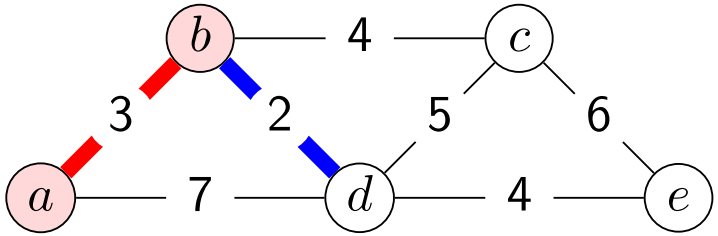
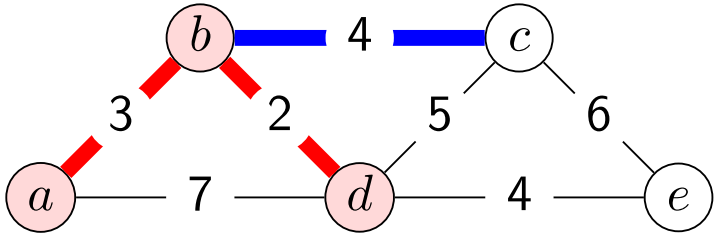
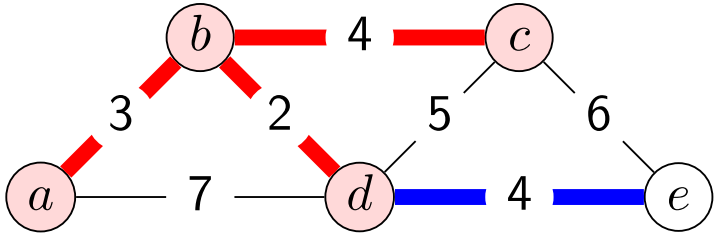
Example 1

Tree	Remaining vertices	Evolving tree
$a(0,-)$	$b(3,a)$, $c(\infty,-)$, $d(7,a)$, $e(\infty,-)$	
$b(3,a)$	$c(3+4,b)$, $d(3+2,b)$, $e(\infty,-)$	

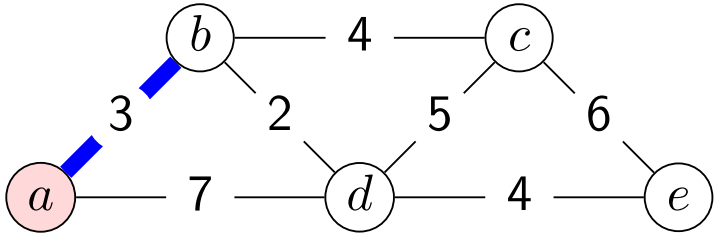
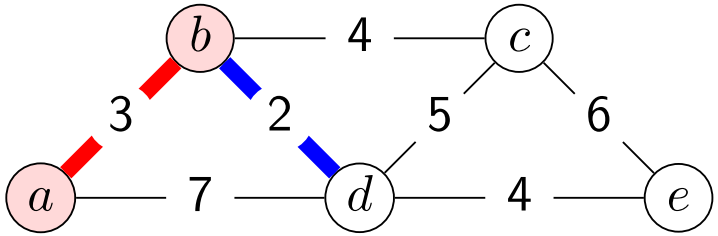
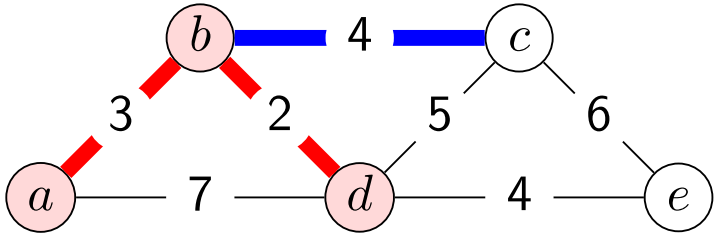
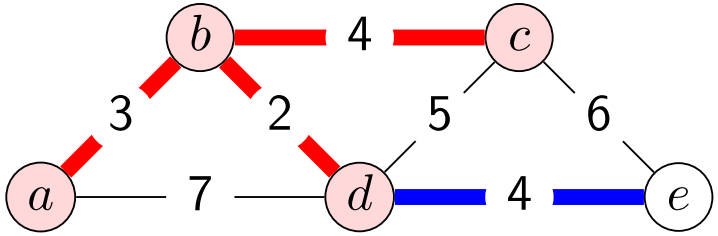
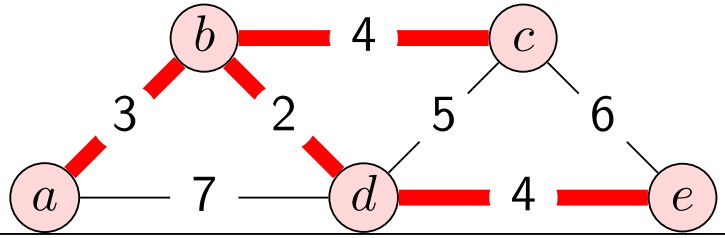
Example 1

Tree	Remaining vertices	Evolving tree
$a(0,-)$	$b(3,a)$, $c(\infty,-)$, $d(7,a)$, $e(\infty,-)$	
$b(3,a)$	$c(3+4,b)$, $d(3+2,b)$, $e(\infty,-)$	
$d(5,b)$	$c(7,b)$, $e(5+4,d)$	

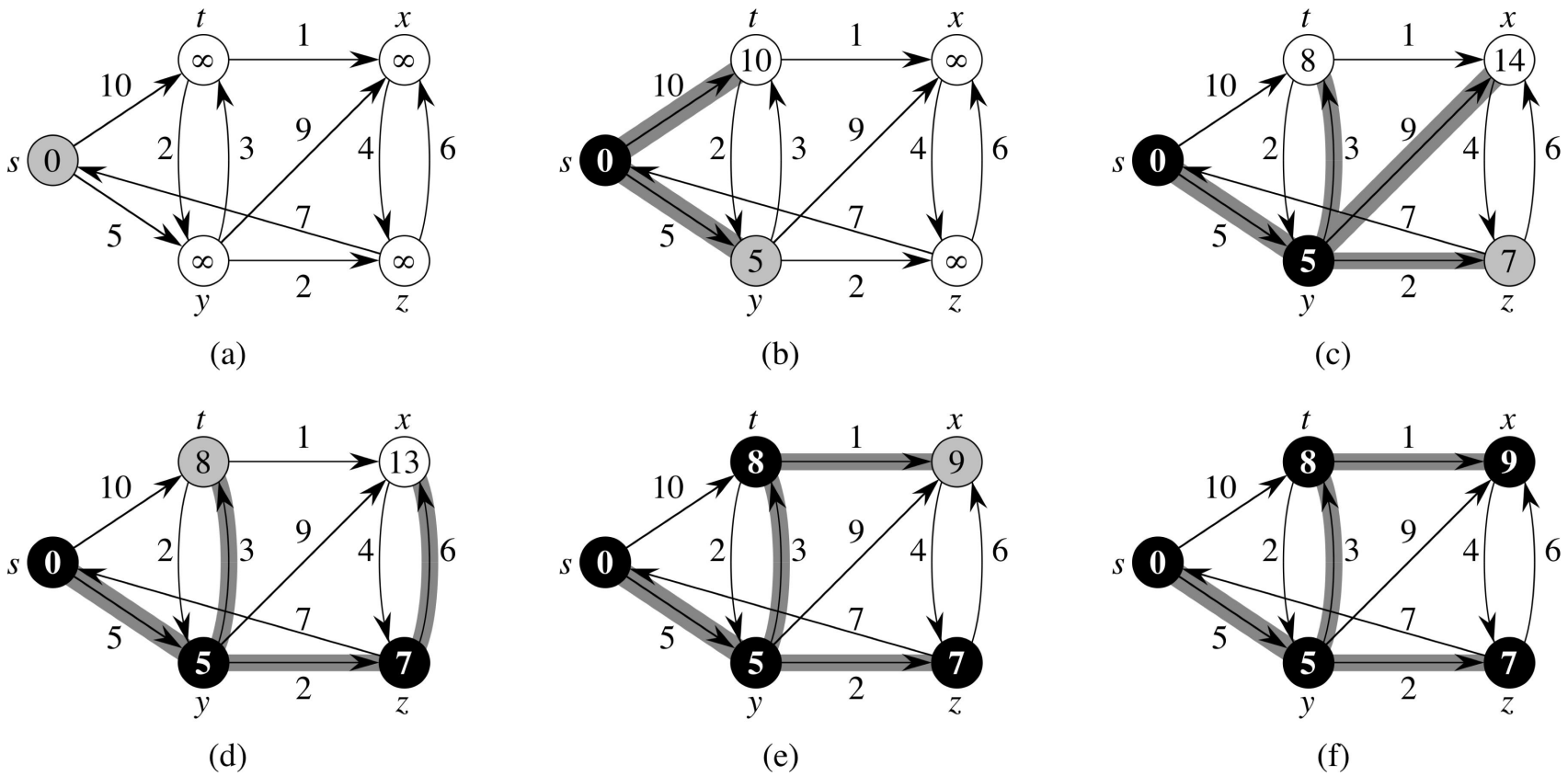
Example 1

Tree	Remaining vertices	Evolving tree
$a(0,-)$	$b(3,a)$, $c(\infty,-)$, $d(7,a)$, $e(\infty,-)$	
$b(3,a)$	$c(3+4,b)$, $d(3+2,b)$, $e(\infty,-)$	
$d(5,b)$	$c(7,b)$, $e(5+4,d)$	
$c(7,b)$	$e(9,d)$	

Example 1

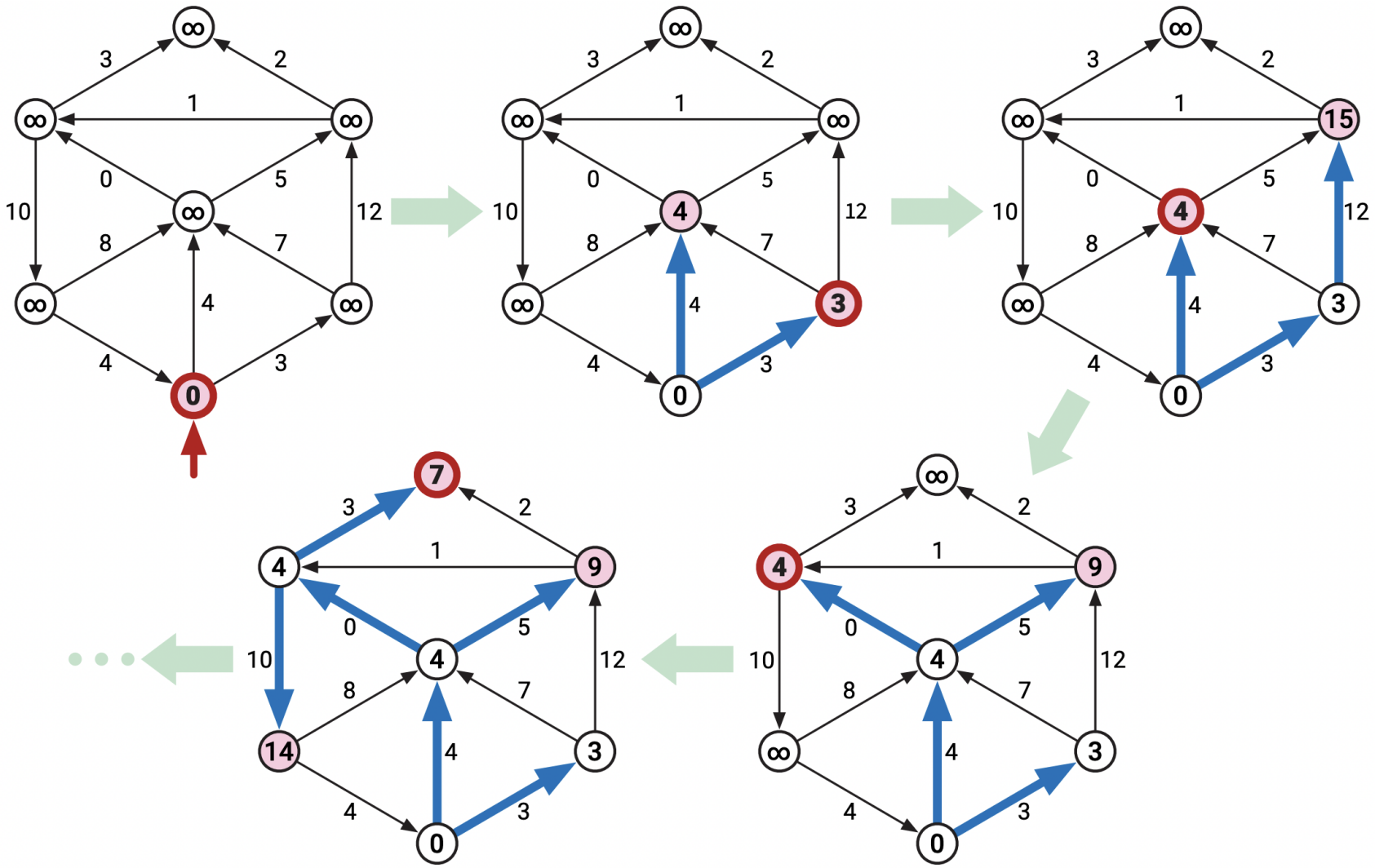
Tree	Remaining vertices	Evolving tree
$a(0,-)$	$b(3,a)$, $c(\infty,-)$, $d(7,a)$, $e(\infty,-)$	
$b(3,a)$	$c(3+4,b)$, $d(3+2,b)$, $e(\infty,-)$	
$d(5,b)$	$c(7,b)$, $e(5+4,d)$	
$c(7,b)$	$e(9,d)$	
$e(9,d)$		

Example 2



Source: CLRS' Introduction to Algorithms textbook

Example 3



Source: Jeff Erickson's Algorithms book. (`https://jeffe.cs.illinois.edu/teaching/algorithms/`)