Homework 8

Lei Zhang

April 13, 2016

1 Exercises 9.1 - 9

a.

Tree vertices	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)	

b.

Tree vertices	Remaining vertices
a(-,-)	**b(a,3)**, c(a,5), d(a,4)
b(a,3)	c(a,5), d(a,4), **e(b,3)**, f(b,6)
e(b,3)	c(a,5), **d(e,1)**, f(e,2), i(e,4)
d(e,1)	**c(d,2)**, f(e,2), i(e,4), h(d,5)
c(d,2)	**f(e,2)**, i(e,4), h(d,5), g(c,4)
f(e,2)	**i(e,4)**, h(d,5), g(c,4), j(f,5)
i(e,4)	h(d,5), g(c,4), **j(i,3)**, l(i,5)
j(i,3)	h(d,5), **g(c,4)**, l(i,5)
g(c,4)	**h(g,3)**, l(i,5), k(g,6)
h(g,3)	**l(i,5)**, k(g,6)
l(i,5)	**k(g,6)**
k(g,6)	

2 Exercises 9.2 - 1

a.

Tree edges	Sorted list of edges
	bc(1), de(2), bd(3), dc(4), ab(5), ad(6), ce(6)
bc(1)	bc(1), **de(2)**, bd(3), dc(4), ab(5), ad(6), ce(6)
de(2)	bc(1), de(2), **bd(3)**, dc(4), ab(5), ad(6), ce(6)
bd(3)	bc(1), de(2), bd(3), dc(4), **ab(5)**, ad(6), ce(6)
ab(5)	

b.

Tree edges	Sorted list of edges
	de(1), cd(2), ef(2), ab(3), be(3), gh(3), ij(3), ad(4), cg(4), ei(4), ac(5), dh(5), fj(5), il(5), bf(6), hi(6), gk(6), hk(7), kl(8), jl(9)
de(1)	de(1), **cd(2)**, ef(2), ab(3), be(3), gh(3), ij(3), ad(4), cg(4), ei(4), ac(5), dh(5), fj(5), il(5), bf(6), hi(6), gk(6), hk(7), kl(8), jl(9)
cd(2)	de(1), cd(2), **ef(2)**, ab(3), be(3), gh(3), ij(3), ad(4), cg(4), ei(4), ac(5), dh(5), fj(5), il(5), bf(6), hi(6), gk(6), hk(7), kl(8), jl(9)
ef(2)	de(1), cd(2), ef(2), **ab(3)**, be(3), gh(3), ij(3), ad(4), cg(4), ei(4), ac(5), dh(5), fj(5), il(5), bf(6), hi(6), gk(6), hk(7), kl(8), jl(9)
ab(3)	de(1), cd(2), ef(2), ab(3), be(3), **gh(3)**, ij(3), ad(4), cg(4), ei(4), ac(5), dh(5), fj(5), il(5), bf(6), hi(6), gk(6), hk(7), kl(8), jl(9)
gh(3)	de(1), cd(2), ef(2), ab(3), be(3), gh(3), **ij(3)**, ad(4), cg(4), ei(4), ac(5), dh(5), fj(5), il(5), bf(6), hi(6), gk(6), hk(7), kl(8), jl(9)
ij(3)	de(1), cd(2), ef(2), ab(3), be(3), gh(3), ij(3), ad(4), **cg(4)**, ei(4), ac(5), dh(5), fj(5), il(5), bf(6), hi(6), gk(6), hk(7), kl(8), jl(9)
cg(4)	de(1), cd(2), ef(2), ab(3), be(3), gh(3), ij(3), ad(4), cg(4), **ei(4)**, ac(5), dh(5), fj(5), il(5), bf(6), hi(6), gk(6), hk(7), kl(8), jl(9)
ei(4)	de(1), cd(2), ef(2), ab(3), be(3), gh(3), ij(3), ad(4), cg(4), ei(4), **ac(5)**, dh(5), fj(5), il(5), bf(6), hi(6), gk(6), hk(7), kl(8), jl(9)
ac(5)	de(1), cd(2), ef(2), ab(3), be(3), gh(3), ij(3), ad(4), cg(4), ei(4), ac(5), dh(5), fj(5), **il(5)**, bf(6), hi(6), gk(6), hk(7), kl(8), jl(9)
il(5)	de(1), cd(2), ef(2), ab(3), be(3), gh(3), ij(3), ad(4), cg(4), ei(4), ac(5), dh(5), fj(5), il(5), bf(6), hi(6), **gk(6)**, hk(7), kl(8), jl(9)
gk(6)	

3 Exercises 9.2 - 2

a.

True. The first step of applying Kruskal's algorithm is to this edge to the result graph.

b.

False. If the edges in a graph all have the same edges, which means every edge is the minimum-weight edge, a minimum-weight edge is not guaranteed to be among edges of each minimum spanning tree of the graph.

c.

True.

Assume there are two different minimum spanning trees A and B, e is in A and not in B. Add e to B, there will be a circle C which contains e. There must be a path f in C that has the weight less than the weight of c because B is a minimum spanning tree. Now removing f, the result graph is connected and has a less weight than B, which is contradict to the assumptions. Thus, the statement is true.

d.

False. Consider a graph with all same weight of edges, all the spanning trees are the mininum spanning trees, and these trees can be different.

4 Exercises 9.3 - 1

a.

No adjustments need to be made.

b.

Just applying Dijkstra's algorithm on either one of the two vertices, the shortest path between two given vertices should be included in the final results.

c.

First applying single-source Dijkstra's algorithm on the graph with reversed directions of original graph, treating the destination vertex as the source vertex. At last, reverse the directions back.

d.

Create a new graph with two times edges as the number of vertex in original graph. Each edge in the new graph is transformed from one vertex in the original graph. The weight of the edges in the new graph is equal to the weight of vertices in the original graph. Now Dijkstra's algorithm can be applied to the new graph. The final result can be transformed to fit the original graph according to the relationship between the two graphs.

5 Exercises 9.3 - 2

a.

Tree vertices	Remaining vertices
a(-,0)	b(-,∞) c(—,∞) d(a,7) e(-,∞)
d(a,7)	b(d,7+2) c(d,7+5) e(-,∞)
b(d,9)	c(d,12) e(-,∞)
c(d,12)	e(c,12+6)
e(c,18)	

from a to d: a - d of length 7
from a to b: a - d - b of length 9
from a to c: a - d - c of length 12
from a to e: a - d - c - e of length 18

b.

Tree vertices	Remaining vertices
a(-,0)	b(a,3) c(a,5) d(a,4)
b(a,3)	c(a,5) d(a,4) e(b,3+3) f(b,3+6)
d(a,4)	c(a,5) e(d,4+1) f(a,9) h(d,4+5)
c(a,5)	e(d,5) f(a,9) h(d,9) g(c,5+4))
e(d,5)	f(e,5+2) h(d,9) g(c,9) i(e,5+4)
f(e,7)	h(d,9) g(c,9) i(e,9) j(f,7+5)
h(d,9)	g(c,9) i(e,9) j(f,12) k(h,9+7))
g(c,9)	i(e,9) j(f,12) k(g,9+6)
i(e,9)	j(f,12) k(g,15) l(i,9+5)
j(f,12)	k(g,15) l(i,14)
l(i,14)	k(g,15)
k(g,15)	

from a to b: a - b of length 3
from a to d: a - d of length 4
from a to c: a - c of length 5
from a to e: a - d - e of length 5
from a to f: a - d - e - f of length 7
from a to h: a - d - h of length 9

from a to g: a - c - g of length 9
from a to i: a - d - e - i of length 9
from a to j: a - d - e - f - j of length 12
from a to l: a - d - e - i - l of length 14
from a to k: a - c - g - k of length 15