HW 6 Problem 2:

A transportation problem:

Plant ↓	DC 1	2	3	4	Supply ↓
1	800	1300	400	700	12
2	1100	1400	600	1000	17
3	600	1200	800	900	11
Demand →	10	10	10	10	

Here, supply = demand, so a solution exists (see pyomo code example)

Now: suppose we can increase the supply in one of the plants by 1 unit, and thus the demand in one of the DCs by one unit. Which ones should we choose?

(look at shadow prices: $y_{\text{plant}1} = -100$, $y_{\text{plant}2} = 0$, $y_{\text{plant}3} = -100$,

$y_{DC1} = 500$, $y_{DC2} = 800$, $y_{DC3} = 400$, $y_{DC4} = 550$

=> Increase in plants 1 or 3 and ship to DC 3. Here, the cost increase will be the lowest, namely $-100 + 400 = 300$

Now: Demand in Center 1 goes up to 15 units, but production cannot be increased, so some centers will be undersupplied.

Problem: We need total supply = total demand for solution to exist.

Solution: Introduce "dummy source" with 0 associated shipping cost.

Plant ↓	DC 1	2	3	4	Supply ↓
1	800	1300	400	700	12
2	1100	1400	600	1000	17
3	600	1200	800	900	11
4	0	0	0	0	5
Demand →	15	10	10	10	

New solution: see promo code (DC2 will be undersupplied).

The shortest path problem again:

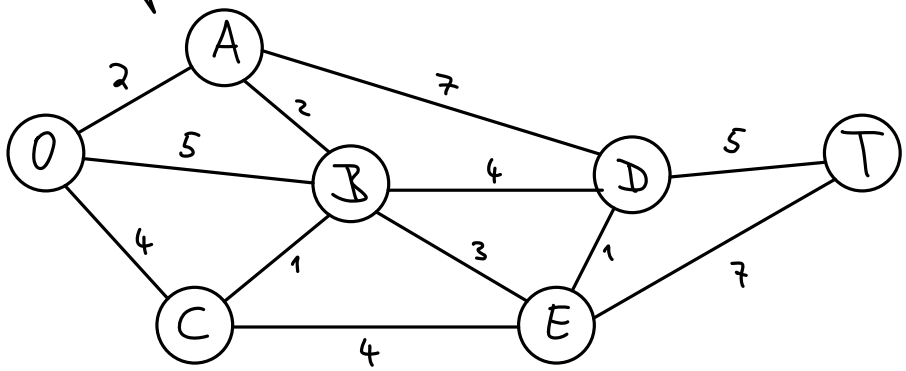
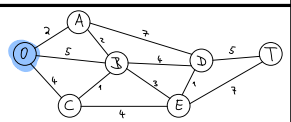
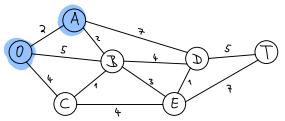
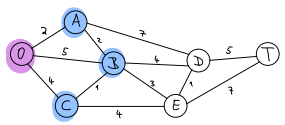
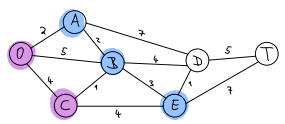
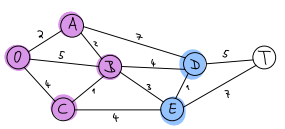


Table for our example:

Iteration step n (total # of solved nodes)	Solved nodes (directly connected to unsolved nodes)	Closest unsolved node	Total distance	n-th nearest node (= new solved node in next step)	Min. distance	Last connection
1	0 	A	2	A	2	0-A
2, 3	0 A 	C B	4 2+2=4	C B	4 4	0-C A-B
4	A B C 	D E E	2+7=9 4+3=7 4+4=8	E	7	B-E
5	A B E 	D D D	2+7=9 4+4=8 7+1=8	D D	8 8	B-D E-D
6	D E 	T T	8+5=13 7+7=14	T	13	D-T

Purple: Solved nodes that are not connected to any unsolved nodes (and can thus be neglected in further steps

Blue: Solved nodes that are connected to unsolved nodes

=> Shortest paths: 0-A-B-E-D-T and 0-A-B-D-T with total distance 13

For Problem 2 (Homework 7):

Minimum Spanning Tree problem:

Similar to Shortest Path problem, but now path needs to connect each pair of nodes, with minimal cost (say, cost is proportional to distance here).

Note: for n nodes we need to find n-1 links, i.e., a tree connecting all nodes
otherwise we could always delete a link to make cost smaller

Algorithm:
- start with any node, add link to nearest node
- connect linked notes to next nearest node; repeat

In our example: