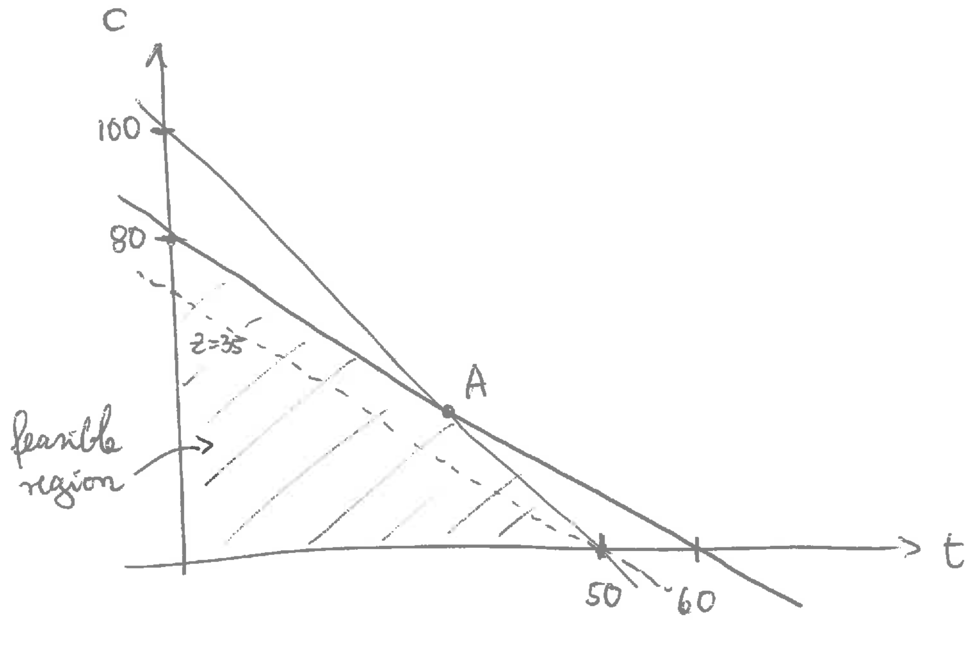
1. A furniture company produces tables and chairs. Each table takes 4 hours of carpentry and 2 hours of painting. Each chair requires 3 hours of carpentry and 1 hour of painting. During the current week, there are 240 hours for carpentry work and 100 hours for painting available. Each table is sold at a profit of €7 and each chair at a profit of €5.

   Find the number of tables and chairs to produce in order to maximize profit. Use the graphical method to solve the resulting linear programming problem. (20)

We need to solve:

maximize $z = 7t + 5c$

subject to
$$4t + 3c \le 240$$
$$2t + c \le 100$$
$$t, c \ge 0$$



The slope of $z = \text{const}$ is between the slope of the two other bounding lines, so optimal value is taken at A. Coordinates of A:

$$\left(\begin{array}{cc|c} 4 & 3 & 240 \\ 2 & 1 & 100 \end{array}\right) \to \left(\begin{array}{cc|c} 1 & \frac{3}{4} & 60 \\ 0 & -\frac{1}{2} & -20 \end{array}\right) \to \left(\begin{array}{cc|c} 1 & \frac{3}{4} & 60 \\ 0 & 1 & 40 \end{array}\right) \to \left(\begin{array}{cc|c} 1 & 0 & 30 \\ 0 & 1 & 40 \end{array}\right)$$

$\Rightarrow t = 30, \; S = 40, \quad z = 7 \cdot 30 + 5 \cdot 40 = 410$

2. Perform one step of the simplex algorithm (choice of entering variable. choice of leaving variable, elimination. check of termination condition) on the following tableau:

x_1	x_2	x_3	s_1	s_2	s_3	
2	1	0	1	0	0	10
1	2	−2	0	1	0	20
0	1	2	0	0	1	5
−1	1	−2	0	0	0	0

(5+5+5+5)

- x_3 is entering (largest negative coefficient in objective function row)
- Pivot in x_3-column must be in R3 (no other positive coefficient available!)
- New simplex tableau: R2+ R3 → R2, R3+R4 → R4, R3/2 → R3

x_1	x_2	x_3	s_1	s_2	s_3	
2	1	0	1	0	0	10
1	3	0	0	1	1	25
0	$\frac{1}{2}$	1	0	0	$\frac{1}{2}$	$\frac{5}{2}$
−1	2	0	0	0	1	5

- There is still a negative coefficient in the objective function row → no termination, x_1 will be new entering variable.

3. The Brick & Mortar Construction Co. needs to excavate a construction pit with a total volume of $100\,\mathrm{m}^3$ for a building extension. The construction site is close to trees and other buildings, so a large excavator could only be used for the first $90\,\mathrm{m}^3$ at a cost of $1\,€/\mathrm{m}^3$ plus a € 1000 flat fee. A mini excavator could be used for all but the last $1\,\mathrm{m}^3$ at a cost of $10\,€/\mathrm{m}^3$ plus a € 100 flat fee. Manual digging can be performed at $100\,€/\mathrm{m}^3$ with no flat fee.

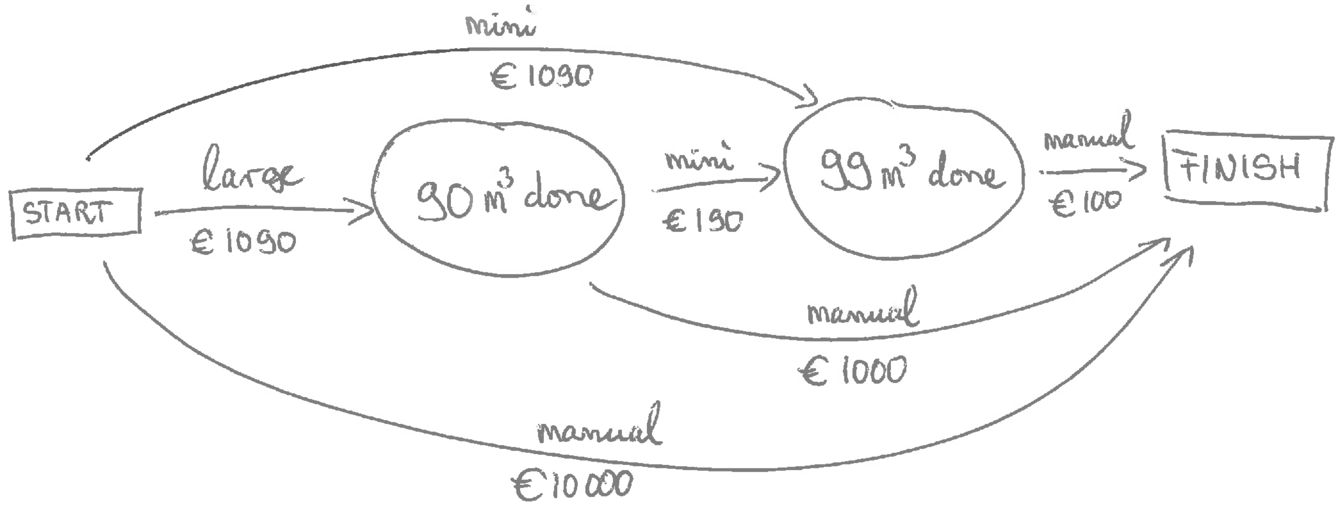
Consider all possible options to divide the work between a large excavator, a mini excavator, and manual digging. The objective is minimizing the total cost.

(a) Recognize this problem as a shortest path problem. Draw the network and assign the cost to each arc.

(b) Solve the problem any way you like.

(10+10)

(a) Note first that if equipment is brought on site, it will be used to the fullest extent, with preference to the equipment with the smallest unit cost. Thus, the network nodes are the maximal completion points for each equipment.



(b) By inspection: The "upper path" is the "shortest": use the mini excavator for the first 99 m³ and manual digging for the last 1 m³ at total cost 1190 €.

4. The Pyomo program on the last page is solving a network optimization problem.

(a) What problem is the program solving? As part of your answer, explain the meaning of the given data and of the objective function!

(b) Draw the network which corresponds to the given data.

(c) Give an interpretation of the dual variables. Draw the dual information obtained from the program output into your graph.

(d) You want to increase the objective function by exactly one. Suggest a change of the data to achieve this.

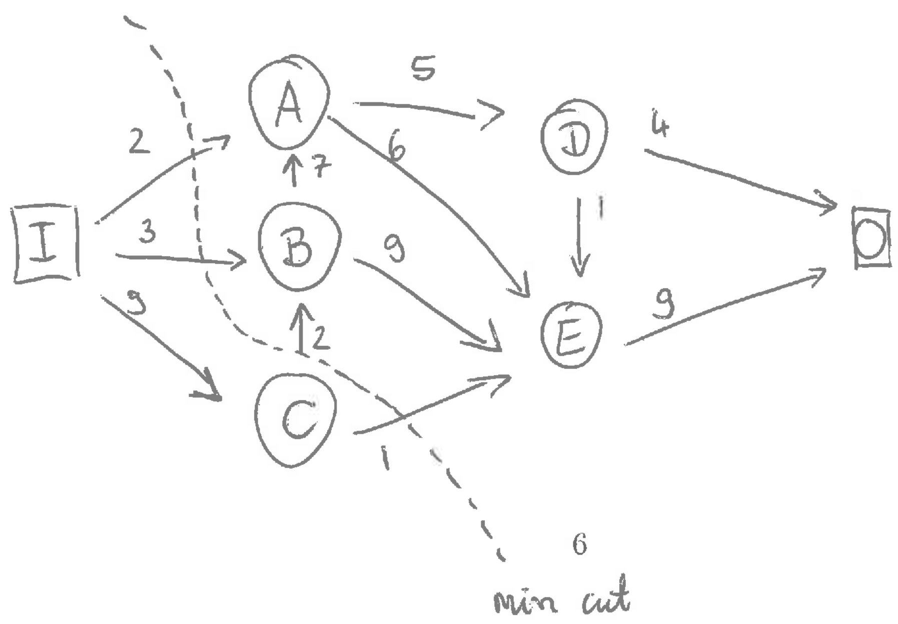
(e) Can you be sure, without re-solving, that your suggested change will achieve its goal? Explain! (Hint: look at the values of the slack variables!)

(f) Suppose the network nodes are "leaky," losing exactly one unit of flow each. Suggest a change to the code to account for this.

(5+5+5+5+5+5)

(a) It's a maximum flow problem. U gives the capacity limit for each arc and the objective function asks for maximizing the flow out of the outflow node 'O'. The inflow node is 'I' and all other nodes are transshipment nodes.

(b)



(c) The dual variables give the marginal increase of outflow if the capacity of the corresponding arc is raised by one unit. As such, they are either 1 or 0 (in the absence of degeneracy). The dual variables with value 1 define a minimal cut: a set of arcs such that each path from I to O contains exactly one of them and their joint capacity equals the max flow.

The min cut is indicated by a dotted line above.

(d) Increase the capacity on any arc on the min cut by 1.

(e) Yes. In general, there could be another min cut of the same capacity, so that the increase in (d) might not help. Here, however, we see that the slack on all arcs off the min cut is ≥ 1, so the rest of the network is able to carry the increased flow.

(f) Add a loss term to each flow balance constraint on the transshipment nodes.
(If O is also "leaky", subtract 1 from the objective function, too.)

See code.

In [1]:
```
from pyomo.environ import *
from pyomo.opt import *
opt = solvers.SolverFactory("glpk")
```

In [2]:
```
T = ['A', 'B', 'C', 'D', 'E']

U = {('I','A'):2,
     ('I','B'):3,
     ('I','C'):9,
     ('A','D'):5,
     ('B','A'):7,
     ('B','E'):9,
     ('E','O'):9,
     ('C','B'):2,
     ('C','E'):1,
     ('D','O'):4,
     ('D','E'):1,
     ('A','E'):6}

A = list(U.keys())
```

In [3]:
```
model = ConcreteModel()
model.f = Var(A, within=NonNegativeReals)

def flow_rule(model, n):
    InFlow  = sum(model.f[i,j] for (i,j) in A if j==n)
    OutFlow = sum(model.f[i,j] for (i,j) in A if i==n)
    return InFlow == OutFlow + 1

model.transshipment = Constraint(T, rule=flow_rule)

def capacity_rule(model, i, j):
    return model.f[i,j] <= U[i,j]

model.capacity = Constraint(A, rule=capacity_rule)

model.objective = Objective(expr = sum(model.f[i,j] for (i,j) in A if j
=='O') - 1, sense=maximize)
```

For (f)

For (f)

In [4]:
```
model.dual = Suffix(direction=Suffix.IMPORT)
results = opt.solve(model)
model.objective.expr()
```

Out[4]: 8.0

In [5]:
```
for (i,j) in A:
    print ((i,j),
           model.dual[model.capacity[i,j]],
           model.capacity[i,j].uslack())
```

```
('D', 'E') 0.0 1.0
('D', 'O') 0.0 2.0
('I', 'C') 0.0 6.0
('B', 'E') 0.0 4.0
('E', 'O') 0.0 3.0
('C', 'B') 1.0 0.0
('C', 'E') 1.0 0.0
('I', 'B') 1.0 0.0
('I', 'A') 1.0 0.0
('A', 'D') 0.0 3.0
('B', 'A') 0.0 7.0
('A', 'E') 0.0 6.0
```