1. Consider linear programming problem

$$\text{maximize } z = 3x_1 + 2x_2 + x_3$$

subject to

$$x_1 + x_2 + x_3 \le 2\,.$$
$$3x_1 + x_2 \le 1\,.$$
$$x_1, x_2, x_3 \ge 0\,.$$

(a) Write out the *dual* problem to this LP.

(b) Solve the dual problem using the graphical method.

(c) Which constraints in the dual formulation are binding?

(d) Which variables in the primal formulation are basic?

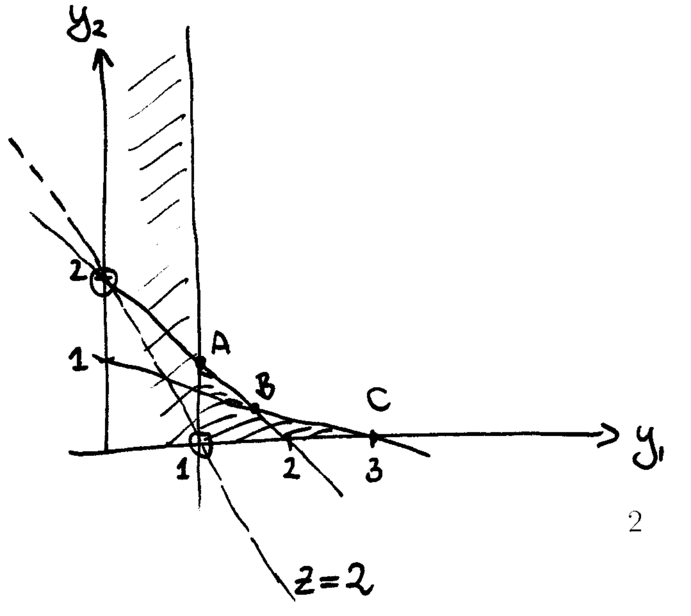
(e) Using your answer to (d), verify your solution by a direct computation on the *primal* problem.

(5+10+3+3+4)

(a) minimize $z = 2y_1 + y_2$

subject to
$$y_1 + 3y_2 \ge 3 \quad \text{(i)}$$
$$y_1 + y_2 \ge 2 \quad \text{(ii)}$$
$$y_1 \ge 1 \quad \text{(iii)}$$
$$y_1, y_2 \ge 0$$

(b)



As the slope of $z = \text{const}$ is steeper than that of the first two constraint equations, the minimum of z must occur at point $A = (1,1)$ where $z = 3$.

(c) ~~At~~ point A, constraints (ii) and (iii) are binding.

(d) Binding constraints in the dual correspond to basic variables in the primal. Thus, x_2 and x_3 are basic.

(e) Since x_1 is non-basic, $x_1 = 0$ at the optimum. Thus, the optimal corner point in the primal is given by

$$\left.\begin{array}{l} x_2 + x_3 = 2 \\ x_2 \qquad = 1 \end{array}\right\} \Rightarrow x_3 = 1$$

Then $3x_1 + 2x_2 + x_3 = 3$, which coincides (as it must by strong duality) with the optimal dual objective function.

2. Consider the linear programming problem

$$\text{maximize } z = 2x_1 + x_2$$

subject to

$$-x_1 + x_2 \leq 1\,,$$
$$x_1 - 2x_2 \leq 2\,,$$
$$x_1, x_2 \geq 0\,.$$

Determine the solvability of this problem and find the solution, if it exists, *using the simplex method.* (20)

Introducing slack variables, the problem corresponds to the simplex tableau

x_1	x_2	s_1	s_2	
-1	1	1	0	1
1	-2	0	1	2
-2	-1	0	0	0

Make x_1 basic:

x_1	x_2	s_1	s_2	
0	-1	1	1	3
1	-2	0	1	2
0	-5	0	2	4

x_2 should be new entering variable, but there isn't a feasible pivot in the column: x_2 can be made arbitrarily large, so the feasible region is unbounded and the problem does not have a finite maximum.

3. You are tasked with setting up an image processing pipeline for a medical laboratory. Images come from a database in LIFF, a format specifically designed for microscope image processing. The lab would like to make these images available in PNG format via their web server. Unfortunately, you do not have an LIFF-to-PNG converter. However, you have access to a large set of image converters which take the following average times (in milliseconds) to convert an image. Missing format converters are indicated by a dash.

	Destination Format				
Source Format	LIFF	PNG	JPG	BPG	BMP
LIFF	-	-	800	700	300
PNG	500	-	600	-	-
JPG	-	900	-	700	
BPG	700	300	100	-	600
BMP	-	600	200	200	-

Find the most efficient way to realize the conversion from LIFF to PNG. Illustrate your solution by drawing an appropriate network representation of the problem. (20)

$$\begin{tikzcd}
& \text{JPG} \arrow[d, shift right, "7"'] \arrow[rd, bend left, "9"] & \\
\text{LIFF} \arrow[ru, "8"] \arrow[r, shift left, "7"] \arrow[rd, "3"'] & \text{BPG} \arrow[l, shift left, "7"] \arrow[u, shift right, "1"'] \arrow[r, "3"] \arrow[d, shift right, "6"'] & \text{PNG} \arrow[lu, bend left, "6"] \\
& \text{BMP} \arrow[u, shift right, "2"'] \arrow[ru, "6"'] \arrow[uu, bend right=60, "2"'] &
\end{tikzcd}$$

By inspection, the shortest path is LIFF → BMP → BPG → PNG

at 300 + 200 + 300 = 800 ms.

4. The Pyomo program on the last page is showing a min-cost flow problem similar to "Distribution Unlimited" Example from Hillier and Lieberman.

(a) Draw the network that is described by the data in the code.

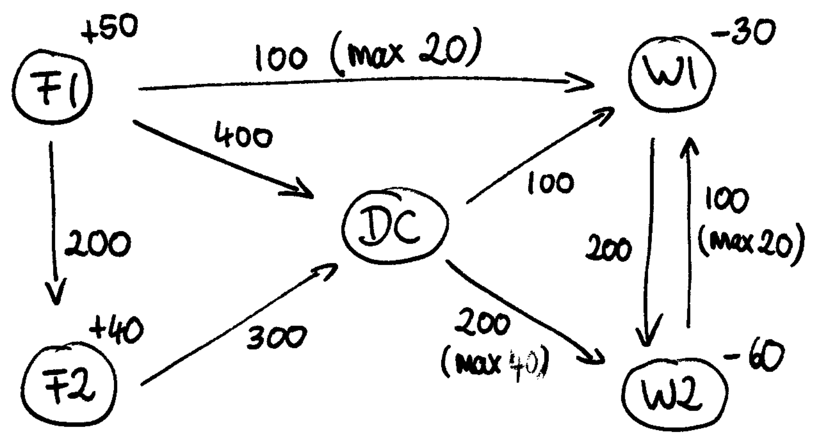
(b) Management is considering investing in expanding the transportation capacities on any of the currently capacity-constrained routes. Which route should be invested in first, assuming that it costs the same on all routes to per unit increase of the maximal capacity, and that the unit cost of transportation will not change?

(c) The route from the distribution center DC to the warehouse is cut due to road damage. Is it possible to still supply the warehouses? If not, how many units can you still deliver to the warehouses? Which problem did you solve to find out?

(d) One truck can be loaded with 10 units of the product. Thus, it is desirable that the optimal solution will be in multiples of 10. Which property of the data ensures that this is possible?

(10+5+5+5)

(a)



numbers on arcs: unit cost of transport (max number of units)

numbers at nodes: number of units produced (+)
or requested (-)

(b) The route with the largest negative shadow price,

i.e. F1 → W1 with a shadow price of −400.

(c) No, the min-cut of the remaining network is F1 → W1

and DC → W2 with a total capacity of 20+40 = 60.

I.e, only a total of 60 units can be supplied.

This can be seen as the dual of a max flow problem,

e.g. from F1 to W2.

(d) The net inflows (production or demand quantities) and the

capacity constraints are integer if expressed as multiples

of 10 units. By the integrality property of the min-cost-flow

problem, a solution with integer multiples of 10 units

exists.

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

In [2]:
```
b = {'F1':50,
     'F2':40,
     'DC':0,
     'W1':-30,
     'W2':-60}

C = {('F1','F2'):200,
     ('F1','DC'):400,
     ('F1','W1'):100,
     ('F2','DC'):300,
     ('DC','W1'):100,
     ('DC','W2'):200,
     ('W1','W2'):200,
     ('W2','W1'):100}

U = {('DC','W2'):40,
     ('F1','W1'):20,
     ('W2','W1'):20}

N = list(b.keys())
A = list(C.keys())
V = 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 + b[n] == OutFlow

model.flow = Constraint(N, rule=flow_rule)

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

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

model.cost = Objective(expr = sum(model.f[a]*C[a] for a in A), sense=minimize
)
```

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

Out[4]:
```
{('DC', 'W1'): 30.0,
 ('DC', 'W2'): 40.0,
 ('F1', 'DC'): 30.0,
 ('F1', 'F2'): 0.0,
 ('F1', 'W1'): 20.0,
 ('F2', 'DC'): 40.0,
 ('W1', 'W2'): 20.0,
 ('W2', 'W1'): 0.0}
```

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

```
('DC', 'W2') -100.0
('F1', 'W1') -400.0
('W2', 'W1') 0.0
```

MAKEUP EXAM

3. *Rock-paper-scissors*[1] is a hand game usually played between two people, in which each player simultaneously forms one of three shapes with an outstretched hand. These shapes are "rock" (a closed fist), "paper" (a flat hand), and "scissors" (a fist with the index and middle fingers extended, forming a V). "Scissors" is identical to the two-fingered V sign except that it is pointed horizontally instead of being held upright in the air. A simultaneous, zero-sum game, it has only two possible outcomes other than a tie: one of the two players wins, and the other player loses.

   A player who decides to play rock will beat another player who has chosen scissors ("rock blunts scissors"), but will lose to one who has played paper ("paper covers rock"); a play of paper will lose to a play of scissors ("scissors cut paper"). If both players choose the same shape, the game is tied.

   (a) Write out a linear program, mimicking the "drug trafficking" expample we discussed in class, to find the optimal strategy. Assume that the payoff for the winning player is 1, the payoff for the losing player is −1, and the payoff for both in the case of a tie is 0.

   (b) What solution do you expect? (There is no need to solve the linear program from (a), but you are expected to give an explicit, well-reasoned argument.)

(10+10)

(a) maximize w

subject to $w \le 0 \cdot p_1 + 1 \cdot p_2 - 1 \cdot p_3$ (opponent chooses rock)

$w \le -1 \cdot p_1 + 0 \cdot p_2 + 1 \cdot p_3$ (opponent chooses paper)

$w \le 1 \cdot p_1 - 1 \cdot p_2 + 0 \cdot p_3$ (opponent chooses scissors)

$p_1 + p_2 + p_3 = 1, \quad p_1, p_2, p_3 \geq 0$

w: expected payoff

p_1, p_2, p_3: probabilities of choosing "rock", "paper", "scissors", resp.

[1] Text adapted from Wikipedia, https://en.wikipedia.org/wiki/Rock-paper-scissors

(b) Since the payoffs are completely symmetric between all choices, the solution must be symmetric, too, so $p_1 = p_2 = p_3 = \frac{1}{3}$.

MAKEUP EXAM

4. The Pyomo program on the following pages is a variation of the "Reliable Construction Co." prototype example from Hillier and Lieberman which we discussed in class.

   (a) Draw the network that is described by the data in the code.

   (b) It turns out that lowering the required completion deadline to 27 (weeks, say) leads to an infeasible linear program. Can you conclude that 28 weeks is the shortest possible completion time?

   (c) Give an alternative network programming formulation to determine the shortest possible completion time. Be very specific about the data which goes into the network program.

   (d) Describe the meaning of the dual variables (in words).

   (e) With extra effort, it is possible to start task F one week before task E is completed. How much extra cost would you be willing to incur for doing so under the premise that the cost from the current model output must not be exceeded? Will the total time to completion decrease if you do this?

(5+5+5+5+5)

(a)

$$\begin{tikzcd}
 & \boxed{\text{Start}} \arrow[d] & & & & \\
 & A \arrow[d] & & & & \\
 & B \arrow[d] & & & & \\
 & C \arrow[dl] \arrow[d] \arrow[r] & I \arrow[drr] & & & \\
D \arrow[d] & E \arrow[r] \arrow[d] & F \arrow[rr] & & J \arrow[dl] \arrow[dr] & \\
G \arrow[r] & H \arrow[d] & & K \arrow[dr] & & L \arrow[dl] \\
 & M \arrow[d] & & & N \arrow[dlll] & \\
 & \boxed{\text{Finish}} & & & &
\end{tikzcd}$$

(b) + (c): The minimal completion time is the longest-path-problem for this network where the "distances" are the completion times for each task after maximal crashing. As such, it satisfies the integrality property. Since the minimal completion times are integers, the critical path must have integer length (this is actually obvious in this case!).

Thus, the answer to (b) is yes – there cannot be a solution between 27 and 28 weeks completion time.

(d) The time duals give the marginal change in crashing expense per unit of time that the dependent task can be started earlier.

The crash rule duals give the marginal change in crashing expense per unit of increase of the feasible time saved.

(e) The cost should not exceed the shadow price on this arc, which is 160 according to the program output.

This will decrease only cost, not completion time, as E-F is not on the critical path.

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

In [2]:
```
T = {'Start':0,
     'A':2,
     'B':4,
     'C':10,
     'D':6,
     'E':4,
     'F':5,
     'G':7,
     'H':9,
     'I':7,
     'J':8,
     'K':4,
     'L':5,
     'M':2,
     'N':6,
     'Finish':0}

R = {'Start':0,
     'A':1,
     'B':2,
     'C':3,
     'D':2,
     'E':1,
     'F':2,
     'G':3,
     'H':3,
     'I':2,
     'J':2,
     'K':1,
     'L':2,
     'M':1,
     'N':3,
     'Finish':0}

C = {'Start':0,
     'A':100,
     'B':50,
     'C':80,
     'D':40,
     'E':160,
     'F':40,
     'G':40,
     'H':60,
     'I':30,
     'J':30,
     'K':40,
     'L':50,
     'M':100,
     'N':60,
     'Finish':0}
```

In [3]:
```
P = {'Start':[],
     'A':['Start'],
     'B':['A'],
     'C':['B'],
     'D':['C'],
     'E':['C'],
     'F':['E'],
     'G':['D'],
     'H':['G','E'],
     'I':['C'],
     'J':['F','I'],
     'K':['J'],
     'L':['J'],
     'M':['H'],
     'N':['K','L'],
     'Finish':['M','N']}

N = list(T.keys())
A = [(j,i) for i in N for j in P[i]]
```

In [4]:
```
model = ConcreteModel()
model.t = Var(N, within=NonNegativeReals)
model.x = Var(N, within=NonNegativeReals)
```

In [5]:
```
def time_rule(model, i, j):
    return model.t[j] >= model.t[i] + T[i] - model.x[i]

model.time = Constraint(A, rule=time_rule)

model.start = Constraint(expr = model.t['Start'] == 0)
model.finish = Constraint(expr = model.t['Finish'] <= 28)

def crash_rule(model, n):
    return model.x[n] <= R[n]

model.crash = Constraint(N, rule=crash_rule)

model.cost = Objective(expr = sum(model.x[n]*C[n] for n in N), sense=minimiz
e)
```

In [6]:
```
model.dual = Suffix(direction=Suffix.IMPORT)
results = opt.solve(model)
model.x.get_values()
```

Out[6]:
```
{'A': 1.0,
 'B': 2.0,
 'C': 3.0,
 'D': 2.0,
 'E': 1.0,
 'F': 2.0,
 'Finish': 0.0,
 'G': 3.0,
 'H': 1.0,
 'I': 1.0,
 'J': 2.0,
 'K': 1.0,
 'L': 2.0,
 'M': 0.0,
 'N': 3.0,
 'Start': 0.0}
```

In [7]:
```
model.t.get_values()
```

Out[7]:
```
{'A': 0.0,
 'B': 1.0,
 'C': 3.0,
 'D': 10.0,
 'E': 10.0,
 'F': 13.0,
 'Finish': 28.0,
 'G': 14.0,
 'H': 18.0,
 'I': 10.0,
 'J': 16.0,
 'K': 22.0,
 'L': 22.0,
 'M': 26.0,
 'N': 25.0,
 'Start': 0.0}
```

In [8]:
```
model.cost.expr()
```

Out[8]:
```
1350.0
```

In [9]:
```
for (i,j) in A:
    print ((i,j), model.dual[model.time[(i,j)]])
```

```
('Start', 'A') 0.0
('A', 'B') -250.0
('B', 'C') -250.0
('C', 'D') -60.0
('C', 'E') -160.0
('E', 'F') -160.0
('D', 'G') -60.0
('G', 'H') -60.0
('E', 'H') 0.0
('C', 'I') -30.0
('F', 'J') -160.0
('I', 'J') -30.0
('J', 'K') -140.0
('J', 'L') -50.0
('H', 'M') -60.0
('K', 'N') -140.0
('L', 'N') -50.0
('M', 'Finish') -60.0
('N', 'Finish') -190.0
```

In [10]:
```
for n in N:
    print(n, model.dual[model.crash[n]])
```

```
Start 0.0
A -150.0
B -200.0
C -170.0
D -20.0
E 0.0
F -120.0
G -20.0
H 0.0
I 0.0
J -160.0
K -100.0
L 0.0
M 0.0
N -130.0
Finish 0.0
```