MINI PROJECT 2

Part a

Condition

```
links = [(1,2),(2,3),(3,4),(1,3),(2,4)]
capacities = {(1,2):6000, (2,3):6000, (3,4):6000, (1,3):3000, (2,4):3000}
travel_times = {(1,2):3, (2,3):3, (3,4):3, (1,3):10, (2,4):10}
demands = {1:5000, 2:3000, 3:-5000, 4:-3000}
```

Decision Variable:

$ f_{12}, f_{23}, f_{13}, f_{24}, f_{34} $

Objective function

$$3f_{12} + 3f_{34} + 3f_{23} + 10f_{13} + 10f_{24}$$

Cost vector

```
c = [travel_times[link] for link in links]
```

Constraints

Capacity constraint

```
A_ub = []
b_ub = []
for link in links:
    A_row = [int(link == l) for l in links]
    A_ub.append(A_row)
    b_ub.append(capacities[link])
```

Flow conservation

```
A_eq = []
b_eq = []
for node in [1,2,3]:
    A_row = [int(link[0] == node) - int(link[1] == node) for
link in links]
    A_eq.append(A_row)
    if node not in demands:
        b_eq.append(0)
    else:
        b_eq.append(demands[node])
```

Solution

```
res = linprog(c=c, A_ub=A_ub, b_ub=b_ub, A_eq = A_eq,
b_eq=b_eq)
res.x
```

```
array([3866.70844779, 5999.99999497, 2133.29154718,
1133.29154802, 866.70845031])
```

Thus, $ f_{12} = 3866.7, f_{23} = 6000, f_{13} = 2133.3, f_{24} = 1133.3, f_{34} = 866.7 $

Part b

Additional constraints

All the flow from node 1 must be equal to the flow from node 1 to node 3

All the flow from node 2 must be equal to the flow from node 2 to node 4

```
A_row = [ sum(int(link == (1,j))-int(link == (1,3)) for j in
[2,3,4]) for link in links]
A_eq.append(A_row)
b_eq.append(0)
A_row = [ sum(int(link == (2,j))-int(link == (2,4)) for j in
[1,3,4]) for link in links]
A_eq.append(A_row)
b_eq.append(0)
```

Solution

```
array([3333.33332771, 4222.22221511,  888.88888739,
1666.66666386, 2111.11110755])
```

Thus,

$$f_{12} = 3333.3, f_{23} = 4222.2, f_{13} = 888.9, f_{24} = 1666.7, f_{34} = 2111.1$$

Code

a)

```
from scipy.optimize import linprog
```

```
links = [(1,2),(2,3),(3,4),(1,3),(2,4)]
capacities = {(1,2):6000, (2,3):6000, (3,4):6000, (1,3):3000,
(2,4):3000}
travel_times = {(1,2):3, (2,3):3, (3,4):3, (1,3):10, (2,4):10}
```

```
demands = {1:5000, 2:3000, 3:-5000, 4:-3000}
```

```
c = [travel_times[link] for link in links]
```

```
c
```

```
[3, 3, 3, 10, 10]
```

```
A_ub = []
b_ub = []
```

```
# capacity constrain
for link in links:
    A_row = [int(link == l) for l in links]
    A_ub.append(A_row)
    b_ub.append(capacities[link])
    A_ub.append([-int(link == l) for l in links])
    b_ub.append(capacities[link])
```

```
A_eq = []
b_eq = []
```

```
# flow conservation
for node in [1,2,3]:
    A_row = [int(link[0] == node) - int(link[1] == node) for
link in links]
    A_eq.append(A_row)
    if node not in demands:
        b_eq.append(0)
    else:
        b_eq.append(demands[node])
```

```
A_eq
```

```
[[1, 0, 0, 1, 0], [-1, 1, 0, 0, 1], [0, -1, 1, -1, 0]]
```

```
res = linprog(c=c, A_ub=A_ub, b_ub=b_ub, A_eq = A_eq,
b_eq=b_eq)
```

```
res.fun
```

```
55999.9999531258
```

```
res.x
```

```
array([3866.70844779, 5999.99999497, 2133.29154718,
1133.29154802,
        866.70845031])
```

b)

```
from scipy.optimize import linprog
```

```
links = [(1,2),(2,3),(3,4),(1,3),(2,4)]
capacities = {(1,2):6000, (2,3):6000, (3,4):6000, (1,3):3000,
(2,4):3000}
travel_times = {(1,2):3, (2,3):3, (3,4):3, (1,3):10, (2,4):10}
```

```
demands = {1:5000, 2:3000, 3:-5000, 4:-3000}
```

```
c = [travel_times[link] for link in links]
```

```
c
```

```
[3, 3, 3, 10, 10]
```

```
A_ub = []
b_ub = []
```

```
# capacity constrain
for link in links:
    A_row = [int(link == l) for l in links]
    A_ub.append(A_row)
    b_ub.append(capacities[link])
```

```
A_eq = []
b_eq = []
```

```
# flow conservation
for node in [1,2,3]:
    A_row = [int(link[0] == node) - int(link[1] == node) for link in links]
    A_eq.append(A_row)
    if node not in demands:
        b_eq.append(0)
    else:
        b_eq.append(demands[node])
# additional
A_row = [ sum(int(link == (1,j))-int(link == (1,3)) for j in [2,3,4]) for link in links]
A_eq.append(A_row)
b_eq.append(0)
A_row = [ sum(int(link == (2,j))-int(link == (2,4)) for j in [1,3,4]) for link in links]
A_eq.append(A_row)
b_eq.append(0)
```

```
A_eq
```

```
[[1, 0, 0, 1, 0],
 [-1, 1, 0, 0, 1],
 [0, -1, 1, -1, 0],
 [1, 0, 0, -2, 0],
 [0, 1, 0, 0, -2]]
```

```
res = linprog(c=c, A_ub=A_ub, b_ub=b_ub, A_eq = A_eq, b_eq=b_eq)
```

```
res.fun
```

```
63111.11100475225
```

```
res.x
```

```
array([3333.33332771, 4222.22221511,  888.88888739,
1666.66666386,
       2111.11110755])
```