In [1]:

```
import numpy as np
```

In [2]:

```
class simplex:
    def __init__(self, A):
        self.A = A
        self.B = {}
        self.N = {}
        self.xb = np.zeros((A.shape[0] - 1))
        self.steps = 1

        self.setup()

    def setup(self):
        self.B['index'] = []
        self.N['index'] = []

        for i in range(0, self.A.shape[1] - 1):
            if self.A[0, i] != 0: self.N['index'].append(i+1)
            else: self.B['index'].append(i+1)

        self.B['cb'] = np.zeros((len(self.B['index'])))
        self.N['cn'] = np.zeros((len(self.N['index'])))

        for i in range(0, len(self.N['cn'])):
            self.N['cn'][i] = self.A[0, self.N['index'][i] - 1]

        self.B['B'] = np.zeros((self.A.shape[0] - 1, self.A.shape[0] - 1))
        self.N['N'] = np.zeros((self.A.shape[0] - 1, self.A.shape[0] - 1))

        for i in range(1, self.A.shape[0]):
            for j in range(0, self.A.shape[0] - 1):
                self.B['B'][i - 1][j] = self.A[i, self.B['index'][j] - 1]
                self.N['N'][i - 1][j] = self.A[i, self.N['index'][j] - 1]

        self.xb = np.linalg.inv(self.B['B']).dot(A[1:, -1])

    def set_nb(self):
        for i in range(1, self.A.shape[0]):
            for j in range(0, self.A.shape[0] - 1):
                self.B['B'][i - 1][j] = self.A[i, self.B['index'][j] - 1]
                self.N['N'][i - 1][j] = self.A[i, self.N['index'][j] - 1]

    def set_c(self):
        for i in range(0, len(self.N['cn'])):
            self.N['cn'][i] = self.A[0, self.N['index'][i] - 1]
        for i in range(0, len(self.B['cb'])):
            self.B['cb'][i] = self.A[0, self.B['index'][i] - 1]

    def step(self, full=False):
        self.set_nb()
        self.set_c()
        lambda_ = np.linalg.inv(self.B['B']).transpose().dot(self.B['cb'])
        sn = self.N['cn'] - self.N['N'].transpose().dot(lambda_)

        comp = 0
        q = 0
        for i in range(0, sn.shape[0]):
            if sn[i] < comp:
                comp = sn[i]
                q = i + 1

        if comp < 0:
            d = np.linalg.inv(self.B['B']).dot(A[1:, q - 1])

            if (d > 0).all():
                z = np.divide(self.xb, d)

                p = np.argmin(z) + 1
                xq = z[p - 1]

                self.xb = self.xb - d*xq
                self.xb[p - 1] = xq

                out = self.B['index'][p - 1]
                self.B['index'][p - 1] = q
                self.N['index'][q - 1] = out

                if full:
                    self.steps += 1
                    self.step(full)
                else:
                    self.print(lambda_, sn)
                    self.steps += 1

            else:
                print('Unbounded: \n')
                self.print(lambda_, sn)
        else:
            print('Optimal found: \n')
            self.print(lambda_, sn)

    def print(self, lambda_, sn):
        print('B-index: ', self.B['index'])
        print('N-index: ', self.N['index'])
        print('xb: ', self.xb)
        print('lambda: ', lambda_)
        print('sn: ', sn)
        print('Iterations: ', self.steps)
```

In [3]:

```
#Example 13.1, p. 371
A = np.array([[-4, -2, 0, 0, 0],
              [1, 1, 1, 0, 5],
              [2, 0.5, 0, 1, 8]])
```

In [4]:

```
s = simplex(A).step(True)
```

```
Optimal found: 

B-index:  [2, 1]
N-index:  [4, 3]
xb:  [1.33333333 3.66666667]
lambda:  [-1.33333333 -1.33333333]
sn:  [1.33333333 1.33333333]
Iterations:  3
```

In [ ]: