In [123]:

```python
"""
ISYE6501: US Army optimization problem with PuLP Modeller

Author: Guillermo de la Hera Casado


Optimization Model Specs:

Variables: amount of servings for each food type.

Constraints: there will be a min and max intake value for each property (cals, fat, sodium...,
across all the foods selected in the diet.

Objective Function: minimize the total cost of the diet, defined as the serving cost of each food
involved in the diet * cost per
serving on that particular food.

Code and solution as per below:
"""

# Import pulp and pandas
from pulp import *
import pandas as pd
import csv
import pprint

foodData = pd.read_excel("diet.xls")
foodData = foodData[0:64]

foodData = foodData.values.tolist()
foods = [x[0] for x in foodData]

#create a dictionary for the cost of the foods
cost = dict([(x[0], float(x[1])) for x in foodData])
cals = dict([(x[0], float(x[3])) for x in foodData])
cho = dict([(x[0], float(x[4])) for x in foodData])
fat = dict([(x[0], float(x[5])) for x in foodData])
sodium = dict([(x[0], float(x[6])) for x in foodData])
carbo = dict([(x[0], float(x[7])) for x in foodData])
fiber = dict([(x[0], float(x[8])) for x in foodData])
protein = dict([(x[0], float(x[9])) for x in foodData])
vitA = dict([(x[0], float(x[10])) for x in foodData])
vitC = dict([(x[0], float(x[11])) for x in foodData])
calcium = dict([(x[0], float(x[12])) for x in foodData])
iron = dict([(x[0], float(x[13])) for x in foodData])

#we define the optimization problem by giving it a name and letting it know we are minimizing
problem = LpProblem("Diet Problem", LpMinimize)

#we define our primary variables (the amount of each food in the diet)
amountVars = LpVariable.dicts("Amounts", foods, 0)

#we add the objective function to the problem, inner product of cost and amountVars
problem += lpSum([cost[i] * amountVars[i] for i in foods]), "total cost"

#next, we hard core all the constraints
problem += lpSum([cals[i] * amountVars[i] for i in foods]) >=1500, 'min cals'
problem += lpSum([cals[i] * amountVars[i] for i in foods]) <=2500, 'max cals'

problem += lpSum([cho[i] * amountVars[i] for i in foods]) >=30, 'min cho'
problem += lpSum([cho[i] * amountVars[i] for i in foods]) <=240, 'max cho'

problem += lpSum([fat[i] * amountVars[i] for i in foods]) >=20, 'min fat'
problem += lpSum([fat[i] * amountVars[i] for i in foods]) <=70, 'max fat'

problem += lpSum([sodium[i] * amountVars[i] for i in foods]) >=800, 'min sodium'
problem += lpSum([sodium[i] * amountVars[i] for i in foods]) <=2000, 'max sodium'

problem += lpSum([carbo[i] * amountVars[i] for i in foods]) >=130, 'min carbo'
problem += lpSum([carbo[i] * amountVars[i] for i in foods]) <=450, 'max carbo'

problem += lpSum([fiber[i] * amountVars[i] for i in foods]) >=125, 'min fiber'
problem += lpSum([fiber[i] * amountVars[i] for i in foods]) <=250, 'max fiber'

problem += lpSum([protein[i] * amountVars[i] for i in foods]) >=60, 'min protein'
problem += lpSum([protein[i] * amountVars[i] for i in foods]) <=100, 'max protein'

problem += lpSum([vitA[i] * amountVars[i] for i in foods]) >=1000, 'min vitA'
problem += lpSum([vitA[i] * amountVars[i] for i in foods]) <=10000, 'max vitA'

problem += lpSum([vitC[i] * amountVars[i] for i in foods]) >=400, 'min vitC'
problem += lpSum([vitC[i] * amountVars[i] for i in foods]) <=5000, 'max vitC'

problem += lpSum([calcium[i] * amountVars[i] for i in foods]) >=700, 'min calcium'
problem += lpSum([calcium[i] * amountVars[i] for i in foods]) <=1500, 'max calcium'

problem += lpSum([iron[i] * amountVars[i] for i in foods]) >=10, 'min iron'
problem += lpSum([iron[i] * amountVars[i] for i in foods]) <=40, 'max iron'


#we solve the problem
problem.solve()

#print out amount values from optimal solution
varsDictionary = {}
for v in problem.variables():
    if v.varValue>0:
        varsDictionary[v.name] = v.varValue

#print(varsDictionary)
pprint.pprint(varsDictionary)
```

```
{'Amounts_Celery,_Raw': 52.64371,
 'Amounts_Frozen_Broccoli': 0.25960653,
 'Amounts_Lettuce,Iceberg,Raw': 63.988506,
 'Amounts_Oranges': 2.2929389,
 'Amounts_Poached_Eggs': 0.14184397,
 'Amounts_Popcorn,Air_Popped': 13.869322}
```

In [70]:

```python
"""
2a) adding the constraint for minimum 1/10 serving when a food is selected
"""



#we define the optimization problem by giving it a name and letting it know we are minimizing
problem = LpProblem("Diet Problem", LpMinimize)

#we define our primary variables (the amount of each food in the diet)
amountVars = LpVariable.dicts("Amounts", foods, 0)
binaryVars = LpVariable.dicts("InorOut", foods, 0, 1, LpBinary)

#we add the objective function to the problem, inner product of cost and amountVars
problem += lpSum([cost[i] * amountVars[i] for i in foods]), "total cost"

#next, we hard core all the constraints
problem += lpSum([cals[i] * amountVars[i] for i in foods]) >=1500, 'min cals'
problem += lpSum([cals[i] * amountVars[i] for i in foods]) <=2500, 'max cals'

problem += lpSum([cho[i] * amountVars[i] for i in foods]) >=30, 'min cho'
problem += lpSum([cho[i] * amountVars[i] for i in foods]) <=240, 'max cho'

problem += lpSum([fat[i] * amountVars[i] for i in foods]) >=20, 'min fat'
problem += lpSum([fat[i] * amountVars[i] for i in foods]) <=70, 'max fat'

problem += lpSum([sodium[i] * amountVars[i] for i in foods]) >=800, 'min sodium'
problem += lpSum([sodium[i] * amountVars[i] for i in foods]) <=2000, 'max sodium'

problem += lpSum([carbo[i] * amountVars[i] for i in foods]) >=130, 'min carbo'
problem += lpSum([carbo[i] * amountVars[i] for i in foods]) <=450, 'max carbo'

problem += lpSum([fiber[i] * amountVars[i] for i in foods]) >=125, 'min fiber'
problem += lpSum([fiber[i] * amountVars[i] for i in foods]) <=250, 'max fiber'

problem += lpSum([protein[i] * amountVars[i] for i in foods]) >=60, 'min protein'
problem += lpSum([protein[i] * amountVars[i] for i in foods]) <=100, 'max protein'
problem += lpSum([protein[i] * amountVars[i] for i in foods]) <=100, 'max protein'

problem += lpSum([vitA[i] * amountVars[i] for i in foods]) >=1000, 'min vitA'
problem += lpSum([vitA[i] * amountVars[i] for i in foods]) <=10000, 'max vitA'

problem += lpSum([vitC[i] * amountVars[i] for i in foods]) >=400, 'min vitC'
problem += lpSum([vitC[i] * amountVars[i] for i in foods]) <=5000, 'max vitC'

problem += lpSum([calcium[i] * amountVars[i] for i in foods]) >=700, 'min calcium'
problem += lpSum([calcium[i] * amountVars[i] for i in foods]) <=1500, 'max calcium'

problem += lpSum([iron[i] * amountVars[i] for i in foods]) >=10, 'min iron'
problem += lpSum([iron[i] * amountVars[i] for i in foods]) <=40, 'max iron'

for i in foods:
    problem += amountVars[i]>= binaryVars[i]*0.1
    problem += amountVars[i]<= binaryVars[i]*1e5
    problem += amountVars[i] >= 1/10*binaryVars[i]

#we solve the problem
problem.solve()

#print out amount values from optimal solution
varsDictionary = {}
for v in problem.variables():
     if v.varValue>0:
        varsDictionary[v.name] = v.varValue

#print(varsDictionary)
pprint.pprint(varsDictionary)
```

```
{'Amounts_Celery,_Raw': 52.64371,
 'Amounts_Frozen_Broccoli': 0.25960653,
 'Amounts_Lettuce,Iceberg,Raw': 63.988506,
 'Amounts_Oranges': 2.2929389,
 'Amounts_Poached_Eggs': 0.14184397,
 'Amounts_Popcorn,Air_Popped': 13.869322,
 'InorOut_Celery,_Raw': 1.0,
 'InorOut_Frozen_Broccoli': 1.0,
 'InorOut_Lettuce,Iceberg,Raw': 1.0,
 'InorOut_Oranges': 1.0,
 'InorOut_Poached_Eggs': 1.0,
 'InorOut_Popcorn,Air_Popped': 1.0}
```

In [79]:

```
"""
2b) Many people dislike brocoli and celery. At most one, but not both can be selected
"""

#we define the optimization problem by giving it a name and letting it know we are minimizing
problem = LpProblem("Diet Problem", LpMinimize)

#we define our primary variables (the amount of each food in the diet)
amountVars = LpVariable.dicts("Amounts", foods, 0)
binaryVars = LpVariable.dicts("InorOut", foods, 0, 1, LpBinary)

#we add the objective function to the problem, inner product of cost and amountVars
problem += lpSum([cost[i] * amountVars[i] for i in foods]), "total cost"

#next, we hard core all the constraints
problem += lpSum([cals[i] * amountVars[i] for i in foods]) >=1500, 'min cals'
problem += lpSum([cals[i] * amountVars[i] for i in foods]) <=2500, 'max cals'

problem += lpSum([cho[i] * amountVars[i] for i in foods]) >=30, 'min cho'
problem += lpSum([cho[i] * amountVars[i] for i in foods]) <=240, 'max cho'

problem += lpSum([fat[i] * amountVars[i] for i in foods]) >=20, 'min fat'
problem += lpSum([fat[i] * amountVars[i] for i in foods]) <=70, 'max fat'

problem += lpSum([sodium[i] * amountVars[i] for i in foods]) >=800, 'min sodium'
problem += lpSum([sodium[i] * amountVars[i] for i in foods]) <=2000, 'max sodium'

problem += lpSum([carbo[i] * amountVars[i] for i in foods]) >=130, 'min carbo'
problem += lpSum([carbo[i] * amountVars[i] for i in foods]) <=450, 'max carbo'
problem += lpSum([fiber[i] * amountVars[i] for i in foods]) >=125, 'min fiber'
problem += lpSum([fiber[i] * amountVars[i] for i in foods]) <=250, 'max fiber'

problem += lpSum([protein[i] * amountVars[i] for i in foods]) >=60, 'min protein'
problem += lpSum([protein[i] * amountVars[i] for i in foods]) <=100, 'max protein'

problem += lpSum([vitA[i] * amountVars[i] for i in foods]) >=1000, 'min vitA'
problem += lpSum([vitA[i] * amountVars[i] for i in foods]) <=10000, 'max vitA'

problem += lpSum([vitC[i] * amountVars[i] for i in foods]) >=400, 'min vitC'
problem += lpSum([vitC[i] * amountVars[i] for i in foods]) <=5000, 'max vitC'

problem += lpSum([calcium[i] * amountVars[i] for i in foods]) >=700, 'min calcium'
problem += lpSum([calcium[i] * amountVars[i] for i in foods]) <=1500, 'max calcium'

problem += lpSum([iron[i] * amountVars[i] for i in foods]) >=10, 'min iron'
problem += lpSum([iron[i] * amountVars[i] for i in foods]) <=40, 'max iron'

for i in foods:
    problem += amountVars[i]>= binaryVars[i]*0.1
    problem += amountVars[i]<= binaryVars[i]*1e5

problem += binaryVars['Frozen Broccoli'] + binaryVars['Celery, Raw'] <=1

#we solve the problem
problem.solve()

#print out amount values from optimal solution
varsDictionary = {}
for v in problem.variables():
     if v.varValue>0:
        varsDictionary[v.name] = v.varValue

#print(varsDictionary)
pprint.pprint(varsDictionary)
```

```
{'Amounts_Celery,_Raw': 43.154119,
 'Amounts_Lettuce,Iceberg,Raw': 80.919121,
 'Amounts_Oranges': 3.0765161,
 'Amounts_Peanut_Butter': 2.0464575,
 'Amounts_Poached_Eggs': 0.14184397,
 'Amounts_Popcorn,Air_Popped': 13.181772,
 'InorOut_Celery,_Raw': 1.0,
 'InorOut_Lettuce,Iceberg,Raw': 1.0,
 'InorOut_Oranges': 1.0,
 'InorOut_Peanut_Butter': 1.0,
 'InorOut_Poached_Eggs': 1.0,
 'InorOut_Popcorn,Air_Popped': 1.0}
```

In [90]:

```
"""
2c) at least 3 kinds of meat/poultry/fish/eggs must be selected
"""

#we define the optimization problem by giving it a name and letting it know we are minimizing
problem = LpProblem("Diet Problem", LpMinimize)

#we define our primary variables (the amount of each food in the diet)
amountVars = LpVariable.dicts("Amounts", foods, 0)
selectionVars = LpVariable.dicts("InorOut", foods, 0, 1, LpBinary)

isProtein = dict([(x[0], 0) for x in foodData])

#we add the objective function to the problem, inner product of cost and amountVars
problem += lpSum([cost[i] * amountVars[i] for i in foods]), "total cost"

#next, we hard core all the constraints
problem += lpSum([cals[i] * amountVars[i] for i in foods]) >=1500, 'min cals'
problem += lpSum([cals[i] * amountVars[i] for i in foods]) <=2500, 'max cals'

problem += lpSum([cho[i] * amountVars[i] for i in foods]) >=30, 'min cho'
problem += lpSum([cho[i] * amountVars[i] for i in foods]) <=240, 'max cho'

problem += lpSum([fat[i] * amountVars[i] for i in foods]) >=20, 'min fat'
problem += lpSum([fat[i] * amountVars[i] for i in foods]) <=70, 'max fat'

problem += lpSum([sodium[i] * amountVars[i] for i in foods]) >=800, 'min sodium'
problem += lpSum([sodium[i] * amountVars[i] for i in foods]) <=2000, 'max sodium'

problem += lpSum([carbo[i] * amountVars[i] for i in foods]) >=130, 'min carbo'
problem += lpSum([carbo[i] * amountVars[i] for i in foods]) <=450, 'max carbo'

problem += lpSum([fiber[i] * amountVars[i] for i in foods]) >=125, 'min fiber'
problem += lpSum([fiber[i] * amountVars[i] for i in foods]) <=250, 'max fiber'

problem += lpSum([protein[i] * amountVars[i] for i in foods]) >=60, 'min protein'
problem += lpSum([protein[i] * amountVars[i] for i in foods]) <=100, 'max protein'

problem += lpSum([vitA[i] * amountVars[i] for i in foods]) >=1000, 'min vitA'
problem += lpSum([vitA[i] * amountVars[i] for i in foods]) <=10000, 'max vitA'

problem += lpSum([vitC[i] * amountVars[i] for i in foods]) >=400, 'min vitC'
problem += lpSum([vitC[i] * amountVars[i] for i in foods]) <=5000, 'max vitC'

problem += lpSum([calcium[i] * amountVars[i] for i in foods]) >=700, 'min calcium'
problem += lpSum([calcium[i] * amountVars[i] for i in foods]) <=1500, 'max calcium'

problem += lpSum([iron[i] * amountVars[i] for i in foods]) >=10, 'min iron'
problem += lpSum([iron[i] * amountVars[i] for i in foods]) <=40, 'max iron'

for i in foods:
    problem += amountVars[i]>= selectionVars[i]*0.1
    problem += amountVars[i]<= selectionVars[i]*1e5

isProtein['Roasted Chicken'] = 1
isProtein['Roasted Chicken'] = 1
isProtein['Poached Eggs'] = 1
isProtein['Scrambled Eggs'] = 1
isProtein['Bologna,Turkey'] = 1
isProtein['Frankfurter, Beef'] = 1
isProtein['Ham,Sliced,Extralean'] = 1
isProtein['Kielbasa,Prk'] = 1
isProtein['Pork'] = 1
isProtein['Sardines in Oil'] = 1
isProtein['White Tuna in Water'] = 1

problem += lpSum([isProtein[i] * selectionVars[i] for i in foods]) >=3, 'min proteins'

#we solve the problem
problem.solve()

#print out amount values from optimal solution
varsDictionary = {}
for v in problem.variables():
     if v.varValue>0:
        varsDictionary[v.name] = v.varValue

#print(varsDictionary)
pprint.pprint(varsDictionary)
```

```
{'Amounts_Bologna,Turkey': 0.1,
 'Amounts_Celery,_Raw': 51.522935,
 'Amounts_Frozen_Broccoli': 0.22815709,
 'Amounts_Lettuce,Iceberg,Raw': 66.955456,
 'Amounts_Oranges': 2.3760495,
 'Amounts_Poached_Eggs': 0.1,
 'Amounts_Popcorn,Air_Popped': 13.847028,
 'Amounts_Scrambled_Eggs': 0.1,
 'InorOut_Bologna,Turkey': 1.0,
 'InorOut_Celery,_Raw': 1.0,
 'InorOut_Frozen_Broccoli': 1.0,
 'InorOut_Lettuce,Iceberg,Raw': 1.0,
 'InorOut_Oranges': 1.0,
 'InorOut_Poached_Eggs': 1.0,
 'InorOut_Popcorn,Air_Popped': 1.0,
 'InorOut_Scrambled_Eggs': 1.0}
```

In [152]:

```
"""
```

```
"""
Long diet problem
"""
import math

foodData = pd.read_excel("diet_large.xls")
foodData = foodData[1:7147]

foodData = foodData.values.tolist()
foods = [x[0] for x in foodData]
protein = dict([(x[0], float(0.0) if math.isnan(x[1]) else float(x[1])) for x in foodData])
calcium = dict([(x[0], float(0.0) if math.isnan(x[6]) else float(x[6])) for x in foodData])
selectionVars = LpVariable.dicts("InorOut", foods, 0, 1, LpBinary)

#create a dictionary for the cholesterol cost of the foods
cho_cost = dict([(x[0], float(0.0) if math.isnan(x[28]) else float(x[28])) for x in foodData])

#we define the optimization problem by giving it a name and letting it know we are minimizing
problem = LpProblem("Diet Large Problem", LpMinimize)

#we define our primary variables (the amount of each food in the diet)
amountVars = LpVariable.dicts("Amounts", foods, 0)


#we add the objective function to the problem, inner product of cholesterol cost and amountVars
problem += lpSum([cho_cost[i] * amountVars[i] for i in foods]), "total cholesterol cost"
#we create some constraints in proteins, so that there is a minimum amount of proteins required in
the diet - otherwise
#the solver would not provide any food - minimizing cholesterol to 0
problem += lpSum([protein[i] * amountVars[i] for i in foods]) >=60, 'min protein'
problem += lpSum([protein[i] * amountVars[i] for i in foods]) <=100, 'max protein'
problem += lpSum([calcium[i] * amountVars[i] for i in foods]) >=100, 'min calcium'
problem += lpSum([calcium[i] * amountVars[i] for i in foods]) <=500, 'max calcium'
for i in foods:
    problem += amountVars[i]>= selectionVars[i]*0.1
    problem += amountVars[i]<= selectionVars[i]*100
#we solve the problem
try:
    problem.solve()
     #print out amount values from optimal solution
    varsDictionary = {}
    for v in problem.variables():
        if v.varValue>0:
            varsDictionary[v.name] = v.varValue
            pprint.pprint(varsDictionary)
except Exception:
     print('Problem infeasible')
```

```
{'Amounts_Fish,_herring,_Pacific,_meat_packed_in_oil,_air_dried_(Alaska_N': 1.2949118}
{'Amounts_Fish,_herring,_Pacific,_meat_packed_in_oil,_air_dried_(Alaska_N': 1.2949118,
 'Amounts_Infant_formula,_NESTLE,_GOOD_START_ESSENTIALS__SOY,__with_iron,': 0.19011407}
{'Amounts_Fish,_herring,_Pacific,_meat_packed_in_oil,_air_dried_(Alaska_N': 1.2949118,
 'Amounts_Infant_formula,_NESTLE,_GOOD_START_ESSENTIALS__SOY,__with_iron,': 0.19011407,
 'InorOut_Fish,_herring,_Pacific,_meat_packed_in_oil,_air_dried_(Alaska_N': 1.0}
{'Amounts_Fish,_herring,_Pacific,_meat_packed_in_oil,_air_dried_(Alaska_N': 1.2949118,
 'Amounts_Infant_formula,_NESTLE,_GOOD_START_ESSENTIALS__SOY,__with_iron,': 0.19011407,
 'InorOut_Fish,_herring,_Pacific,_meat_packed_in_oil,_air_dried_(Alaska_N': 1.0,
 'InorOut_Infant_formula,_NESTLE,_GOOD_START_ESSENTIALS__SOY,__with_iron,': 1.0}
```

```
In [ ]:
```