Homework11

April 2, 2025

1 Question 15.2

In the videos, we saw the "diet problem". (The diet problem is one of the first large-scale optimization problems to be studied in practice. Back in the 1930's and 40's, the Army wanted to meet the nutritional requirements of its soldiers while minimizing the cost.) In this homework you get to solve a diet problem with real data. The data is given in the file diet.xls.

1. Formulate an optimization model (a linear program) to find the cheapest diet that satisfies the maximum and minimum daily nutrition constraints, and solve it using PuLP. Turn in your code and the solution. (The optimal solution should be a diet of air-popped popcorn, poached eggs, oranges, raw iceberg lettuce, raw celery, and frozen broccoli. UGH!)

2. Please add to your model the following constraints (which might require adding more variables) and solve the new model: - If a food is selected, then a minimum of 1/10 serving must be chosen. (Hint: now you will need two variables for each food i: whether it is chosen, and how much is part of the diet. You'll also need to write a constraint to link them.) - Many people dislike celery and frozen broccoli. So at most one, but not both, can be selected. - To get day-to-day variety in protein, at least 3 kinds of meat/poultry/fish/eggs must be selected. [If something is ambiguous (e.g., should bean-and-bacon soup be considered meat?), just call it whatever you think is appropriate – I want you to learn how to write this type of constraint, but I don't really care whether we agree on how to classify foods!]

If you want to see what a more full-sized problem would look like, try solving your models for the file diet_large.xls, which is a low-cholesterol diet model (rather than minimizing cost, the goal is to minimize cholesterol intake). I don't know anyone who'd want to eat this diet – the optimal solution includes dried chrysanthemum garland, raw beluga whale flipper, freeze-dried parsley, etc. – which shows why it's necessary to add additional constraints beyond the basic ones we saw in the video! [Note: there are many optimal solutions, all with zero cholesterol, so you might get a different one. It probably won't be much more appetizing than mine.]

```
from pulp import * import pandas as pd diet = pd.read_excel('/Users/arianaesmailian/Documents/isye6501/hw11
15.2/diet.xls',header=0) display(diet)
```

```
diet_df = diet[0:64] display(diet_df)
```

[87]:

```
exp_nutrient = {'min':[v1 for v1 in diet.iloc[65,3:]], 'max': [v2 for v2 in␣
 ↪diet.iloc[66,3:]]}
print(exp_nutrient)
```

```
{'min': [1500.0, 30.0, 20.0, 800.0, 130.0, 125.0, 60.0, 1000.0, 400.0, 700.0,
10.0], 'max': [2500.0, 240.0, 70.0, 2000.0, 450.0, 250.0, 100.0, 10000.0,
5000.0, 1500.0, 40.0]}
```

[89]:

```
print(cost)
```

```
{'Frozen Broccoli': 0.16, 'Carrots,Raw': 0.07, 'Celery, Raw': 0.04, 'Frozen
Corn': 0.18, 'Lettuce,Iceberg,Raw': 0.02, 'Peppers, Sweet, Raw': 0.53,
'Potatoes, Baked': 0.06, 'Tofu': 0.31, 'Roasted Chicken': 0.84, 'Spaghetti W/
Sauce': 0.78, 'Tomato,Red,Ripe,Raw': 0.27, 'Apple,Raw,W/Skin': 0.24, 'Banana':
0.15, 'Grapes': 0.32, 'Kiwifruit,Raw,Fresh': 0.49, 'Oranges': 0.15, 'Bagels':
0.16, 'Wheat Bread': 0.05, 'White Bread': 0.06, 'Oatmeal Cookies': 0.09, 'Apple
Pie': 0.16, 'Chocolate Chip Cookies': 0.03, 'Butter,Regular': 0.05, 'Cheddar
Cheese': 0.25, '3.3% Fat,Whole Milk': 0.16, '2% Lowfat Milk': 0.23, 'Skim Milk':
0.13, 'Poached Eggs': 0.08, 'Scrambled Eggs': 0.11, 'Bologna,Turkey': 0.15,
'Frankfurter, Beef': 0.27, 'Ham,Sliced,Extralean': 0.33, 'Kielbasa,Prk': 0.15,
"Cap'N Crunch": 0.31, 'Cheerios': 0.28, "Corn Flks, Kellogg'S": 0.28, "Raisin
Brn, Kellg'S": 0.34, 'Rice Krispies': 0.32, 'Special K': 0.38, 'Oatmeal': 0.82,
'Malt-O-Meal,Choc': 0.52, 'Pizza W/Pepperoni': 0.44, 'Taco': 0.59, 'Hamburger
W/Toppings': 0.83, 'Hotdog, Plain': 0.31, 'Couscous': 0.39, 'White Rice': 0.08,
'Macaroni,Ckd': 0.17, 'Peanut Butter': 0.07, 'Pork': 0.81, 'Sardines in Oil':
0.45, 'White Tuna in Water': 0.69, 'Popcorn,Air-Popped': 0.04, 'Potato
Chips,Bbqflvr': 0.22, 'Pretzels': 0.12, 'Tortilla Chip': 0.19, 'Chicknoodl
Soup': 0.39, 'Splt Pea&Hamsoup': 0.67, 'Vegetbeef Soup': 0.71, 'Neweng
Clamchwd': 0.75, 'Tomato Soup': 0.39, 'New E Clamchwd,W/Mlk': 0.99, 'Crm Mshrm
Soup,W/Mlk': 0.65, 'Beanbacn Soup,W/Watr': 0.67}
```

[91]:

```
food = diet_df['Foods'].values.tolist()
print(food[0:5])
```

```
['Frozen Broccoli', 'Carrots,Raw', 'Celery, Raw', 'Frozen Corn',
'Lettuce,Iceberg,Raw']
```

[93]:

```
diet_df1 = diet.copy()
diet_df1.drop(['Price/ Serving', 'Serving Size'], axis = 1, inplace = True)
display(diet_df1)
```

```
                   Foods  Calories  Cholesterol mg  Total_Fat g  Sodium mg  \
0        Frozen Broccoli      73.8             0.0          0.8       68.2
1            Carrots,Raw      23.7             0.0          0.1       19.2
2            Celery, Raw       6.4             0.0          0.1       34.8
3            Frozen Corn      72.2             0.0          0.6        2.5
4    Lettuce,Iceberg,Raw       2.6             0.0          0.0        1.8
..                     …         …               …            …          …
62   Crm Mshrm Soup,W/Mlk    203.4            19.8         13.6     1076.3
```

```
63  Beanbacn Soup,W/Watr     172.0             2.5          5.9      951.3
64                     NaN       NaN             NaN          NaN        NaN
65                     NaN    1500.0            30.0         20.0      800.0
66                     NaN    2500.0           240.0         70.0     2000.0

    Carbohydrates g  Dietary_Fiber g  Protein g  Vit_A IU  Vit_C IU  \
0              13.6              8.5        8.0    5867.4     160.2
1               5.6              1.6        0.6   15471.0       5.1
2               1.5              0.7        0.3      53.6       2.8
3              17.1              2.0        2.5     106.6       5.2
4               0.4              0.3        0.2      66.0       0.8
..              …             …        …       …       …
62             15.0              0.5        6.1     153.8       2.2
63             22.8              8.6        7.9     888.0       1.5
64              NaN              NaN        NaN       NaN       NaN
65            130.0            125.0       60.0    1000.0     400.0
66            450.0            250.0      100.0   10000.0    5000.0

    Calcium mg  Iron mg
0        159.0      2.3
1         14.9      0.3
2         16.0      0.2
3          3.3      0.3
4          3.8      0.1
..         …      …
62       178.6      0.6
63        81.0      2.0
64         NaN      NaN
65       700.0     10.0
66      1500.0     40.0

[67 rows x 12 columns]
```

[95]:
```python
food_nutrient = diet_df1.set_index('Foods').to_dict('dict')
print(food_nutrient)
```

```
{'Calories': {'Frozen Broccoli': 73.8, 'Carrots,Raw': 23.7, 'Celery, Raw': 6.4,
'Frozen Corn': 72.2, 'Lettuce,Iceberg,Raw': 2.6, 'Peppers, Sweet, Raw': 20.0,
'Potatoes, Baked': 171.5, 'Tofu': 88.2, 'Roasted Chicken': 277.4, 'Spaghetti W/
Sauce': 358.2, 'Tomato,Red,Ripe,Raw': 25.8, 'Apple,Raw,W/Skin': 81.4, 'Banana':
104.9, 'Grapes': 15.1, 'Kiwifruit,Raw,Fresh': 46.4, 'Oranges': 61.6, 'Bagels':
78.0, 'Wheat Bread': 65.0, 'White Bread': 65.0, 'Oatmeal Cookies': 81.0, 'Apple
Pie': 67.2, 'Chocolate Chip Cookies': 78.1, 'Butter,Regular': 35.8, 'Cheddar
Cheese': 112.7, '3.3% Fat,Whole Milk': 149.9, '2% Lowfat Milk': 121.2, 'Skim
Milk': 85.5, 'Poached Eggs': 74.5, 'Scrambled Eggs': 99.6, 'Bologna,Turkey':
56.4, 'Frankfurter, Beef': 141.8, 'Ham,Sliced,Extralean': 37.1, 'Kielbasa,Prk':
80.6, "Cap'N Crunch": 119.6, 'Cheerios': 111.0, "Corn Flks, Kellogg'S": 110.5,
"Raisin Brn, Kellg'S": 115.1, 'Rice Krispies': 112.2, 'Special K': 110.8,
'Oatmeal': 145.1, 'Malt-O-Meal,Choc': 607.2, 'Pizza W/Pepperoni': 181.0, 'Taco':
369.4, 'Hamburger W/Toppings': 275.0, 'Hotdog, Plain': 242.1, 'Couscous': 100.8,
'White Rice': 102.7, 'Macaroni,Ckd': 98.7, 'Peanut Butter': 188.5, 'Pork':
710.8, 'Sardines in Oil': 49.9, 'White Tuna in Water': 115.6, 'Popcorn,Air-
Popped': 108.3, 'Potato Chips,Bbqflvr': 139.2, 'Pretzels': 108.0, 'Tortilla
Chip': 142.0, 'Chicknoodl Soup': 150.1, 'Splt Pea&Hamsoup': 184.8, 'Vegetbeef
Soup': 158.1, 'Neweng Clamchwd': 175.7, 'Tomato Soup': 170.7, 'New E
Clamchwd,W/Mlk': 163.7, 'Crm Mshrm Soup,W/Mlk': 203.4, 'Beanbacn Soup,W/Watr':
172.0, nan: 2500.0}, 'Cholesterol mg': {'Frozen Broccoli': 0.0, 'Carrots,Raw':
0.0, 'Celery, Raw': 0.0, 'Frozen Corn': 0.0, 'Lettuce,Iceberg,Raw': 0.0,
'Peppers, Sweet, Raw': 0.0, 'Potatoes, Baked': 0.0, 'Tofu': 0.0, 'Roasted
Chicken': 129.9, 'Spaghetti W/ Sauce': 0.0, 'Tomato,Red,Ripe,Raw': 0.0,
'Apple,Raw,W/Skin': 0.0, 'Banana': 0.0, 'Grapes': 0.0, 'Kiwifruit,Raw,Fresh':
0.0, 'Oranges': 0.0, 'Bagels': 0.0, 'Wheat Bread': 0.0, 'White Bread': 0.0,
'Oatmeal Cookies': 0.0, 'Apple Pie': 0.0, 'Chocolate Chip Cookies': 5.1,
'Butter,Regular': 10.9, 'Cheddar Cheese': 29.4, '3.3% Fat,Whole Milk': 33.2, '2%
Lowfat Milk': 18.3, 'Skim Milk': 4.4, 'Poached Eggs': 211.5, 'Scrambled Eggs':
211.2, 'Bologna,Turkey': 28.1, 'Frankfurter, Beef': 27.4,
'Ham,Sliced,Extralean': 13.3, 'Kielbasa,Prk': 17.4, "Cap'N Crunch": 0.0,
'Cheerios': 0.0, "Corn Flks, Kellogg'S": 0.0, "Raisin Brn, Kellg'S": 0.0, 'Rice
Krispies': 0.0, 'Special K': 0.0, 'Oatmeal': 0.0, 'Malt-O-Meal,Choc': 0.0,
'Pizza W/Pepperoni': 14.2, 'Taco': 56.4, 'Hamburger W/Toppings': 42.8, 'Hotdog,
Plain': 44.1, 'Couscous': 0.0, 'White Rice': 0.0, 'Macaroni,Ckd': 0.0, 'Peanut
Butter': 0.0, 'Pork': 105.1, 'Sardines in Oil': 34.1, 'White Tuna in Water':
35.7, 'Popcorn,Air-Popped': 0.0, 'Potato Chips,Bbqflvr': 0.0, 'Pretzels': 0.0,
'Tortilla Chip': 0.0, 'Chicknoodl Soup': 12.3, 'Splt Pea&Hamsoup': 7.2,
'Vegetbeef Soup': 10.0, 'Neweng Clamchwd': 10.0, 'Tomato Soup': 0.0, 'New E
Clamchwd,W/Mlk': 22.3, 'Crm Mshrm Soup,W/Mlk': 19.8, 'Beanbacn Soup,W/Watr':
2.5, nan: 240.0}, 'Total_Fat g': {'Frozen Broccoli': 0.8, 'Carrots,Raw': 0.1,
'Celery, Raw': 0.1, 'Frozen Corn': 0.6, 'Lettuce,Iceberg,Raw': 0.0, 'Peppers,
Sweet, Raw': 0.1, 'Potatoes, Baked': 0.2, 'Tofu': 5.5, 'Roasted Chicken': 10.8,
'Spaghetti W/ Sauce': 12.3, 'Tomato,Red,Ripe,Raw': 0.4, 'Apple,Raw,W/Skin': 0.5,
'Banana': 0.5, 'Grapes': 0.1, 'Kiwifruit,Raw,Fresh': 0.3, 'Oranges': 0.2,
'Bagels': 0.5, 'Wheat Bread': 1.0, 'White Bread': 1.0, 'Oatmeal Cookies': 3.3,
'Apple Pie': 3.1, 'Chocolate Chip Cookies': 4.5, 'Butter,Regular': 4.1, 'Cheddar
Cheese': 9.3, '3.3% Fat,Whole Milk': 8.1, '2% Lowfat Milk': 4.7, 'Skim Milk':
0.4, 'Poached Eggs': 5.0, 'Scrambled Eggs': 7.3, 'Bologna,Turkey': 4.3,
'Frankfurter, Beef': 12.8, 'Ham,Sliced,Extralean': 1.4, 'Kielbasa,Prk': 7.1,
"Cap'N Crunch": 2.6, 'Cheerios': 1.8, "Corn Flks, Kellogg'S": 0.1, "Raisin Brn,
Kellg'S": 0.7, 'Rice Krispies': 0.2, 'Special K': 0.1, 'Oatmeal': 2.3, 'Malt-O-
Meal,Choc': 1.5, 'Pizza W/Pepperoni': 7.0, 'Taco': 20.6, 'Hamburger W/Toppings':
10.2, 'Hotdog, Plain': 14.5, 'Couscous': 0.1, 'White Rice': 0.2, 'Macaroni,Ckd':
0.5, 'Peanut Butter': 16.0, 'Pork': 72.2, 'Sardines in Oil': 2.7, 'White Tuna in
Water': 2.1, 'Popcorn,Air-Popped': 1.2, 'Potato Chips,Bbqflvr': 9.2, 'Pretzels':
1.0, 'Tortilla Chip': 7.4, 'Chicknoodl Soup': 4.6, 'Splt Pea&Hamsoup': 4.0,
'Vegetbeef Soup': 3.8, 'Neweng Clamchwd': 5.0, 'Tomato Soup': 3.8, 'New E
Clamchwd,W/Mlk': 6.6, 'Crm Mshrm Soup,W/Mlk': 13.6, 'Beanbacn Soup,W/Watr': 5.9,
nan: 70.0}, 'Sodium mg': {'Frozen Broccoli': 68.2, 'Carrots,Raw': 19.2, 'Celery,
Raw': 34.8, 'Frozen Corn': 2.5, 'Lettuce,Iceberg,Raw': 1.8, 'Peppers, Sweet,
Raw': 1.5, 'Potatoes, Baked': 15.2, 'Tofu': 8.1, 'Roasted Chicken': 125.6,
'Spaghetti W/ Sauce': 1237.1, 'Tomato,Red,Ripe,Raw': 11.1, 'Apple,Raw,W/Skin':
0.0, 'Banana': 1.1, 'Grapes': 0.5, 'Kiwifruit,Raw,Fresh': 3.8, 'Oranges': 0.0,
'Bagels': 151.4, 'Wheat Bread': 134.5, 'White Bread': 132.5, 'Oatmeal Cookies':
68.9, 'Apple Pie': 75.4, 'Chocolate Chip Cookies': 57.8, 'Butter,Regular': 41.3,
'Cheddar Cheese': 173.7, '3.3% Fat,Whole Milk': 119.6, '2% Lowfat Milk': 121.8,
'Skim Milk': 126.2, 'Poached Eggs': 140.0, 'Scrambled Eggs': 168.0,
'Bologna,Turkey': 248.9, 'Frankfurter, Beef': 461.7, 'Ham,Sliced,Extralean':
405.1, 'Kielbasa,Prk': 279.8, "Cap'N Crunch": 213.3, 'Cheerios': 307.6, "Corn
Flks, Kellogg'S": 290.5, "Raisin Brn, Kellg'S": 204.4, 'Rice Krispies': 340.8,
'Special K': 265.5, 'Oatmeal': 2.3, 'Malt-O-Meal,Choc': 16.5, 'Pizza
W/Pepperoni': 267.0, 'Taco': 802.0, 'Hamburger W/Toppings': 563.9, 'Hotdog,
Plain': 670.3, 'Couscous': 4.5, 'White Rice': 0.8, 'Macaroni,Ckd': 0.7, 'Peanut
Butter': 155.5, 'Pork': 38.4, 'Sardines in Oil': 121.2, 'White Tuna in Water':
333.2, 'Popcorn,Air-Popped': 1.1, 'Potato Chips,Bbqflvr': 212.6, 'Pretzels':
486.2, 'Tortilla Chip': 149.7, 'Chicknoodl Soup': 1862.2, 'Splt Pea&Hamsoup':
964.8, 'Vegetbeef Soup': 1915.1, 'Neweng Clamchwd': 1864.9, 'Tomato Soup':
1744.4, 'New E Clamchwd,W/Mlk': 992.0, 'Crm Mshrm Soup,W/Mlk': 1076.3, 'Beanbacn
Soup,W/Watr': 951.3, nan: 2000.0}, 'Carbohydrates g': {'Frozen Broccoli': 13.6,
'Carrots,Raw': 5.6, 'Celery, Raw': 1.5, 'Frozen Corn': 17.1,
'Lettuce,Iceberg,Raw': 0.4, 'Peppers, Sweet, Raw': 4.8, 'Potatoes, Baked': 39.9,
'Tofu': 2.2, 'Roasted Chicken': 0.0, 'Spaghetti W/ Sauce': 58.3,
'Tomato,Red,Ripe,Raw': 5.7, 'Apple,Raw,W/Skin': 21.0, 'Banana': 26.7, 'Grapes':
4.1, 'Kiwifruit,Raw,Fresh': 11.3, 'Oranges': 15.4, 'Bagels': 15.1, 'Wheat
Bread': 12.4, 'White Bread': 11.8, 'Oatmeal Cookies': 12.4, 'Apple Pie': 9.6,
'Chocolate Chip Cookies': 9.3, 'Butter,Regular': 0.0, 'Cheddar Cheese': 0.4,
'3.3% Fat,Whole Milk': 11.4, '2% Lowfat Milk': 11.7, 'Skim Milk': 11.9, 'Poached
Eggs': 0.6, 'Scrambled Eggs': 1.3, 'Bologna,Turkey': 0.3, 'Frankfurter, Beef':
0.8, 'Ham,Sliced,Extralean': 0.3, 'Kielbasa,Prk': 0.6, "Cap'N Crunch": 23.0,
'Cheerios': 19.6, "Corn Flks, Kellogg'S": 24.5, "Raisin Brn, Kellg'S": 27.9,
'Rice Krispies': 24.8, 'Special K': 21.3, 'Oatmeal': 25.3, 'Malt-O-Meal,Choc':
128.2, 'Pizza W/Pepperoni': 19.9, 'Taco': 26.7, 'Hamburger W/Toppings': 32.7,
'Hotdog, Plain': 18.0, 'Couscous': 20.9, 'White Rice': 22.3, 'Macaroni,Ckd':
19.8, 'Peanut Butter': 6.9, 'Pork': 0.0, 'Sardines in Oil': 0.0, 'White Tuna in
Water': 0.0, 'Popcorn,Air-Popped': 22.1, 'Potato Chips,Bbqflvr': 15.0,
'Pretzels': 22.5, 'Tortilla Chip': 17.8, 'Chicknoodl Soup': 18.7, 'Splt
Pea&Hamsoup': 26.8, 'Vegetbeef Soup': 20.4, 'Neweng Clamchwd': 21.8, 'Tomato
Soup': 33.2, 'New E Clamchwd,W/Mlk': 16.6, 'Crm Mshrm Soup,W/Mlk': 15.0,
'Beanbacn Soup,W/Watr': 22.8, nan: 450.0}, 'Dietary_Fiber g': {'Frozen
Broccoli': 8.5, 'Carrots,Raw': 1.6, 'Celery, Raw': 0.7, 'Frozen Corn': 2.0,
'Lettuce,Iceberg,Raw': 0.3, 'Peppers, Sweet, Raw': 1.3, 'Potatoes, Baked': 3.2,
'Tofu': 1.4, 'Roasted Chicken': 0.0, 'Spaghetti W/ Sauce': 11.6,
'Tomato,Red,Ripe,Raw': 1.4, 'Apple,Raw,W/Skin': 3.7, 'Banana': 2.7, 'Grapes':
0.2, 'Kiwifruit,Raw,Fresh': 2.6, 'Oranges': 3.1, 'Bagels': 0.6, 'Wheat Bread':
1.3, 'White Bread': 1.1, 'Oatmeal Cookies': 0.6, 'Apple Pie': 0.5, 'Chocolate
Chip Cookies': 0.0, 'Butter,Regular': 0.0, 'Cheddar Cheese': 0.0, '3.3%
Fat,Whole Milk': 0.0, '2% Lowfat Milk': 0.0, 'Skim Milk': 0.0, 'Poached Eggs':
```

```
0.0, 'Scrambled Eggs': 0.0, 'Bologna,Turkey': 0.0, 'Frankfurter, Beef': 0.0,
'Ham,Sliced,Extralean': 0.0, 'Kielbasa,Prk': 0.0, "Cap'N Crunch": 0.5,
'Cheerios': 2.0, "Corn Flks, Kellogg'S": 0.7, "Raisin Brn, Kellg'S": 4.0, 'Rice
Krispies': 0.4, 'Special K': 0.7, 'Oatmeal': 4.0, 'Malt-O-Meal,Choc': 0.0,
'Pizza W/Pepperoni': 0.0, 'Taco': 0.0, 'Hamburger W/Toppings': 0.0, 'Hotdog,
Plain': 0.0, 'Couscous': 1.3, 'White Rice': 0.3, 'Macaroni,Ckd': 0.9, 'Peanut
Butter': 2.1, 'Pork': 0.0, 'Sardines in Oil': 0.0, 'White Tuna in Water': 0.0,
'Popcorn,Air-Popped': 4.3, 'Potato Chips,Bbqflvr': 1.2, 'Pretzels': 0.9,
'Tortilla Chip': 1.8, 'Chicknoodl Soup': 1.5, 'Splt Pea&Hamsoup': 4.1,
'Vegetbeef Soup': 4.0, 'Neweng Clamchwd': 1.5, 'Tomato Soup': 1.0, 'New E
Clamchwd,W/Mlk': 1.5, 'Crm Mshrm Soup,W/Mlk': 0.5, 'Beanbacn Soup,W/Watr': 8.6,
nan: 250.0}, 'Protein g': {'Frozen Broccoli': 8.0, 'Carrots,Raw': 0.6, 'Celery,
Raw': 0.3, 'Frozen Corn': 2.5, 'Lettuce,Iceberg,Raw': 0.2, 'Peppers, Sweet,
Raw': 0.7, 'Potatoes, Baked': 3.7, 'Tofu': 9.4, 'Roasted Chicken': 42.2,
'Spaghetti W/ Sauce': 8.2, 'Tomato,Red,Ripe,Raw': 1.0, 'Apple,Raw,W/Skin': 0.3,
'Banana': 1.2, 'Grapes': 0.2, 'Kiwifruit,Raw,Fresh': 0.8, 'Oranges': 1.2,
'Bagels': 3.0, 'Wheat Bread': 2.2, 'White Bread': 2.3, 'Oatmeal Cookies': 1.1,
'Apple Pie': 0.5, 'Chocolate Chip Cookies': 0.9, 'Butter,Regular': 0.0, 'Cheddar
Cheese': 7.0, '3.3% Fat,Whole Milk': 8.0, '2% Lowfat Milk': 8.1, 'Skim Milk':
8.4, 'Poached Eggs': 6.2, 'Scrambled Eggs': 6.7, 'Bologna,Turkey': 3.9,
'Frankfurter, Beef': 5.4, 'Ham,Sliced,Extralean': 5.5, 'Kielbasa,Prk': 3.4,
"Cap'N Crunch": 1.4, 'Cheerios': 4.3, "Corn Flks, Kellogg'S": 2.3, "Raisin Brn,
Kellg'S": 4.0, 'Rice Krispies': 1.9, 'Special K': 5.6, 'Oatmeal': 6.1, 'Malt-O-
Meal,Choc': 17.3, 'Pizza W/Pepperoni': 10.1, 'Taco': 20.7, 'Hamburger
W/Toppings': 13.6, 'Hotdog, Plain': 10.4, 'Couscous': 3.4, 'White Rice': 2.1,
'Macaroni,Ckd': 3.3, 'Peanut Butter': 7.7, 'Pork': 13.8, 'Sardines in Oil': 5.9,
'White Tuna in Water': 22.7, 'Popcorn,Air-Popped': 3.4, 'Potato Chips,Bbqflvr':
2.2, 'Pretzels': 2.6, 'Tortilla Chip': 2.0, 'Chicknoodl Soup': 7.9, 'Splt
Pea&Hamsoup': 11.1, 'Vegetbeef Soup': 11.2, 'Neweng Clamchwd': 10.9, 'Tomato
Soup': 4.1, 'New E Clamchwd,W/Mlk': 9.5, 'Crm Mshrm Soup,W/Mlk': 6.1, 'Beanbacn
Soup,W/Watr': 7.9, nan: 100.0}, 'Vit_A IU': {'Frozen Broccoli': 5867.4,
'Carrots,Raw': 15471.0, 'Celery, Raw': 53.6, 'Frozen Corn': 106.6,
'Lettuce,Iceberg,Raw': 66.0, 'Peppers, Sweet, Raw': 467.7, 'Potatoes, Baked':
0.0, 'Tofu': 98.6, 'Roasted Chicken': 77.4, 'Spaghetti W/ Sauce': 3055.2,
'Tomato,Red,Ripe,Raw': 766.3, 'Apple,Raw,W/Skin': 73.1, 'Banana': 92.3,
'Grapes': 24.0, 'Kiwifruit,Raw,Fresh': 133.0, 'Oranges': 268.6, 'Bagels': 0.0,
'Wheat Bread': 0.0, 'White Bread': 0.0, 'Oatmeal Cookies': 2.9, 'Apple Pie':
35.2, 'Chocolate Chip Cookies': 101.8, 'Butter,Regular': 152.9, 'Cheddar
Cheese': 296.5, '3.3% Fat,Whole Milk': 307.4, '2% Lowfat Milk': 500.2, 'Skim
Milk': 499.8, 'Poached Eggs': 316.0, 'Scrambled Eggs': 409.2, 'Bologna,Turkey':
0.0, 'Frankfurter, Beef': 0.0, 'Ham,Sliced,Extralean': 0.0, 'Kielbasa,Prk': 0.0,
"Cap'N Crunch": 40.6, 'Cheerios': 1252.2, "Corn Flks, Kellogg'S": 1252.2,
"Raisin Brn, Kellg'S": 1250.2, 'Rice Krispies': 1252.2, 'Special K': 1252.2,
'Oatmeal': 37.4, 'Malt-O-Meal,Choc': 0.0, 'Pizza W/Pepperoni': 281.9, 'Taco':
855.0, 'Hamburger W/Toppings': 126.3, 'Hotdog, Plain': 0.0, 'Couscous': 0.0,
'White Rice': 0.0, 'Macaroni,Ckd': 0.0, 'Peanut Butter': 0.0, 'Pork': 14.7,
'Sardines in Oil': 53.8, 'White Tuna in Water': 68.0, 'Popcorn,Air-Popped':
55.6, 'Potato Chips,Bbqflvr': 61.5, 'Pretzels': 0.0, 'Tortilla Chip': 55.6,
'Chicknoodl Soup': 1308.7, 'Splt Pea&Hamsoup': 4872.0, 'Vegetbeef Soup': 3785.1,
'Neweng Clamchwd': 20.1, 'Tomato Soup': 1393.0, 'New E Clamchwd,W/Mlk': 163.7,
'Crm Mshrm Soup,W/Mlk': 153.8, 'Beanbacn Soup,W/Watr': 888.0, nan: 10000.0},
'Vit_C IU': {'Frozen Broccoli': 160.2, 'Carrots,Raw': 5.1, 'Celery, Raw': 2.8,
'Frozen Corn': 5.2, 'Lettuce,Iceberg,Raw': 0.8, 'Peppers, Sweet, Raw': 66.1,
'Potatoes, Baked': 15.6, 'Tofu': 0.1, 'Roasted Chicken': 0.0, 'Spaghetti W/
Sauce': 27.9, 'Tomato,Red,Ripe,Raw': 23.5, 'Apple,Raw,W/Skin': 7.9, 'Banana':
10.4, 'Grapes': 1.0, 'Kiwifruit,Raw,Fresh': 74.5, 'Oranges': 69.7, 'Bagels':
0.0, 'Wheat Bread': 0.0, 'White Bread': 0.0, 'Oatmeal Cookies': 0.1, 'Apple
Pie': 0.9, 'Chocolate Chip Cookies': 0.0, 'Butter,Regular': 0.0, 'Cheddar
Cheese': 0.0, '3.3% Fat,Whole Milk': 2.3, '2% Lowfat Milk': 2.3, 'Skim Milk':
2.4, 'Poached Eggs': 0.0, 'Scrambled Eggs': 0.1, 'Bologna,Turkey': 0.0,
'Frankfurter, Beef': 10.8, 'Ham,Sliced,Extralean': 7.4, 'Kielbasa,Prk': 5.5,
"Cap'N Crunch": 0.0, 'Cheerios': 15.1, "Corn Flks, Kellogg'S": 15.1, "Raisin
Brn, Kellg'S": 0.0, 'Rice Krispies': 15.1, 'Special K': 15.1, 'Oatmeal': 0.0,
'Malt-O-Meal,Choc': 0.0, 'Pizza W/Pepperoni': 1.6, 'Taco': 2.2, 'Hamburger
W/Toppings': 2.6, 'Hotdog, Plain': 0.1, 'Couscous': 0.0, 'White Rice': 0.0,
'Macaroni,Ckd': 0.0, 'Peanut Butter': 0.0, 'Pork': 0.0, 'Sardines in Oil': 0.0,
'White Tuna in Water': 0.0, 'Popcorn,Air-Popped': 0.0, 'Potato Chips,Bbqflvr':
9.6, 'Pretzels': 0.0, 'Tortilla Chip': 0.0, 'Chicknoodl Soup': 0.0, 'Splt
Pea&Hamsoup': 7.0, 'Vegetbeef Soup': 4.8, 'Neweng Clamchwd': 4.8, 'Tomato Soup':
133.0, 'New E Clamchwd,W/Mlk': 3.5, 'Crm Mshrm Soup,W/Mlk': 2.2, 'Beanbacn
Soup,W/Watr': 1.5, nan: 5000.0}, 'Calcium mg': {'Frozen Broccoli': 159.0,
'Carrots,Raw': 14.9, 'Celery, Raw': 16.0, 'Frozen Corn': 3.3,
'Lettuce,Iceberg,Raw': 3.8, 'Peppers, Sweet, Raw': 6.7, 'Potatoes, Baked': 22.7,
'Tofu': 121.8, 'Roasted Chicken': 21.9, 'Spaghetti W/ Sauce': 80.2,
'Tomato,Red,Ripe,Raw': 6.2, 'Apple,Raw,W/Skin': 9.7, 'Banana': 6.8, 'Grapes':
3.4, 'Kiwifruit,Raw,Fresh': 19.8, 'Oranges': 52.4, 'Bagels': 21.0, 'Wheat
Bread': 10.8, 'White Bread': 26.2, 'Oatmeal Cookies': 6.7, 'Apple Pie': 3.1,
'Chocolate Chip Cookies': 6.2, 'Butter,Regular': 1.2, 'Cheddar Cheese': 202.0,
'3.3% Fat,Whole Milk': 291.3, '2% Lowfat Milk': 296.7, 'Skim Milk': 302.3,
'Poached Eggs': 24.5, 'Scrambled Eggs': 42.6, 'Bologna,Turkey': 23.8,
'Frankfurter, Beef': 9.0, 'Ham,Sliced,Extralean': 2.0, 'Kielbasa,Prk': 11.4,
"Cap'N Crunch": 4.8, 'Cheerios': 48.6, "Corn Flks, Kellogg'S": 0.9, "Raisin Brn,
Kellg'S": 12.9, 'Rice Krispies': 4.0, 'Special K': 8.2, 'Oatmeal': 18.7, 'Malt-
O-Meal,Choc': 23.1, 'Pizza W/Pepperoni': 64.6, 'Taco': 220.6, 'Hamburger
W/Toppings': 51.4, 'Hotdog, Plain': 23.5, 'Couscous': 7.2, 'White Rice': 7.9,
'Macaroni,Ckd': 4.9, 'Peanut Butter': 13.1, 'Pork': 59.9, 'Sardines in Oil':
91.7, 'White Tuna in Water': 3.4, 'Popcorn,Air-Popped': 2.8, 'Potato
Chips,Bbqflvr': 14.2, 'Pretzels': 10.2, 'Tortilla Chip': 43.7, 'Chicknoodl
Soup': 27.1, 'Splt Pea&Hamsoup': 33.6, 'Vegetbeef Soup': 32.6, 'Neweng
Clamchwd': 82.8, 'Tomato Soup': 27.6, 'New E Clamchwd,W/Mlk': 186.0, 'Crm Mshrm
Soup,W/Mlk': 178.6, 'Beanbacn Soup,W/Watr': 81.0, nan: 1500.0}, 'Iron mg':
{'Frozen Broccoli': 2.3, 'Carrots,Raw': 0.3, 'Celery, Raw': 0.2, 'Frozen Corn':
0.3, 'Lettuce,Iceberg,Raw': 0.1, 'Peppers, Sweet, Raw': 0.3, 'Potatoes, Baked':
4.3, 'Tofu': 6.2, 'Roasted Chicken': 1.8, 'Spaghetti W/ Sauce': 2.3,
'Tomato,Red,Ripe,Raw': 0.6, 'Apple,Raw,W/Skin': 0.2, 'Banana': 0.4, 'Grapes':
0.1, 'Kiwifruit,Raw,Fresh': 0.3, 'Oranges': 0.1, 'Bagels': 1.0, 'Wheat Bread':
```

```
0.7, 'White Bread': 0.8, 'Oatmeal Cookies': 0.5, 'Apple Pie': 0.1, 'Chocolate
Chip Cookies': 0.4, 'Butter,Regular': 0.0, 'Cheddar Cheese': 0.2, '3.3%
Fat,Whole Milk': 0.1, '2% Lowfat Milk': 0.1, 'Skim Milk': 0.1, 'Poached Eggs':
0.7, 'Scrambled Eggs': 0.7, 'Bologna,Turkey': 0.4, 'Frankfurter, Beef': 0.6,
'Ham,Sliced,Extralean': 0.2, 'Kielbasa,Prk': 0.4, "Cap'N Crunch": 7.5,
'Cheerios': 4.5, "Corn Flks, Kellogg'S": 1.8, "Raisin Brn, Kellg'S": 16.8, 'Rice
Krispies': 1.8, 'Special K': 4.5, 'Oatmeal': 1.6, 'Malt-O-Meal,Choc': 4.7,
'Pizza W/Pepperoni': 0.9, 'Taco': 2.4, 'Hamburger W/Toppings': 2.5, 'Hotdog,
Plain': 2.3, 'Couscous': 0.3, 'White Rice': 0.9, 'Macaroni,Ckd': 1.0, 'Peanut
Butter': 0.6, 'Pork': 0.4, 'Sardines in Oil': 0.7, 'White Tuna in Water': 0.5,
'Popcorn,Air-Popped': 0.8, 'Potato Chips,Bbqflvr': 0.5, 'Pretzels': 1.2,
'Tortilla Chip': 0.4, 'Chicknoodl Soup': 1.5, 'Splt Pea&Hamsoup': 2.1,
'Vegetbeef Soup': 2.2, 'Neweng Clamchwd': 2.8, 'Tomato Soup': 3.5, 'New E
Clamchwd,W/Mlk': 1.5, 'Crm Mshrm Soup,W/Mlk': 0.6, 'Beanbacn Soup,W/Watr': 2.0,
nan: 40.0}}
```

[97]:

```
nutrient_name = diet_df1.columns.tolist()[1:]
print(nutrient_name)
```

```
['Calories', 'Cholesterol mg', 'Total_Fat g', 'Sodium mg', 'Carbohydrates g',
'Dietary_Fiber g', 'Protein g', 'Vit_A IU', 'Vit_C IU', 'Calcium mg', 'Iron mg']
```

[115]:

```
prob=LpProblem('Food_Optimization',LpMinimize)
foodVars = LpVariable.dicts('Foods',food,0)
prob += lpSum([cost[f]*foodVars[f] for f in food]), 'Total_Cost'
for i in range(len(nutrient_name)):
    prob += lpSum([food_nutrient[nutrient_name[i]][j] * foodVars[j] for j in␣
 ↪food]) >= exp_nutrient['min'][i], 'min_nutrient' + str(i)
    prob += lpSum([food_nutrient[nutrient_name[i]][j] * foodVars[j] for j in␣
 ↪food]) <= exp_nutrient['max'][i], 'max_nutrient' + str(i)
prob.solve()
```

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /opt/anaconda3/lib/python3.12/site-
packages/pulp/apis/../solverdir/cbc/osx/i64/cbc /var/folders/8j/cchz4dpd4r5229b5
6p8bk_7r0000gn/T/071314bbd5114edbb5180c6ee0e6ff0b-pulp.mps -timeMode elapsed
-branch -printingOptions all -solution /var/folders/8j/cchz4dpd4r5229b56p8bk_7r0
000gn/T/071314bbd5114edbb5180c6ee0e6ff0b-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 27 COLUMNS
At line 1286 RHS
At line 1309 BOUNDS
At line 1310 ENDATA
Problem MODEL has 22 rows, 64 columns and 1194 elements
```

```
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 22 (0) rows, 64 (0) columns and 1194 (0) elements
0  Obj 0 Primal inf 21.63092 (11)
9  Obj 4.3371168
Optimal - objective value 4.3371168
Optimal objective 4.33711681 - 9 iterations time 0.002
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.00   (Wallclock seconds):       0.02
```

[115]: 1

[119]:

```
for var in prob.variables():
    if var.varValue > 0:
        print(str(var.varValue) + " units of " + str(var))
```

```
52.64371 units of Foods_Celery,_Raw
0.25960653 units of Foods_Frozen_Broccoli
63.988506 units of Foods_Lettuce,Iceberg,Raw
2.2929389 units of Foods_Oranges
0.14184397 units of Foods_Poached_Eggs
13.869322 units of Foods_Popcorn,Air_Popped
```

[121]:

```
print('Total Cost is ' + '$' + str(round(value(prob.objective),2)))
```

```
Total Cost is $4.34
```

[133]:

```
prob_a = LpProblem('Food_optimization_a', LpMinimize)
foodVars = LpVariable.dicts('Foods',food,0)
foodVars_selected = LpVariable.dicts('Food_Selected',food,0,1,LpBinary)
prob_a += lpSum([cost[f]*foodVars[f] for f in food]), 'Total Cost'
for i in range(len(nutrient_name)):
    prob_a+= lpSum([food_nutrient[nutrient_name[i]][j]*foodVars[j] for j in␣
 ↪food]) >= exp_nutrient['min'][i],'min_nutrient ' + str(i)
    prob_a+= lpSum([food_nutrient[nutrient_name[i]][j]*foodVars[j] for j in␣
 ↪food]) >= exp_nutrient['max'][i],'max_nutrient ' + str(i)
for item in food:
    prob_a += foodVars[item] >= 0.1 * foodVars_selected[item]
for item in food:
    prob_a += foodVars_selected[item] >= foodVars[item]*0.0000001
prob_a += foodVars_selected['Frozen Broccoli'] + foodVars_selected['Celery,␣
 ↪Raw'] <=1
prob_a += foodVars_selected['Roasted Chicken'] + foodVars_selected['Poached␣
 ↪Eggs'] \
        + foodVars_selected['Scrambled Eggs'] +␣
 ↪foodVars_selected['Bologna,Turkey'] \
        + foodVars_selected['Frankfurter, Beef'] +␣
 ↪foodVars_selected['Ham,Sliced,Extralean'] \
        + foodVars_selected['Kielbasa,Prk'] + foodVars_selected['Pizza W/
 ↪Pepperoni'] \
        + foodVars_selected['Hamburger W/Toppings'] \
        + foodVars_selected['Hotdog, Plain'] + foodVars_selected['Pork'] \
        + foodVars_selected['Sardines in Oil'] + foodVars_selected['White Tuna␣
 ↪in Water'] \
        + foodVars_selected['Chicknoodl Soup'] + foodVars_selected['Splt␣
 ↪Pea&Hamsoup'] \
        + foodVars_selected['Vegetbeef Soup'] + foodVars_selected['Neweng␣
 ↪Clamchwd'] \
        + foodVars_selected['New E Clamchwd,W/Mlk'] +␣
 ↪foodVars_selected['Beanbacn Soup,W/Watr'] >=3
prob_a.solve()
for var in prob_a.variables():
    if var.varValue > 0 and "Food_Selected" not in var.name:
        print(str(var.varValue) + " units of " + str(var))
print("Total Cost is " + "$" + str(round(value(prob_a.objective),2)))
```

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /opt/anaconda3/lib/python3.12/site-
packages/pulp/apis/../solverdir/cbc/osx/i64/cbc /var/folders/8j/cchz4dpd4r5229b5
6p8bk_7r0000gn/T/840294e8cd8f41b8819a26b8bd71ba11-pulp.mps -timeMode elapsed
-branch -printingOptions all -solution /var/folders/8j/cchz4dpd4r5229b56p8bk_7r0
000gn/T/840294e8cd8f41b8819a26b8bd71ba11-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 157 COLUMNS
At line 1821 RHS
At line 1974 BOUNDS
At line 2039 ENDATA
Problem MODEL has 152 rows, 128 columns and 1471 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Continuous objective value is 5.27892 - 0.00 seconds
Cgl0004I processed model has 141 rows, 128 columns (64 integer (64 of which
binary)) and 874 elements
Cbc0038I Initial state - 1 integers unsatisfied sum - 3.38015e-06
Cbc0038I Pass   1: suminf.    0.00000 (1) obj. 5.77938 iterations 69
Cbc0038I Pass   2: suminf.    0.00000 (1) obj. 5.77938 iterations 0
Cbc0038I Pass   3: suminf.    0.00000 (1) obj. 6.33298 iterations 24
Cbc0038I Pass   4: suminf.    0.00000 (1) obj. 6.33298 iterations 0
Cbc0038I Pass   5: suminf.    0.00000 (1) obj. 6.33298 iterations 0
```

```
Cbc0038I Pass   6: suminf.    0.00000 (1) obj. 6.61854 iterations 48
Cbc0038I Pass   7: suminf.    0.00000 (1) obj. 6.61854 iterations 0
Cbc0038I Pass   8: suminf.    0.00000 (1) obj. 6.61854 iterations 0
Cbc0038I Pass   9: suminf.    0.00000 (0) obj. 10.5312 iterations 27
Cbc0038I Solution found of 10.5312
Cbc0038I Relaxing continuous gives 6.37534
Cbc0038I Rounding solution of 5.77938 is better than previous of 6.37534

Cbc0038I Before mini branch and bound, 25 integers at bound fixed and 24
continuous
Cbc0038I Full problem 141 rows 128 columns, reduced to 90 rows 79 columns
Cbc0038I Mini branch and bound improved solution from 5.77938 to 5.27892 (0.01
seconds)
Cbc0038I After 0.02 seconds - Feasibility pump exiting with objective of 5.27892
- took 0.01 seconds
Cbc0012I Integer solution of 5.2789227 found by feasibility pump after 0
iterations and 0 nodes (0.02 seconds)
Cbc0001I Search completed - best objective 5.278922710215079, took 0 iterations
and 0 nodes (0.02 seconds)
Cbc0035I Maximum depth 0, 0 variables fixed on reduced cost
Cuts at root node changed objective from 5.27892 to 5.27892
Probing was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
Gomory was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
Knapsack was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
Clique was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
MixedIntegerRounding2 was tried 0 times and created 0 cuts of which 0 were
active after adding rounds of cuts (0.000 seconds)
FlowCover was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
TwoMirCuts was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
ZeroHalf was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)

Result - Optimal solution found

Objective value:                5.27892271
Enumerated nodes:               0
Total iterations:               0
Time (CPU seconds):             0.01
Time (Wallclock seconds):       0.02

Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.01   (Wallclock seconds):       0.02
```

```
31.186798 units of Foods_Frozen_Broccoli
0.1 units of Foods_Kielbasa,Prk
2.4021706 units of Foods_Peanut_Butter
1.0266667 units of Foods_Poached_Eggs
0.21249543 units of Foods_Potatoes,_Baked
0.1 units of Foods_Scrambled_Eggs
Total Cost is $5.28
```

2 Cost-Optimized Diet Plan: A Linear Programming Approach

2.1 Introduction

The diet problem is a fundamental optimization challenge aimed at determining the cheapest possible diet that meets all daily nutritional requirements. Originally developed for the U.S. Army, this problem has modern applications in healthcare, food assistance programs, and personal budgeting.

This study implements **linear programming (LP)** to optimize diet selection using real-world nutritional data. We solve two models: 1. **Basic Optimization Model** – A purely cost-minimizing approach
2. **Enhanced Model** – Introduces practical constraints to ensure portion control, food variety, and real-world applicability

---

2.2 Formulating the Diet Optimization Model

2.2.1 Decision Variables

For each food *i*: - $ x_i $: The number of servings of food $ i $ consumed per day

Enhanced Model Additional Variables: - $ y_i $: A binary variable indicating whether food $ i $ is included in the diet

2.2.2 Objective Function

Minimize the total daily cost of food:

$$\text{Minimize} \sum_i c_i x_i$$

where $ c_i $ represents the cost per serving of each food item.

2.2.3 Constraints

1. **Nutritional Balance:** Each selected food must meet daily minimum and maximum nutritional requirements (calories, protein, fat, carbohydrates, vitamins, and minerals).
2. **Minimum Serving Size:** If a food is included, at least **1/10 serving** must be chosen:

$$x_i \geq 0.1 y_i, \quad \forall i$$

3. **Celery vs. Broccoli Restriction:** Since many people dislike both, **at most one can be selected**:
$$y_{\text{Celery}} + y_{\text{Broccoli}} \leq 1$$
4. **Protein Variety:** At least **three** different sources of **meat, poultry, fish, or eggs** must be included:
$$y_{\text{Meat}_1} + y_{\text{Meat}_2} + y_{\text{Meat}_3} \geq 3$$

2.3 Results & Comparison

2.3.1 1. Basic Optimization Model

Food Item	**Quantity (Servings)**
Celery, Raw	52.64
Frozen Broccoli	0.26
Iceberg Lettuce	63.99
Oranges	2.29
Poached Eggs	0.14
Air-Popped Popcorn	13.87

Total Cost: $4.34 **Advantages:**
Extremely cheap—satisfies all nutritional requirements at minimal cost

Disadvantages:
Impractical quantities—consuming **63 servings of lettuce per day** is unrealistic
Lacks protein variety—minimal inclusion of eggs, no meat
Excessive fiber intake—which could cause digestive discomfort

---

2.3.2 2. Enhanced Model with Additional Constraints

Food Item	**Quantity (Servings)**
Frozen Broccoli	31.19
Kielbasa, Pork	0.1
Peanut Butter	2.40
Poached Eggs	1.03
Baked Potatoes	0.21
Scrambled Eggs	0.1

Total Cost: $5.28 **Advantages:**
More realistic portions—no extreme quantities of a single food
Includes sufficient protein variety—meat (kielbasa), eggs, and peanut butter
Balanced intake of macronutrients—improves dietary feasibility

Disadvantages:
Slightly higher cost—but still very affordable at $5.28/day

2.4 Nutritional & Practical Considerations

2.4.1 Macronutrient Balance

- **Basic Model:** Overwhelmingly high in **fiber and carbohydrates** but lacking in sufficient **protein and fat**.

- **Enhanced Model:** Ensures **protein, fat, and carbohydrate balance**, making it more **nutritionally complete**.

2.4.2 Feasibility in Real Life

- The **basic model** is not realistic for any individual, given its reliance on excessive servings of vegetables and popcorn.

- The **enhanced model** mimics a **more normal diet**, including **protein sources** and **reasonable portion sizes**.

2.4.3 Cost vs. Nutrition Trade-Off

- The **basic model achieves the lowest possible cost** ($4.34) but lacks **practicality**.

- The **enhanced model** slightly increases cost **(+$0.94 per day)** to create a **viable meal plan**.

2.5 Sensitivity Analysis: How Changes Affect the Model

2.5.1 1. Impact of Price Fluctuations

- If the **cost of eggs or peanut butter increases**, the model might shift to cheaper protein alternatives, like legumes.

- If **vegetables become more expensive**, the diet might shift towards **grains or dairy products**.

2.5.2 2. Impact of Additional Constraints

- Removing the **celery vs. broccoli restriction** could lower costs further.

- Increasing the **minimum serving size** requirement (e.g., from 0.1 to 0.5 servings) might lead to different food selections.

2.5.3 3. Adding a Dietary Preference Constraint

- If a person is **vegetarian**, the model would exclude meat and favor **plant-based proteins**.

- If someone prefers a **low-carb diet**, the model would prioritize **meat, eggs, and non-starchy vegetables** over grains.

2.6 Conclusion

The basic model (4.34 per day) achieves the lowest cost but is nutritionally imbalanced and unrealistic. The enhanced model (5.28 per day) is a better solution, balancing cost, nutrition, and practicality.

2.6.1 Key Takeaways:

Linear programming effectively minimizes diet costs
Real-world constraints make the model significantly more applicable
Diet planning requires a balance between cost and feasibility

Future Work: We could extend this analysis by:
Optimizing for **nutritional diversity** beyond just cost
Implementing **different diet preferences** (vegan, low-carb, high-protein)
Incorporating **regional price variations** to adapt the model for different markets

[ ]: