DM545 – Linear and Integer Programming

Answers to the Take-home Assignment, Winter 2025

In this assignment I'm sometimes using functions that I've defined in previous subtasks. I'm mentioning it because of the split-assignments-by-subtask rule. I try to indicate when I do that, but I am not defining them again, as I think it would make the assignment unnecessearly lengthy.

Additionally, I'm using the tableau function from the course github files.

Task 1

Subtask 1.a

We notice that the variables are bounded on both sides, so as the first inspection we can put those variables - whose coefficients in the objective function are positive - to their maximum value, and vice-versa for the negative ones, and check if the constraints are satisfied.

$$x_1 = 5, x_2 = 1, x_3 = -2$$

$$4 \cdot 5 + 5 \cdot 1 - 7 \cdot (-2) = 39$$
$$-5 - 1 + (-2) = -8 \leq 2$$
$$-5 \cdot 5 + 10 \cdot (-2) = -45 \leq 10$$

As we can see - all constraints are satisfied and the objective function is maximized for all possible values of $\mathbf{x}$, therefore $x_1 = 5, x_2 = 1, x_3 = -2$ is a feasible and optimal solution for this problem.

Subtask 1.b

1. $s = 2, t = 1$
2. $s = 1, t = 1$
3. impossible
4. $s = 0, t = 0$

The three examples can be easily checked, but I suspect a few more words should be said about the third point.

For positive values s and t - x's can just be put to 0 which gives a feasible solution.
For negative values s and t - the constraint will be satisfied by any pair of non-negative x's.
For one positive and one negative - similarly, the x with the positive coefficient can be put to 0 and the other can be anything.

Task 2

Subtask 2.a

In order to write the tableau, we need to bring the problem into the equational standard form:

$$\begin{array}{ll} \min & z = 3x_1 + 2x_2 + 7x_3 \\ s.t. & -x_1 + x_2 = 10 \\ & 2x_1 - x_2 + x_3 \geq 10 \\ & x_1, x_2, x_3 \geq 0 \end{array} \quad \rightarrow \quad \begin{array}{ll} \max & w = -3x_1 - 2x_2 - 7x_3 \\ s.t. & -x_1 + x_2 + 0x_3 = 10 \\ & -2x_1 + x_2 - x_3 + s_1 = -10 \\ & x_1, x_2, x_3, s_1 \geq 0 \end{array}$$

```
import numpy as np

T = np.array([
    [-1, 1, 0, 0, 0, 10],
    [-2, 1, -1, 1, 0, -10],
    [-3, -2, -7, 0, 1, 0]
])
```

```
from util import tableau
tableau(T)
```

```
|------+------+------+------+------+------+
|   x1 |   x2 |   x3 |   x4 |   -z |    b |
|------+------+------+------+------+------+
|   -1 |    1 |    0 |    0 |    0 |   10 |
|   -2 |    1 |   -1 |    1 |    0 |  -10 |
|------+------+------+------+------+------+
|   -3 |   -2 |   -7 |    0 |    1 |    0 |
|------+------+------+------+------+------+
```

However, to be able to say anything about the optimality the tableau needs to be in canonical form - we can achieve that by bringing eg. x_2 in the basis:

```
def pivot(T, row, col):
    T[row] /= T[row][col] # normalize to 1
    for j in range(len(T)):
        if j != row:
            T[j] -= T[row] * T[j][col] # all other rows in col -> 0

pivot(T, 0, 1) # row 1 col 2
tableau(T)
```

```
|------+------+------+------+------+------+
|   x1 |   x2 |   x3 |   x4 |   -z |    b |
|------+------+------+------+------+------+
|   -1 |    1 |    0 |    0 |    0 |   10 |
```

-1	0	-1	1	0	-20
-5	0	-7	0	1	20

For a maximization problem (which it is in this form) the optimality condition is that all reduced costs must be negative (no further improvement). We can see that it is the case.

However, the current tableau is clearly not optimal (negative b-term) - we would need to apply dual simplex (basically just pivot on row 2 column 1) and check if the feasibility has been retrieved.

Subtask 2.b

According to the largest pivoting rule x_6 would have to be brought in the basis, however - it turns out that there is no most constraining constraint for the new value of x_6.

More formally:

$$\max_{\theta}\{\bar{b} - a_{.6}\theta \geq 0\} = \infty$$

Which means that the problem is unbounded.

No matter what other pivoting rules we might apply, the conclusion will either be the same, or to bring x_3 in the basis (as it is the only other non-basic variable with a positive reduced cost) - so let's see what happens then:

For the iteration I will use the pivot function, which I've defined in the previous subtask.

```
T = np.array([
    [0, 0, 0, 1, 1, 0, 0, 0],
    [0, 1, 1, 2, 0, -1, 0, 30],
    [1, 0, 1, 1, 0, -1, 0, 20],
    [0, 0, 2, -7, 0, 5, 1, -120]
])
```

```
pivot(T, 2, 2) # column 3 row 3 because of ratio test
tableau(T)
```

```
|-------+-------+-------+-------+-------+-------+-------+-------+
|    x1 |    x2 |    x3 |    x4 |    x5 |    x6 |    -z |     b |
|-------+-------+-------+-------+-------+-------+-------+-------+
|     0 |     0 |     0 |     1 |     1 |     0 |     0 |     0 |
|    -1 |     1 |     0 |     1 |     0 |     0 |     0 |    10 |
|     1 |     0 |     1 |     1 |     0 |    -1 |     0 |    20 |
|-------+-------+-------+-------+-------+-------+-------+-------+
|    -2 |     0 |     0 |    -9 |     0 |     7 |     1 |  -160 |
|-------+-------+-------+-------+-------+-------+-------+-------+
```

And the conclusion is the same - The only variable left with a positive reduced cost is x_6 and it's value upon bringing into the basis is unconstrained.

Task 3

Subtask 3.a

- solution: $x_1 = 5, x_2 = 15$, objective value: 110
- reduced costs: $\bar{c} = [0, 0, -2, -2]^T$
- dual variables: $y_1 = 2, y_2 = 2$
- shadow prices: equal to the dual variables - $[2, 2]^T$
- no slack-capacity (as all slack variables are not in the basis, or because the dual variables are non-zero $\implies$ from the complementary slackness theorem all constraints are binding)

Task 4

Subtask 4.a

Integrate the constraints into the objective function:

$$P = \begin{array}{ll} \min & \sum_i c_i y_i \\ \forall_i & \sum_j a_j x_{ij} \leq b y_i \\ \forall_j & \sum_i x_{ij} = 1 \\ \forall_i & y_i \leq 1 \\ \forall_{ij} & x_{ij} \geq 0 \\ \forall_i & y_i \geq 0 \end{array} \quad \rightarrow \quad PR(\alpha, \beta, \gamma) = \begin{array}{ll} \min_{x,y} & \left\{ \begin{array}{c} \sum_i c_i y_i \\ + \sum_i \alpha_i (b y_i - \sum_j a_j x_{ij}) \\ + \sum_j \beta_j (1 - \sum_i x_{ij}) \\ + \sum_i \gamma_i (1 - y_i) \end{array} \right\} \\ \forall_{ij} & x_{ij} \geq 0 \\ \forall_i & y_i \geq 0 \end{array}$$

Because of the original constraints we have:

$$(\mathbf{x}, \mathbf{y}) = \arg\min_{x,y} \left\{ \begin{array}{c} \sum_i c_i y_i \\ + \sum_i \alpha_i (b y_i - \sum_j a_j x_{ij}) \\ + \sum_j \beta_j (1 - \sum_i x_{ij}) \\ + \sum_i \gamma_i (1 - y_i) \end{array} \right\} \quad \Longrightarrow \quad (\mathbf{x}, \mathbf{y}) = \arg\min_{x,y} \left\{ \begin{array}{c} \sum_i c_i y_i \\ + \sum_i \alpha_i (b y_i - \sum_j a_j x_{ij}) \\ + \sum_j \beta_j (1 - \sum_i x_{ij}) \\ + \sum_i \gamma_i (1 - y_i) \end{array} \right\}$$

$$\begin{array}{ll} \wedge \forall_{ij} & \sum_j a_j \mathbf{x}_{ij} \leq b \mathbf{y}_i \\ \wedge \forall_{ij} & \sum_i \mathbf{x}_{ij} = 1 \\ \wedge \forall_i & \mathbf{y}_i \leq 1 \end{array} \qquad \begin{array}{ll} \wedge \forall_i & \alpha_i \leq 0 \\ \wedge \forall_j & \beta_j \in \mathbb{R} \\ \wedge \forall_i & \gamma_i \leq 0 \end{array}$$

Which yields the bounds for the dual problem:

- $\alpha \leq \mathbb{0}$
- $\beta \in \mathbb{R}^m$
- $\gamma \leq \mathbb{0}$

And also implies:

$$\Longrightarrow \; opt(PR(\alpha, \beta, \gamma)) \leq opt(P)$$

So to best "approximate" $opt(P)$ we maximize $opt(PR(\alpha, \beta, \gamma))$:

$$\max_{\alpha,\beta,\gamma} \{opt(PR(\alpha, \beta, \gamma))\} =$$

$$= \max_{\alpha,\beta,\gamma} \left\{ \min_{x,y} \left\{ \begin{array}{c} \sum_i c_i y_i \\ + \sum_i \alpha_i (b y_i - \sum_j a_j x_{ij}) \\ + \sum_j \beta_j (1 - \sum_i x_{ij}) \\ + \sum_i \gamma_i (1 - y_i) \end{array} \right\} \right\} = \max_{\alpha,\beta,\gamma} \left\{ \min_{x,y} \left\{ \begin{array}{c} \sum_j \beta_j + \sum_i \gamma_i \\ + \sum_i \sum_j (-\alpha_i a_j - \beta_j) x_{ij} \\ + \sum_i (c_i + \alpha_i b - \gamma_i) y_i \end{array} \right\} \right\}$$

In order to maximize this quantity we need:

- $\forall_{ij}(-\alpha_i a_j - \beta_j \geq 0)$
- $\forall_i(c_i + \alpha_i b - \gamma_i \geq 0)$

Because else:

$$\begin{array}{c} \exists_i (c_i + \alpha_i b - \gamma_i < 0) \\ \implies \min_{x,y}\{(c_i + \alpha_i b - \gamma_i) y_i + ...\} = -\infty \\ \implies \max_{\alpha,\beta,\gamma}\{\min_{x,y}\{...\}\} = -\infty \end{array}$$

For $y_i = \infty$, and vice-versa for the coefficient next to x_{ij}.

Now we can observe that with these additional constraints:

$$\begin{array}{c} \forall_{ij}(-\alpha_i a_j - \beta_j \geq 0) \\ \wedge \forall_i (c_i + \alpha_i b - \gamma_i \geq 0) \end{array} \implies \arg\min_{x,y} \left\{ \begin{array}{c} \sum_j \beta_j + \sum_i \gamma_i \\ + \sum_i \sum_j (-\alpha_i a_j - \beta_j) x_{ij} \\ + \sum_i (c_i + \alpha_i b - \gamma_i) y_i \end{array} \right\} = (\mathbb{0}, \mathbb{0})$$

$$\implies \max_{\alpha,\beta,\gamma} \{opt(PR(\alpha,\beta,\gamma))\} = \max_{\alpha,\beta,\gamma} \left\{ \sum_j \beta_j + \sum_i \gamma_i \right\}$$

And therefore we're left with the final dual problem:

$$D = \begin{array}{ll} \max & \sum_j \beta_j + \sum_i \gamma_i \\ \forall_{ij} & \alpha_i a_j + \beta_j \leq 0 \\ \forall_i & -\alpha_i b + \gamma_i \leq c_i \\ \forall_i & \alpha_i \leq 0 \\ \forall_j & \beta_j \in \mathbb{R} \\ \forall_i & \gamma_i \leq 0 \end{array}$$

Q.E.D.

Task 5

Subtask 5.a

To determine if the solution is optimal, we need to calculate the reduced costs corresponding to given non-basis variables. For that we can use the formula from the lecture:

$$\bar{c}_N^T = c_N^T - c_B^T A_B^{-1} A_N$$

And first let's bring the problem into maximization equational standard form for it's relaxation:

$$\begin{array}{ll} \min & x_1 + x_2 \\ & -3x_1 + 7x_2 \leq -21 \\ & x_1 - 4x_2 \leq 4 \\ & x_1, x_2 \in \mathbf{Z}^+ \end{array} \quad \rightarrow \quad \begin{array}{ll} \max & -x_1 - x_2 \\ & -3x_1 + 7x_2 + s_1 = -21 \\ & x_1 - 4x_2 + s_2 = 4 \\ & x_1, x_2, s_1, s_2 \geq 0 \end{array}$$

```
n = 2
m = 2

# model in standard form
A = np.array([
    [-3, 7],
    [1, -4]
])
b = np.array([-21, 4])
c = np.array([-1, -1])

# equational standard form - add slacks
A = np.hstack([A, np.identity(m)])
c = np.hstack([c, np.zeros(m)])

# new basis
B = [0, 1] # 1, 2
N = [2, 3] # 3, 4

# not perfectly efficient, but correct way to calculate c_bar
A_Binv = np.linalg.inv(A[:, B])
c_bar = c[N] - c[B] @ A_Binv @ A[:, N] # c bar
```

```
c_bar
```

```
['-1', '-2']
```

Since this is a maximization problem and all non-basic reduced costs are negative, the solution associated with basis $B = \{1, 2\}$ is optimal.

Subtask 5.b

To make life easier I will print the tableau corresponding to the solution, using revised simplex:

```
# A and c - basic
T[:m, B] = np.identity(m)
T[-1, B] = np.zeros(m)

# A and c - nonbasic
T[:m, N] = A_Binv @ A[:, N]
T[-1, N] = c[N] - c[B] @ T[:m, N]

# b and d
T[:m, -1] = A_Binv @ b
T[-1, -1] = - c[B] @ T[:m, -1]
```

```
tableau(T)
```

```
|-------+-------+-------+-------+-------+-------+
|    x1 |    x2 |    x3 |    x4 |    -z |     b |
|-------+-------+-------+-------+-------+-------+
|     1 |     0 |  -4/5 |  -7/5 |     0 |  56/5 |
|     0 |     1 |  -1/5 |  -3/5 |     0 |   9/5 |
|-------+-------+-------+-------+-------+-------+
|     0 |     0 |    -1 |    -2 |     1 |    13 |
|-------+-------+-------+-------+-------+-------+
```

Now we have a clear view at what's going on. We're in a basic optimal solution, but infeasible for the integer problem and both basic variables are fractional.

The Gomory cuts are of form:

$$\sum_{j \in N} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_i \geq \bar{b}_i - \lfloor \bar{b}_i \rfloor$$

Which in the equational standard form will become:

$$\sum_{j \in N} (-\bar{a}_{ij} + \lfloor \bar{a}_{ij} \rfloor) x_i + s_{m+i} = -\bar{b}_i + \lfloor \bar{b}_i \rfloor$$

```
def gomory(T, i):
    T = np.column_stack([ # new column for slack
        T[:, :-2],
        np.zeros_like(T[:, 0]),
        T[:, -2:]
    ])

    cut = -T[i, :] + np.floor(T[i, :]) # gomory cut row
    cut[-3] = 1 # for slack

    T = np.vstack([ # insert row as last
        T[:-1, :],
        cut,
        T[-1, :]
    ])
    return T

T = gomory(T, 0)
T = gomory(T, 1)
m += 2
```

```
tableau(T)
```

```
|------+------+------+------+------+------+------+------+
|   x1 |   x2 |   x3 |   x4 |   x5 |   x6 |   -z |    b |
|------+------+------+------+------+------+------+------+
|    1 |    0 | -4/5 | -7/5 |    0 |    0 |    0 | 56/5 |
|    0 |    1 | -1/5 | -3/5 |    0 |    0 |    0 |  9/5 |
|    0 |    0 | -1/5 | -3/5 |    1 |    0 |    0 | -1/5 |
|    0 |    0 | -4/5 | -2/5 |    0 |    1 |    0 | -4/5 |
|------+------+------+------+------+------+------+------+
|    0 |    0 |   -1 |   -2 |    0 |    0 |    1 |   13 |
|------+------+------+------+------+------+------+------+
```

And here we have the updated tableau with the Chvatal-Gomory cuts. As expected - it is infeasible.

Now we can extract the cuts' inequalities on original variables by bringing the tableau into the initial basis:

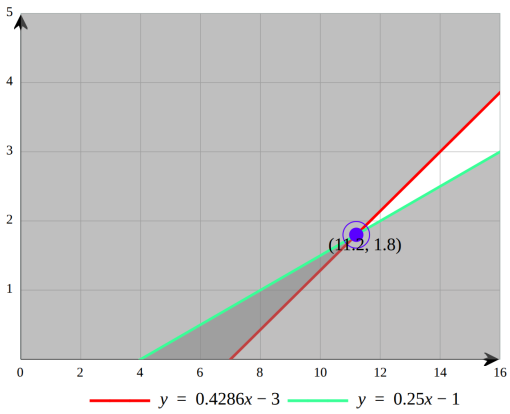
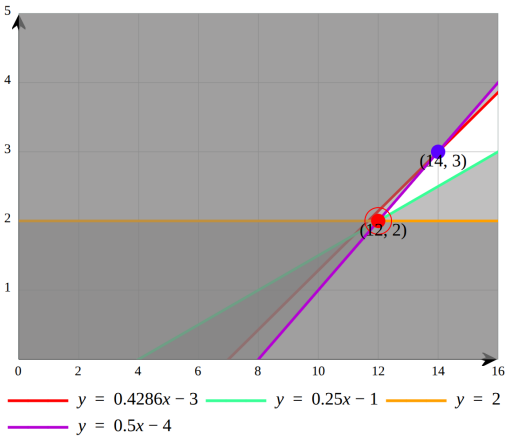
```
pivot(T, 0, 2) # row 1 col 3 - slack 1
pivot(T, 1, 3) # row 2 col 4 - slack 2
tableau(T)
```

x1	x2	x3	x4	x5	x6	-z	b
-3	7	1	0	0	0	0	-21
1	-4	0	1	0	0	0	4
0	-1	0	0	1	0	0	-2
-2	4	0	0	0	1	0	-16
-1	-1	0	0	0	0	1	0

So the Gomory cuts are:

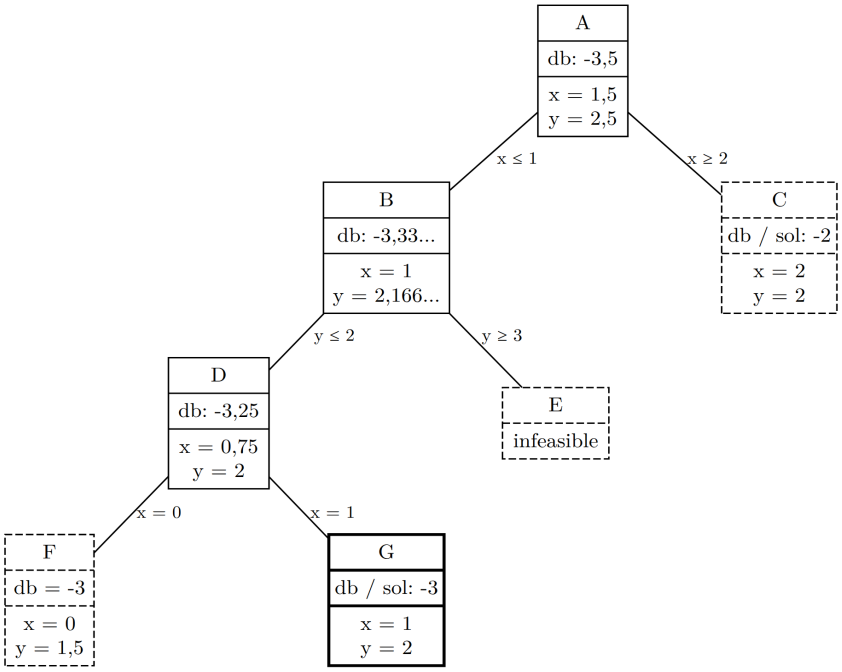
$$-x_2 \leq -2$$
$$-2x_1 + 4x_2 \leq -16$$

Subtask 5.c

Without Gomory Cuts	With Gomory Cuts
	

Task 6

Subtask 6.a



Code I used to solve linear programming problems at the nodes:

```
from mip import Model # mip solver

m = Model()

x = m.add_var(lb = 0) # variables
y = m.add_var(lb = 0)

m.objective = x - 2 * y # objective (min)

# === Changed For Each Node ===
m += -4 * x + 6 * y <= 9 # constraints
m += x + y <= 4
m += x == 1
m += y <= 2
# ============================

m.optimize()

print(f"x = {x.x}, y = {y.x}")
```

Task 7

Subtask 7.a

Let:

- $x_{ik} \in \{0, 1\}$ - k-th letter of i-th string
- $i \in \{1...N\}$
- $k \in \{1...D\}$

Then the first model arising straight from the definition of Hamming distance would be:

$$\max_{x} \min_{i \neq j} \sum_{k} x_{ik} \operatorname{xor} x_{jk}$$

But it's kind of useless and I doubt it would give me maximum points for this task, so let's try and bring it into a linear programming problem.

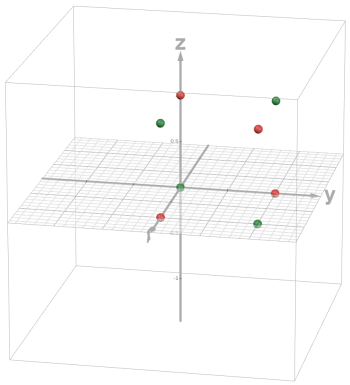
Firstly, we can introduce a variable that will represent the Hamming distance between all strings, to get rid of the min expression in the model. This will become the value that we want to maximize, and we need to make sure it actually is the smallest distance between any pair of strings - which yields such constraints:

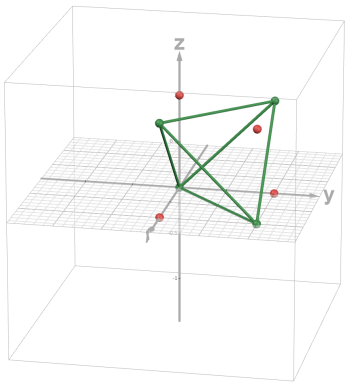
$$\begin{array}{ll} \max & t \\ \forall_{i \neq j} & \sum_{k} x_{ik} \operatorname{xor} x_{jk} \geq t \\ \forall_{ik} & x_{ik} \in \{0, 1\} \\ & t \in \mathbb{Z} \end{array}$$

Now for the xor we will also need to introduce an auxilary variable for each $k, i \neq j$, which can be reduced to $k, i < j$ because Hamming distance is symmetrical, but it's still quite a lot - $\Theta(DN^2)$ additional variables.

But the additional challenge is to constrain each of them to correctly represent the xor operator.

$z = x \operatorname{xor} y$	Formulation of $z = x \operatorname{xor} y$





A correct set of constraints should make the points, marked above as red, infeasible. Unfortunattely as we can see for xor there is no way to "cut away" the infeasible points with less than 4 linear constraints per one xor auxilary variable. So we can use the 4 halfspaces describing the formulation, or - instead - we can use the convex hull description of the formulation, which will introduce more variables but less constraints and additionally will yield the constraints in an equational form which would eliminate the need for any slack variables when solving the problem.

- $u_{ijk} \in \{0, 1\}$ - auxilary variable representing $x_{ik} \operatorname{xor} x_{jk}$

$$(x_{ik}, x_{jk}, u_{ijk}) \in \left\{ \begin{matrix} \mathbb{0} + \alpha_{ijk}(1,1,0) \\ +\beta_{ijk}(1,0,1) \\ +\gamma_{ijk}(0,1,1) \end{matrix} \;\middle|\; \begin{matrix} \alpha_{ijk}, \beta_{ijk}, \gamma_{ijk} \geq 0 \\ \alpha_{ijk} + \beta_{ijk} + \gamma_{ijk} = 1 \end{matrix} \right\}$$

So we have:

$$\begin{aligned} x_{ik} &= \alpha_{ijk} + \beta_{ijk} \\ x_{jk} &= \alpha_{ijk} + \gamma_{ijk} \\ u_{ijk} &= \beta_{ijk} + \gamma_{ijk} \\ \alpha_{ijk} + \beta_{ijk} + \gamma_{ijk} &= 1 \\ \alpha_{ijk}, \beta_{ijk}, \gamma_{ijk} &\geq 0 \end{aligned}$$

Where $\alpha_{ijk} + \beta_{ijk} + \gamma_{ijk}$ "choose" which vertex of the formulation we're in, so they can either be continuous or binary.

We notice that the variable u_{ijk} can now be completely ommited. And we're left with the final model for the problem:

$$
\begin{array}{ll}
\max & t \\
\forall_{i<j} & \sum_k (\beta_{ijk} + \gamma_{ijk}) \geq t \\
\forall_{i<j,k} & x_{ik} = \alpha_{ijk} + \beta_{ijk} \\
\forall_{l<i,k} & x_{ik} = \alpha_{lik} + \gamma_{lik} \\
\forall_{i<j,k} & \alpha_{ijk} + \beta_{ijk} + \gamma_{ijk} = 1 \\
 & t \in \mathbb{Z} \\
\forall_{ik} & x_{ik} \in \{0,1\} \\
\forall_{i<j,k} & \alpha_{ijk}, \beta_{ijk}, \gamma_{ijk} \in \{0,1\}
\end{array}
$$

Subtask 7.b

My model classifies as an Integer Linear Programming Problem, but could also be made into a fully Binary Linear Programming Problem if we (without loss of generality) substituted t with the distance between first two strings (so with $\sum_k(\beta_{12k} + \gamma_{12k})$ assuming of course $N \geq 2$) which would reduce the only non-binary integer variable and could potentially work better with branching.

Task 8

Subtask 8.a

I will be using same notation as in the task:

- $x_j \geq 0$ - fraction of investment put into asset j (decision)
- r_{jt} - historical return of investment j in moths t - $t+1$
- $\hat{R}_j = \frac{1}{T}\sum_t r_{jt}$ - estimated average return of asset j
- $a = \max\{0, \min_j \hat{R}_j\}$ - minimum estimated return over all assets or 0
- $b = \max_j \hat{R}_j$ - maximum estimated return over all assets
- $B = a + \epsilon(b-a)$ - minimum desired reward
- $j \in \{1...N\}$
- $t \in \{1...T\}$

$$\begin{aligned} \min \quad & \sum_t \left[\left| \sum_j x_j (r_{jt} - \hat{R}_j) \right| \right] \\ & \sum_j x_j \hat{R}_j \geq B \\ & \sum_j x_j = 1 \\ \forall_j \quad & x_j \geq 0 \end{aligned}$$

This would be the first, non-linear model that we could write, coming straight from the definitions of risk and reward given in the task and I think it needs no further explanation.

Now the goal is to linearize it.

Since the values $B, \hat{R}_j, r_{jt}$ are given and are not to be decided upon in any way - the above model is almost linear. The only thing that needs to be taken care of is the absolute value in the objective function. But since it's in a minimization term, we can use the classic trick:

- $d_t \geq 0$ - absolute deviation from the expected mean return in months t - $t+1$

$$\forall_t \; d_t \geq \sum_j x_j (r_{jt} - \hat{R}_j)$$
$$\forall_t \; d_t \geq -\sum_j x_j (r_{jt} - \hat{R}_j)$$

Which yields the Linear Programming Model:

$$
\begin{array}{ll}
\min & \sum_t d_t \\
& \sum_j x_j \hat{R}_j \geq B \\
& \sum_j x_j = 1 \\
\forall_t & d_t \geq \sum_j x_j (r_{jt} - \hat{R}_j) \\
\forall_t & d_t \geq -\sum_j x_j (r_{jt} - \hat{R}_j) \\
\forall_j & x_j \geq 0 \\
\forall_t & d_t \geq 0
\end{array}
$$

Subtask 8.b