Testing Edgecases

Infeasible Start

```python
from mip import Model, maximize, xsum
```

```python
m = Model()

x1 = m.add_var(lb = 0) # variables
x2 = m.add_var(lb = 0)

m += x1 + x2 <= 2
m += 2 * x1 + 2 * x2 >= 2 # constraints

m.objective = maximize(x1 - x2)

m.optimize()
```

```
Coin0506I Presolve 2 (0) rows, 2 (0) columns and 4 (0) elements
Clp1000I sum of infeasibilities 0 - average 0, 0 fixed columns
Coin0506I Presolve 2 (0) rows, 2 (0) columns and 4 (0) elements
Clp0006I 0  Obj 10 Dual inf 200 (2)
Clp0029I End of values pass after 2 iterations
Clp0000I Optimal - objective value 10
Clp0000I Optimal - objective value 10
Clp0000I Optimal - objective value 10
Clp0032I Optimal objective 10 - 0 iterations time 0.002, Idiot 0.00
Starting solution of the Linear programming problem using Primal Simplex
```

```
<OptimizationStatus.OPTIMAL: 0>
```

We notice no errors which looks promising.

```
x = [x1.x, x2.x] # get values, not objects
print(f"x: {x}")
```

```
x: [2.0, 0.0]
```

This is a correct solution which brings us to a conclusion that pythonmip correctly handles infeasible start edgecase.

Unboudedness

```
m = Model()

x1 = m.add_var(lb = 0)
x2 = m.add_var(lb = 0)

m += -x1 + x2 <= 1
m += x2 <= 5

m.objective = maximize(2 * x1 + x2)

m.optimize()
```

```
Coin0506I Presolve 0 (-2) rows, 0 (-2) columns and 0 (-4) elements
Clp0000I Optimal - objective value 2
Coin0511I After Postsolve, objective 2, infeasibilities - dual 0 (0), primal 0 (0)
Clp0032I Optimal objective 2 - 0 iterations time 0.002, Presolve 0.00, Idiot 0.00
Starting solution of the Linear programming problem using Primal Simplex

<OptimizationStatus.UNBOUNDED: 2>
```

Pythonmip correctly finds that a problem is unbounded.Let's see if the variables have been changed.

```
x = [x1.x, x2.x]
print(f"x: {x}")
```

```
x: [None, None]
```

Worth noting: variables get set to None whenever pythonmip identifies an unbounded problem.

Infinite Solutions

```python
m = Model()

x1 = m.add_var(lb = 0)
x2 = m.add_var(lb = 0)

m += 5 * x1 + 10 * x2 <= 60
m += 4 * x1 + 4 * x2 <= 40

m.objective = maximize(x1 + x2)
```

We can already see that one of the constraints is a multiple of the objective function, so there's a big chance to have an entire face of solutions. We're interested in seeing which solution the algorithm will give us.

```python
m.optimize()

x = [x1.x, x2.x]
print(f"x: {x}")
```

```
Coin0508I Presolve thinks problem is unbounded
Clp3003W Analysis indicates model infeasible or unbounded
Clp1000I sum of infeasibilities 0 - average 0, 0 fixed columns
Coin0508I Presolve thinks problem is unbounded
Clp0006I 0  Obj 4.9999498 Primal inf 3.9999488 (1) Dual inf 1.01e+12 (2)
Clp0029I End of values pass after 1 iterations
Clp0002I Dual infeasible - objective value 13
Clp0002I Dual infeasible - objective value 13
Clp0002I Dual infeasible - objective value 13
Clp0032I DualInfeasible objective 13 - 0 iterations time 0.002, Idiot 0.00
Starting solution of the Linear programming problem using Primal Simplex

x: [10.0, 0.0]
```

So pythonmip just gives us the first seen solution.

The model in matrix standard form

Let's take the previous problem

$$A = \begin{bmatrix} 5 & 10 \\ 4 & 4 \end{bmatrix}, b = \begin{bmatrix} 60 \\ 40 \end{bmatrix}, c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{array}{rl} max & z = c^T x \\ s.t. & Ax <= b \\ & x >= 0 \end{array}$$

All necessary variables and initial tableau for the problem:

```python
import numpy as np
from util import tableau

A = [
    [5, 10],
    [4, 4]
]
b = [[60], [40]]
c = [[1], [1]]

T = np.concatenate((
    np.concatenate((A, np.identity(2), [[0], [0]], b), axis = 1),
    np.concatenate((np.transpose(c), [[0, 0, 1, 0]]), axis = 1)
    ), axis = 0
)

tableau(T)
```

```
|-------+-------+-------+-------+-------+-------+
|    x1 |    x2 |    x3 |    x4 |    -z |     b |
|-------+-------+-------+-------+-------+-------+
|   5.0 |  10.0 |   1.0 |   0.0 |   0.0 |  60.0 |
|   4.0 |   4.0 |   0.0 |   1.0 |   0.0 |  40.0 |
|-------+-------+-------+-------+-------+-------+
|   1.0 |   1.0 |   0.0 |   0.0 |   1.0 |   0.0 |
|-------+-------+-------+-------+-------+-------+
```

Now another way we can specify the model is:

```
m = Model()

x = [m.add_var(lb = 0) for i in range(2)]

for j in range(2):
    m += xsum(A[j][i] * x[i] for i in range(2)) <= b[j][0]

m.objective = maximize(xsum(c[i][0] * x[i] for i in range(2)))

m.optimize()
```

```
Coin0506I Presolve 2 (0) rows, 2 (0) columns and 4 (0) elements
Clp1000I sum of infeasibilities 0 - average 0, 0 fixed columns
Coin0506I Presolve 2 (0) rows, 2 (0) columns and 4 (0) elements
Clp0006I 0  Obj 10 Dual inf 200 (2)
Clp0029I End of values pass after 2 iterations
Clp0000I Optimal - objective value 10
Clp0000I Optimal - objective value 10
Clp0000I Optimal - objective value 10
Clp0032I Optimal objective 10 - 0 iterations time 0.002, Idiot 0.00
Starting solution of the Linear programming problem using Primal Simplex
```

```
<OptimizationStatus.OPTIMAL: 0>
```

Which is a bit more open-close if we have the problem in standard form. \

We get again:

$$x = [10.0, 0.0]$$