OUTPUT:
Number of pivot steps
Objective Value
Optimal Solution

## 1. Porządek leksykograficzny, minimum:

LEXICOGRAPHICAL MIN – jako zmienną wchodzącą i wychodzącą wybieramy te zmienne, które są najmniejsze względem porządku leksykograficznego

```
def lexicographical_min_entering(self):
  return min(self.possible_entering())

def lexicographical_min_leaving(self):
  return min(self.possible_leaving())
```

```
I. American Steel Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 5
-150050000.0
(3000.0, 2000.0, 3000.0, 4000.0, 3000.0, 3000.0, 2000.0, 0.0, 3000.0, 2000.0,
3000.0, 1000.0, 2000.0, 4000.0, 2000.0)

II. Beer Distribution Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 2
-86000000
(0, 700, 200, 900, 0, 0, 0, 300, 200, 1800, 0)

III. Computer Plant Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 7
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700,
1000)

IV. Furniture Manufacturing Problem
Number of pivot steps: 2
32000000.0
(8.0, 16.0)

V. Problem: GAMSMOD
The initial dictionary is infeasible, solving auxiliary problem.
The original problem is infeasible.
Number of pivot steps: 0
0
```

None

VI. Sponge Roll Problem

VII. The Whiskas Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 7
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

VIII. The Whiskas Problem2
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 11
-4800.0
(0.0, 0.0, 0.0, 0.0, 60.0, 0.0)

IX. Optimized Diet Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 0
-705.109100585
(0.03826503459, 0.2948908994, 0.0, 0.0095263438)

X. HR Problem
Number of pivot steps: 5
2770000
(0, 0, 0, 1, 1, 0, 0, 1, 0, 0)

XI. Paper Rolls problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 2
-4522500
(0, 97, 0, 0, 0, 0, 0, 0, 0, 395/2, 0, 0, 631/4)

**2. Porządek leksykograficzny, maksimum:**

LEXICOGRAPHICAL MAX – jako zmienną wchodzącą i wychodzącą wybieramy te zmienne, które są największe względem porządku leksykograficznego

```
def lexicographical_max_entering(self):
  return max(self.possible_entering())

def lexicographical_max_leaving(self):
  return max(self.possible_leaving())
```

I. American Steel Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  3
-150050000.0
(3000.0, 2000.0, 3000.0, 4000.0, 3000.0, 3000.0, 2000.0, 0.0, 3000.0, 2000.0, 3000.0, 1000.0, 2000.0, 4000.0, 2000.0)

II. Beer Distribution Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 2
-86000000
(0, 700, 200, 900, 0, 0, 0, 300, 200, 1800, 0)

III. Computer Plant Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 7
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

IV. Furniture Manufacturing Problem
Number of pivot steps: 2
32000000.0
(8.0, 16.0)

V. Problem: GAMSMOD
The initial dictionary is infeasible, solving auxiliary problem.
The original problem is infeasible.
Number of pivot steps: 0
0
None

VI. Sponge Roll Problem

VII. The Whiskas Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 7
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

VIII. The Whiskas Problem2
The initial dictionary is infeasible, solving auxiliary problem.

```
Back to the original problem.
Number of pivot steps: 11
-4800.0
(0.0, 0.0, 0.0, 0.0, 60.0, 0.0)
```

IX. Optimized Diet Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 0
-705.109100585
(0.03826503459, 0.2948908994, 0.0, 0.0095263438)
```

X. HR Problem

```
Number of pivot steps: 5
2770000
(0, 0, 0, 1, 1, 0, 0, 1, 0, 0)
```

XI. Paper Rolls problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 2
-4522500
(0, 97, 0, 0, 0, 0, 0, 0, 0, 395/2, 0, 0, 631/4)
```

**3. Wybór zmiennej wejściowej o największym współczynniku funkcji celu**

LARGEST COEFFICIENT – wybieramy zmienną o największym współczynniku funkcji celu. Zasada ta maksymalizuje wzrost funkcji celu.

```
def get_objective_coefficient(self, variable):
  for i in range(0, len(self.nonbasic_variables())):
    if self.nonbasic_variables()[i] == variable:
      return self.objective_coefficients()[i]

def largest_coefficient_entering(self):
  cand = self.possible_entering()[0]
  for c in self.possible_entering():
   if get_objective_coefficient(self, cand)<get_objective_coefficient(self, c):
     cand = c
  return cand

def largest_coefficient_leaving(self):
  return max(self.possible_leaving())
```

I. American Steel Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  3
-150050000.0
```

(3000.0, 2000.0, 3000.0, 4000.0, 3000.0, 3000.0, 2000.0, 0.0, 3000.0, 2000.0, 3000.0, 1000.0, 2000.0, 4000.0, 2000.0)

II. Beer Distribution Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 2
-86000000
(0, 700, 200, 900, 0, 0, 0, 300, 200, 1800, 0)

III. Computer Plant Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 8
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

IV. Furniture Manufacturing Problem
Number of pivot steps: 2
32000000.0
(8.0, 16.0)

V. Problem: GAMSMOD
The initial dictionary is infeasible, solving auxiliary problem.
The original problem is infeasible.
Number of pivot steps: 0
0
None

VI. Sponge Roll Problem

VII. The Whiskas Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 8
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

VIII. The Whiskas Problem2
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 9
-4800.0
(0.0, 0.0, 0.0, 0.0, 60.0, 0.0)

IX. Optimized Diet Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 0
-705.109100585
(0.03826503459, 0.2948908994, 0.0, 0.0095263438)
```

X. HR Problem

```
Number of pivot steps: 12
2770000
(0, 0, 0, 1, 1, 0, 0, 1, 0, 0)
```

XI. Paper Rolls problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 2
-4522500
(0, 97, 0, 0, 0, 0, 0, 0, 0, 395/2, 0, 0, 631/4)
```

**4. Wybór zmiennej wejściowej o najmniejszym współczynniku funkcji celu**

LEAST COEFFICIENT – wybieramy zmienną, która ma najmniejszy współczynnik w funkcji celu

```
def get_objective_coefficient(self, variable):
  for i in range(0, len(self.nonbasic_variables())):
   if self.nonbasic_variables()[i] == variable:
    return self.objective_coefficients()[i]

def least_coefficient_entering(self):
  cand = self.possible_entering()[0]
  for c in self.possible_entering():
   if get_objective_coefficient(self, cand)>get_objective_coefficient(self, c):
    cand = c
  return cand

def least_coefficient_leaving(self):
  return min(self.possible_leaving())
```

I. American Steel Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  5
-150050000.0
(3000.0, 2000.0, 3000.0, 4000.0, 3000.0, 3000.0, 2000.0, 0.0, 3000.0, 2000.0,
3000.0, 1000.0, 2000.0, 4000.0, 2000.0)
```

II. Beer Distribution Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
```

Number of pivot steps: 4
-86000000
(0, 700, 200, 900, 0, 0, 0, 300, 200, 1800, 0)

III. Computer Plant Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 6
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

IV. Furniture Manufacturing Problem
Number of pivot steps: 2
32000000.0
(8.0, 16.0)

V. Problem: GAMSMOD
The initial dictionary is infeasible, solving auxiliary problem.
The original problem is infeasible.
Number of pivot steps: 0
0
None

VI. Sponge Roll Problem

VII. The Whiskas Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 6
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

VIII. The Whiskas Problem 2
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 5
-4800.0
(0.0, 0.0, 0.0, 0.0, 60.0, 0.0)

IX. Optimized Diet Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 0
-705.109100585
(0.03826503459, 0.2948908994, 0.0, 0.0095263438)

X. HR Problem

```
Number of pivot steps: 14
2770000
(0, 0, 0, 1, 1, 0, 0, 1, 0, 0)
```

XI. Paper Rolls problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 3
-4522500
(0, 97, 0, 0, 0, 0, 0, 0, 0, 395/2, 0, 0, 631/4)
```

**5. Wybór zmiennej, który prowadzi do największego wzrostu funkcji celu**

LARGEST INCREASE - wybieramy tą parę zmiennych wchodzących i wychodzących, która wpływa na maksymalny przyrost funkcji celu. Przeprowadzenie tego wyboru jest stosunkowo bardziej kosztowne niż inne reguły, gdyż wymaga przejrzenia potencjalnie wielu kombinacji zmiennych, ale gwarantuje najlepsze zachowanie algorytmu w skali lokalnej.

```
def all_combinations(self):
     result = []
     for ent in self.possible_entering():
          self1 = deepcopy(self)
          self1.enter(ent)
          for lev in self1.possible_leaving():
               self2 = deepcopy(self1)
               self2.leave(lev)
               self2.update()
               result.append([self2.objective_value(), ent, lev])
     return result

def max_increase_entering(self):
     all_comb = all_combinations(self)
     best = 0
     for i in range(0, len(all_comb)):
          if all_comb[best][0] < all_comb[i][0]:
               best = i
     return all_comb[best][1]

def max_increase_leaving(self):
     all_comb = all_combinations(self)
     best = 0
     for i in range(0, len(all_comb)):
          if all_comb[best][0] < all_comb[i][0]:
               best = i
     return all_comb[best][2]
```

I. American Steel Problem

The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  5
-150050000.0
(3000.0, 2000.0, 3000.0, 4000.0, 3000.0, 3000.0, 2000.0, 0.0, 3000.0, 2000.0, 3000.0, 1000.0, 2000.0, 4000.0, 2000.0)

II. Beer Distribution Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 2
-86000000
(0, 700, 200, 900, 0, 0, 0, 300, 200, 1800, 0)

III. Computer Plant Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 9
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

IV. Furniture Manufacturing Problem
Number of pivot steps: 2
32000000.0
(8.0, 16.0)

V. Problem: GAMSMOD
The initial dictionary is infeasible, solving auxiliary problem.
The original problem is infeasible.
Number of pivot steps: 0
0
None

VI. Sponge Roll Problem

VII. The Whiskas Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 9
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

VIII. The Whiskas Problem 2
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 3

```
-4800.0
(0.0, 0.0, 0.0, 0.0, 60.0, 0.0)
```

IX. Optimized Diet Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 0
-705.109100585
(0.03826503459, 0.2948908994, 0.0, 0.0095263438)
```

X. HR Problem

```
Number of pivot steps: 5
2770000
(0, 0, 0, 1, 1, 0, 0, 1, 0, 0)
```

XI. Paper Rolls problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 2
-4522500
(0, 97, 0, 0, 0, 0, 0, 0, 0, 395/2, 0, 0, 631/4)
```

**6. Wybór zmiennej, który prowadzi do najmniejszego wzrostu funckji celu**

LOWEST INCREASE - wybieramy tą pare zmiennych wchodzących i wychodzących, która wpływa na minimalny przyrost funkcji celu. Przeprowadzenie tego wyboru jest stosunkowo bardziej kosztowne niż inne reguły, gdyż wymaga przejrzenia potencjalnie wielu kombinacji zmiennych, ale gwarantuje najlepsze zachowanie algorytmu w skali lokalnej.

```
def all_combinations(self):
      result = []
      for ent in self.possible_entering():
            self1 = deepcopy(self)
            self1.enter(ent)
            for lev in self1.possible_leaving():
                  self2 = deepcopy(self1)
                  self2.leave(lev)
                  self2.update()
                  result.append([self2.objective_value(), ent, lev])
      return result

def min_increase_entering(self):
      all_comb = all_combinations(self)
      best = 0
      for i in range(0, len(all_comb)):
            if all_comb[best][0] > all_comb[i][0]:
                  best = i
      return all_comb[best][1]
```

```
def min_increase_leaving(self):
      all_comb = all_combinations(self)
      best = 0
      for i in range(0, len(all_comb)):
            if all_comb[best][0] > all_comb[i][0]:
                  best = i
      return all_comb[best][2]
```

I. American Steel Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  5
-150050000.0
(3000.0, 2000.0, 3000.0, 4000.0, 3000.0, 3000.0, 2000.0, 0.0, 3000.0, 2000.0,
3000.0, 1000.0, 2000.0, 4000.0, 2000.0)
```

II. Beer Distribution Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 4
-86000000
(0, 700, 200, 900, 0, 0, 0, 300, 200, 1800, 0)
```

III. Computer Plant Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 8
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700,
1000)
```

IV. Furniture Manufacturing Problem

```
Number of pivot steps: 2
32000000.0
(8.0, 16.0)
```

V. Problem: GAMSMOD

```
The initial dictionary is infeasible, solving auxiliary problem.
The original problem is infeasible.
Number of pivot steps: 0
0
None
```

VI. Sponge Roll Problem

VII. The Whiskas Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
```

```
Back to the original problem.
Number of pivot steps: 8
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700,
1000)
```

VIII. The Whiskas Problem 2

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 11
-4800.0
(0.0, 0.0, 0.0, 0.0, 60.0, 0.0)
```

IX. Optimized Diet Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 0
-705.109100585
(0.03826503459, 0.2948908994, 0.0, 0.0095263438)
```

X. HR Problem

```
Number of pivot steps: 18
2770000
(0, 0, 0, 1, 1, 0, 0, 1, 0, 0)
```

XI. Paper Rolls problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 2
-4522500
(0, 97, 0, 0, 0, 0, 0, 0, 0, 395/2, 0, 0, 631/4)
```

## 7. Wybór zmiennej, który prowadzi do wierzchołka w kierunku najbliższym wektorowi c (gradientowi funkcji celu)

STEEPEST EDGE MAX - wybieramy zmienną, która prowadzi do wierzchołka w kierunku najbliższym wektorowi c, czyli gradientowi funkcji celu. Tak więc maksymalizujemy stosunek

$$\frac{c^T(x1 - x2)}{||x1 - x2||}$$

gdzie x2 jest podstawowym wykonalnym rozwiązaniem dla obecnej tabeli sympleksowej. Natomiast x1 jest to rozwiązanie dla tabeli, które zostanie uzyskane poprzez wprowadzanie zmiennej do podstaw problemu.

```
def st_edge_values(self):
  values = {}
  for v1 in self.possible_entering():
   x1 = deepcopy(self)
```

```
    x1.enter(v1)
    for v2 in x1.possible_leaving():
     x2 = deepcopy(x1)
     x2.leave(v2)
     x2.update()
     y = x2.basic_solution()-self.basic_solution()
     values[(v1, v2)] = np.dot(self.objective_coefficients(),
                                                          y/np.linalg.norm(y))
  return values

def st_edge_max_entering(self):
  val = st_edge_values(self)
  return max(val.iteritems(), key=operator.itemgetter(1))[0][0]

def st_edge_max_leaving(self):
  val = st_edge_values(self)
  return max(val.iteritems(), key=operator.itemgetter(1))[0][1]
```

I. American Steel Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  4
-150050000.0
(3000.0, 2000.0, 3000.0, 4000.0, 3000.0, 3000.0, 2000.0, 0.0, 3000.0, 2000.0,
3000.0, 1000.0, 2000.0, 4000.0, 2000.0)
```

II. Beer Distribution Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  2
-86000000
(0, 700, 200, 900, 0, 0, 0, 300, 200, 1800, 0)
```

III. Computer Plant Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  7
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700,
1000)
```

IV. Furniture Manufacturing Problem

```
Number of pivot steps: 2
32000000.0
(8.0, 16.0)
```

V. Problem: GAMSMOD

```
The initial dictionary is infeasible, solving auxiliary problem.
The original problem is infeasible.
```

Number of pivot steps: 0
0
None

VI. Sponge Roll Problem

VII. The Whiskas Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 7
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

VIII. The Whiskas Problem 2
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 7
-4800.0
(0.0, 0.0, 0.0, 0.0, 60.0, 0.0)

IX. Optimized Diet Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 0
-705.109100585
(0.03826503459, 0.2948908994, 0.0, 0.0095263438)

X. HR Problem
Number of pivot steps: 9
2770000
(0, 0, 0, 1, 1, 0, 0, 1, 0, 0)

XI. Paper Rolls problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 2
-4522500
(0, 97, 0, 0, 0, 0, 0, 0, 0, 395/2, 0, 0, 631/4)

**8. Wybór zmiennej, który prowadzi do wierzchołka w kierunku najdalszym od wektora c (gradientu funkcji celu)**

STEEPEST EDGE MIN - wybieramy zmienną, która prowadzi do wierzchołka w kierunku najdalszym wektorowi c, czyli gradientowi funkcji celu. Tak więc minimalizujemy stosunek

$$\frac{c^T(x1 - x2)}{||x1 - x2||}$$

gdzie x2 jest podstawowym wykonalnym rozwiązaniem dla obecnej tabeli sympleksowej. Natomiast x1 jest to rozwiązanie dla tabeli, które zostanie uzyskane poprzez wprowadzanie zmiennej do podstaw problemu.

```
def st_edge_values(self):
  values = {}
  for v1 in self.possible_entering():
   x1 = deepcopy(self)
   x1.enter(v1)
   for v2 in x1.possible_leaving():
    x2 = deepcopy(x1)
    x2.leave(v2)
    x2.update()
    y = x2.basic_solution()-self.basic_solution()
    values[(v1, v2)] = np.dot(self.objective_coefficients(),
                                                    y/np.linalg.norm(y))
  return values

def st_edge_min_entering(self):
  val = st_edge_values(self)
  return min(val.iteritems(), key=operator.itemgetter(1))[0][0]

def st_edge_min_leaving(self):
  val = st_edge_values(self)
  return min(val.iteritems(), key=operator.itemgetter(1))[0][1]
```

I. American Steel Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  4
-150050000.0
(3000.0, 2000.0, 3000.0, 4000.0, 3000.0, 3000.0, 2000.0, 0.0, 3000.0, 2000.0,
3000.0, 1000.0, 2000.0, 4000.0, 2000.0)
```

II. Beer Distribution Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  4
-86000000
(0, 700, 200, 900, 0, 0, 0, 300, 200, 1800, 0)
```

III. Computer Plant Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  8
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700,
1000)
```

IV. Furniture Manufacturing Problem
Number of pivot steps:  2
32000000.0
(8.0, 16.0)

V. Problem: GAMSMOD
The initial dictionary is infeasible, solving auxiliary problem.
The original problem is infeasible.
Number of pivot steps: 0
0
None

VI. Sponge Roll Problem

VII. The Whiskas Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  8
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

VIII. The Whiskas Problem 2
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  5
-4800.0
(0.0, 0.0, 0.0, 0.0, 60.0, 0.0)

IX. Optimized Diet Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 0
-705.109100585
(0.03826503459, 0.2948908994, 0.0, 0.0095263438)

X. HR Problem
Number of pivot steps: 14
2770000
(0, 0, 0, 1, 1, 0, 0, 1, 0, 0)

XI. Paper Rolls problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps: 4
-4522500
(0, 97, 0, 0, 0, 0, 0, 0, 0, 395/2, 0, 0, 631/4)

**9. Wybór zmiennej wchodzącej o najmniejszym indeksie; jeżeli jest wiele wyborów zmiennej wychodzącej, to wybór zmiennej wychodzącej o najmniejszym indeksie**

BLAND RULE - wybieramy zmienną o najmniejszym indeksie. Jeśli istnieje kilka możliwości na zmienną wychodzącą, także wybieramy tę o najmniejszym indeksie.

```
def blandd_rule_entering(self):
     return min(self.possible_entering())

def blandd_rule_leaving(self):
     return min(self.possible_leaving())
```

I. American Steel Problem
```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  5
-150050000.0
(3000.0, 2000.0, 3000.0, 4000.0, 3000.0, 3000.0, 2000.0, 0.0, 3000.0, 2000.0,
3000.0, 1000.0, 2000.0, 4000.0, 2000.0)
```

II. Beer Distribution Problem
```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  2
-86000000
(0, 700, 200, 900, 0, 0, 0, 300, 200, 1800, 0)
```

III. Computer Plant Problem
```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  7
-2178000000
```

IV. Furniture Manufacturing Problem
```
Number of pivot steps:  2
32000000.0
(8.0, 16.0)
```

V. Problem: GAMSMOD
```
The initial dictionary is infeasible, solving auxiliary problem.
The original problem is infeasible.
Number of pivot steps:  0
0
None
```

VI. Sponge Roll Problem

VII. The Whiskas Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  7
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700,
1000)
```

VIII. The Whiskas Problem 2

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  11
-4800.0
(0.0, 0.0, 0.0, 0.0, 60.0, 0.0)
```

IX. Optimized Diet Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  0
-705.109100585
(0.03826503459, 0.2948908994, 0.0, 0.0095263438)
```

X. HR Problem

```
Number of pivot steps:  5
2770000
(0, 0, 0, 1, 1, 0, 0, 1, 0, 0)
```

XI. Paper Rolls problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  2
-4522500
(0, 97, 0, 0, 0, 0, 0, 0, 0, 395/2, 0, 0, 631/4)
```

## 10. Wybór losowy (prawdopodobieństwo jednostajne)

RANDOM RULE - zarówno zmienną wchodzącą jak i wychodzącą wybieramy w sposób losowy. Oba te losowania są od siebie niezależne.

```
def random_edge_entering(self):
     return random.choice(self.possible_entering())

def random_edge_leaving(self):
     return random.choice(self.possible_leaving())
```

I. American Steel Problem

```
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  3
```

-150050000.0
(3000.0, 2000.0, 3000.0, 4000.0, 3000.0, 3000.0, 2000.0, 0.0, 3000.0, 2000.0, 3000.0, 1000.0, 2000.0, 4000.0, 2000.0)

II. Beer Distribution Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  2
-86000000
(0, 700, 200, 900, 0, 0, 0, 300, 200, 1800, 0)

III. Computer Plant Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  6
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

IV. Furniture Manufacturing Problem
Number of pivot steps:  2
32000000.0
(8.0, 16.0)

V. Problem: GAMSMOD
The initial dictionary is infeasible, solving auxiliary problem.
The original problem is infeasible.
Number of pivot steps:  0
0
None

VI. Sponge Roll Problem

VII. The Whiskas Problem
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  8
-2178000000
(0, 0, 0, 0, 27/20, 1500, 0, 0, 0, 0, 0, 0, 1200, 0, 0, 0, 0, 0, 27/20, 1700, 1000)

VIII. The Whiskas Problem 2
The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  2
-4800.0
(0.0, 0.0, 0.0, 0.0, 60.0, 0.0)

IX. Optimized Diet Problem

The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  0
-705.109100585
(0.03826503459, 0.2948908994, 0.0, 0.0095263438)

X. HR Problem

Number of pivot steps:  13
2770000
(0, 0, 0, 1, 1, 0, 0, 1, 0, 0)

XI. Paper Rolls Problem

The initial dictionary is infeasible, solving auxiliary problem.
Back to the original problem.
Number of pivot steps:  2
-4522500
(0, 97, 0, 0, 0, 0, 0, 0, 0, 395/2, 0, 0, 631/4)

| | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. |
|---|---|---|---|---|---|---|---|---|---|---|
| I. | 5 | 3 | 3 | 5 | 5 | 5 | 4 | 4 | 5 | 3 |
| II. | 2 | 2 | 2 | 4 | 2 | 4 | 2 | 4 | 2 | 2 |
| III. | 7 | 7 | 8 | 6 | 9 | 8 | 7 | 8 | 7 | 6 |
| IV. | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| V. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| VI. | - | - | - | - | - | - | - | - | - | - |
| VII. | 7 | 7 | 8 | 6 | 9 | 8 | 7 | 8 | 7 | 8 |
| VIII. | 11 | 11 | 9 | 5 | 3 | 11 | 7 | 5 | 11 | 2 |
| IX. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| X. | 5 | 5 | 12 | 14 | 5 | 18 | 9 | 14 | 5 | 13 |
| XI. | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 4 | 2 | 2 |

STEPPEST EDGE jest najlepszą metodą wyboru zmiennych w metodzie sympleks. Rownie dobrą metodą jest RANDOM EDGE. Pozwala ona na możliwie najlepsze określenie granicy liczby kroków metody sympleks.