# AA_LAB_12_Assignment

# CE_054

**Aim :-** Implementation the Simplex Tabular Method of Linear Programming.

1. Simplex Tabular Method :-

Code :-

```
#Author : Dhruv B Kakadiya

import itertools
import collections
import numpy as np

class simplex_tabular_solver:

    def __init__(self, eq1, eq2, max_eq):
        self.eq1 = eq1
        self.eq1.insert(len(self.eq1) - 1, 0)
        self.eq1.insert(len(self.eq1) - 1, 1)
        self.eq2 = eq2
        self.eq2.insert(len(self.eq2) - 1, 1)
        self.eq2.insert(len(self.eq2) - 1, 0)
        self.max_eq = max_eq + [0, 0]
        self.Cb = [0, 0]
        self.Xb = [len(self.eq1) - 3, len(self.eq1) - 2]
        self.ratio = [1, 1]
        self.Z = [0] * 5
        self.ci_sub_zi = [0] * (len(self.eq1) - 1)
        self.ci_zi()

    def ci_zi(self):
        self.Z = [(self.eq1[i] * self.Cb[0] + self.eq2[i] * self.Cb[1]) for i
in range(len(self.eq1))]
        self.ci_sub_zi = [(self.max_eq[i] - self.Z[i]) for i in range(len(self
.Z) - 1)]

    def next_turn(self):
        if max(self.ci_sub_zi) > 0:
            index_enter = self.ci_sub_zi.index(max(self.ci_sub_zi))
        else:
            return (True)

        self.ratio[0] = (self.eq1[len(self.eq1)-
1]/self.eq1[index_enter]) if self.eq1[index_enter] > 0 else -1
        self.ratio[1] = (self.eq2[len(self.eq2)-
1]/self.eq2[index_enter]) if self.eq2[index_enter] > 0 else -1
        min_ratio = min(self.ratio)
```

```
        if(min_ratio > 0):
            index_exit = self.ratio.index(min_ratio)
        elif(max(self.ratio) > 0):
            index_exit = self.ratio.index(max(self.ratio))
        else:
            return True

        print("\nEntering var : x%d:= %d, Exiting var : x%d:= %d " % (index_en
ter, self.max_eq[index_enter], self.Xb[index_exit], self.Cb[index_exit]))
        self.Xb[index_exit] = index_enter
        self.Cb[index_exit] = self.max_eq[index_enter]

        if(index_exit == 0):
            self.eq1 = [(self.eq1[i]/self.eq1[index_enter]) for i in range(len
(self.eq1))]
            row_key = self.eq2[index_enter]
            self.eq2 = [(self.eq2[i] - row_key*self.eq1[i]) for i in range(len
(self.eq2))]
            self.ci_zi()
        else:
            self.eq2 = [(self.eq2[i]/self.eq2[index_enter])
                        for i in range(len(self.eq2))]
            row_key = self.eq1[index_enter]
            self.eq1 = [(self.eq1[i] - row_key*self.eq2[i])
                        for i in range(len(self.eq1))]
            self.ci_zi()
        return False

    def table_printing(self):
        print("Cb\t", "Xb\t", "previous ratios:\t\t", self.max_eq, ", rhs]")
        print(self.Cb[0], "\t", self.Xb[0], "\t", self.ratio[0], "\t\t\t\t", s
elf.eq1)
        print(self.Cb[1], "\t", self.Xb[1], "\t", self.ratio[1], "\t\t\t\t", s
elf.eq2)
        print("-\t", "-\tcj - zj\t\t\t\t", self.ci_sub_zi)
        print("\n")
        return ""

    def fully_solved(self):
        finish = False
        while not finish:
            finish = self.next_turn()
            self.table_printing()
        print("After Solving the Value of function is :- ", self.Z[len(self.Z)
 - 1])
        print("using x%d : = %.1f, x%d : = %.1f " %(self.Xb[0], self.eq1[-
1], self.Xb[1], self.eq2[-1]))
```

```
print("\nEquation in format of x1 x2 c1\n")
expression1 = list(map(int, input("Enter the equation - 1 :- ").split()))
expression2 = list(map(int, input("Enter the equation - 2 :- ").split()))
maxEquation = list(map(int, input("Maximization the following function :- ").s
plit()))

ans_partially = simplex_tabular_solver(expression1, expression2, maxEquation)
ans_partially.fully_solved()
```

Output :-

Maximize => 10x1 + 9x2

Eq1 => 3x1 + 3x2 <= 21

Eq2 => 4x1 + 3x2 <= 24

Where x1, x2 >= 0

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

D:\clg2021\AA\LAb12>python Simplex.py

Equation in format of x1 x2 c1

Enter the equation - 1 :- 3 3 21
Enter the equation - 2 :- 4 3 24
Maximization the following function :- 10 9

Entering var : x0:= 10, Exiting var : x3:= 0
Cb      Xb      previous ratios:            [10, 9, 0, 0] , rhs]
0       2       7.0                         [0.0, 0.75, -0.75, 1.0, 3.0]
10      0       6.0                         [1.0, 0.75, 0.25, 0.0, 6.0]
-       -      cj - zj                      [0.0, 1.5, -2.5, 0.0]



Entering var : x1:= 9, Exiting var : x2:= 0
Cb      Xb      previous ratios:            [10, 9, 0, 0] , rhs]
9       1       4.0                         [0.0, 1.0, -1.0, 1.3333333333333333, 4.0]
10      0       8.0                         [1.0, 0.0, 1.0, -1.0, 3.0]
-       -      cj - zj                      [0.0, 0.0, -1.0, -2.0]


Cb      Xb      previous ratios:            [10, 9, 0, 0] , rhs]
9       1       4.0                         [0.0, 1.0, -1.0, 1.3333333333333333, 4.0]
10      0       8.0                         [1.0, 0.0, 1.0, -1.0, 3.0]
-       -      cj - zj                      [0.0, 0.0, -1.0, -2.0]


After Solving the Value of function is :-  66.0
using x1 : = 4.0, x0 : = 3.0

D:\clg2021\AA\LAb12>
```

Maximize => 6x1 + 5x2

Eq1 => x1 + x2 <= 5

Eq2 => 3x1 + 3x2 <= 12

Where x1, x2 >= 0