ASSIGNMENT-4

Annam Deddeepya Purnima (cs22btech11006)
Mane Pooja Vinod (cs22btech11035)
Surbhi (cs22btech11057)

Question:

Assumption
1. Rak of A is n

Implement the simplex algorithm to maximize the objective function, You need to implement the method discussed in class.

Input: CSV file with m+1 rows and n+1 column.
The first row excluding the last element is the cost vector c of length n
The last column excluding the top element is the constraint vector b of length m
Rows two to m+1 and column one to n is the matrix A of size m*n

Output: You need to print the sequence of vertices visited and the value of the objective function at that vertex

REPORT

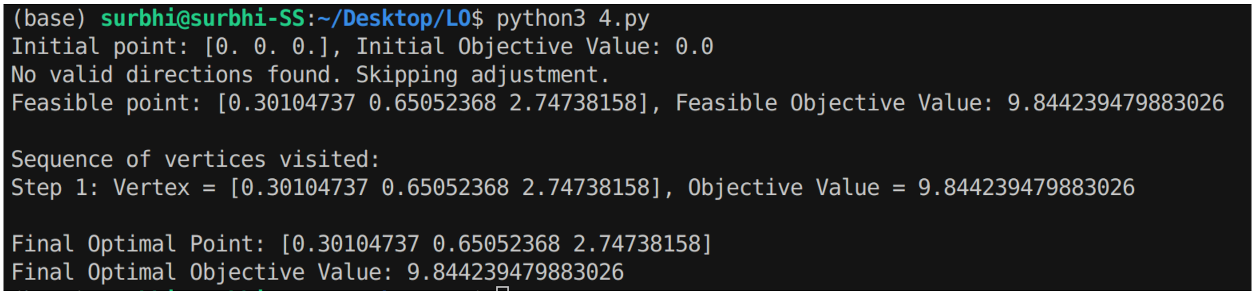
Code implements the **Simplex algorithm** for solving linear programming problems. It finds the optimal solution to a linear programming problem by iterating over feasible solutions while improving the objective function. The code is designed to handle degeneracy and ensure that the algorithm doesn't get stuck in non-optimal solutions.

- ➔ Parsing the input file to extract:
  - Constraint bounds **(b)**
  - Cost vector **(c)**
  - Constraints matrix **(A)**
- ➔ The **is_feasible_point(A, b, z, n)** function checks if the given point z satisfies all the constraints. This is done by computing **A * z** and ensuring that it is less than or equal to b for all constraints.
- ➔ The **adjust_to_basic_feasible** function adjusts the initial point z to a feasible point by ensuring that it satisfies the constraints. It does so by finding a direction

(possibly using the null space) and stepping in that direction until a feasible point is found.

- **handle_degeneracy(c, inverse_active, A, b, z, computed_b, tight)** functions addresses degeneracy, which occurs when there are multiple optimal solutions or the algorithm fails to make progress. It computes reduced costs for each variable and checks if any direction can improve the objective function. If no valid direction is found, the optimization is terminated.
- The **simplex(c, A, b, z, n)** function iterates over feasible points, searching for an optimal solution by moving from one vertex (feasible point) to another, improving the objective value at each step. It stops when no further improvements can be made (i.e., when no valid directions can be found). Tracks the sequence of visited vertices and their associated objective function values.
- Uses pseudoinverses **(np.linalg.pinv)** to handle active constraint matrices could be computationally expensive for large problems, especially with high-dimensional data.
- We took **input as test4.csv at line 132**, we can change according to our test cases.

OUTPUT:



```
(base) surbhi@surbhi-SS:~/Desktop/LO$ python3 4.py
Initial point: [0. 0. 0.], Initial Objective Value: 0.0
No valid directions found. Skipping adjustment.
Feasible point: [0.30104737 0.65052368 2.74738158], Feasible Objective Value: 9.844239479883026

Sequence of vertices visited:
Step 1: Vertex = [0.30104737 0.65052368 2.74738158], Objective Value = 9.844239479883026

Final Optimal Point: [0.30104737 0.65052368 2.74738158]
Final Optimal Objective Value: 9.844239479883026
```