# Assignment 2

**Objective:**

This assignment aims to familiarize students with the Gomory Cut method as an essential algorithm for solving integer linear programming (ILP) problems. Through this assignment, students will apply the Gomory Cut method to a series of ILP problems, understand how to identify fractional solutions and learn to iteratively refine these solutions to meet integer constraints.

**Instructions:**

- Given a test case in "input_ilp.txt", read the file to extract matrices A, b, and c, along with the objective (maximize or minimize) and constraint types. All the variables given in the test cases will be non-negative.
- Ensure the ILP problems are in standard form. If not, convert them accordingly.
- Solve the ILP problems using the Gomory Cut method, illustrating all steps:
    - Identify the initial feasible solution using the simplex method.
    - Apply the Gomory Cut method to iteratively add cuts and eliminate fractional variable solutions.
    - Continue the process until an optimal integer solution is reached.
- For computational analysis, inputs for A, b, and c matrices will be taken from a text file (input_ilp.txt) following a specified format.
- Students are required to submit a Python file named EntryNumber1_EntryNumber2_EntryNumber3_EntryNumber4_A2.py.

**Input Format:**

Your A, b, and c matrices, along with the constraint types and objective ('maximize' or 'minimize'), should be taken as input from a text file (input_ilp.txt) following the specified format. An example for the same has been provided.

- The [objective] section should clearly state whether the problem is a "maximize" or "minimize" problem. This will dictate the direction of your optimization in solving the problem.
- The [A] section contains the coefficients of the variables in the linear constraints. Each row in this section represents a constraint, and each column corresponds to a variable.
- The [b] section lists the right-hand side (RHS) values for each constraint, with each row corresponding to a constraint in the [A] section.
- The [constraint_types] section specifies the type of each constraint corresponding to the rows in [A] and [b]. Use <= for less than or equal to, >= for greater than or equal to, and = for equality constraints.
- The [c] section contains the coefficients of the objective function's variables in one row.

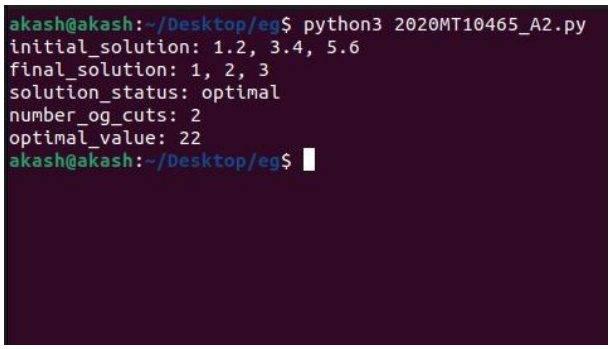
**Submission:**

Submit a Python file as EntryNumber1_EntryNumber2_EntryNumber3_EntryNumber4_A2.py containing a function named “gomory_cut_algo” which reads “input_ilp.txt” and does not take any argument as parameter and print the following information in the given order on command line:

- Initial Solution (initial_solution): The initial feasible solution obtained by the simplex method.
- Final Solution (final_solution): The optimal integer solution after applying Gomory cuts.
- Solution Status (solution_status): “optimal”, “infeasible”, or “unbounded”.
- Number of Cuts (number_of_cuts): The total number of Gomory cuts applied to reach the final solution.
- Optimal Value (optimal_value): The value of the objective function at the optimal solution.

Students may use NumPy or any other library for matrix operations. The deadline for submission is 30th March, 2024 EOD.

Example: If there are 3 variables whose initial optimal value via simplex is 1.2, 3.4, and 5.6, final is 1, 2, and 3 and this is optimal with 2 Gomory cut and 22 as optimal value, then the output of command line must look like this:




Since an auto-grader will be used to evaluate this assignment, be very precise with the output format as any additional information might lead to a mismatch and deduction of marks.

**Grading:**

Grading will be done out of 8.5 marks. Since this is an easy assignment, the marking scheme is divided in 3 parts:

**Part 1: correctness of code**
This section holds 7.5 marks evaluated based on test cases like assignment 1

**Part 2: early bird (bonus)**
Since this assignment is just an easy extension of Assignment 1, as a result A2 carries marks for submission. The early you submit; more marks u get. Suppose you are the first one to submit, you will get 0.5 marks.
After each submission the marks will decay to $0.5e^{-0.05n}$, where n is the number of submissions till now.

In case any group tries to act extra smartly and submit more than ones to make this decay fast, that would lead to a straight 0 for this section, i.e. one submission per grp.

**Part 3: bonus 2.0**

This is to promote individual submissions. (think about how this bonus marks promotes individual submissions). This part of the assignment is derived from Game Theory. In this every group member needs to submit a number (can be any real number) up to 5 decimal digits.

Rules: The group member whose marks are close to the marks obtained in part 1 and part 2 combined will get this bonus 0.5 mark, and no two group members are allowed to place the same bet on marks or even have the same absolute difference with marks otherwise it would result to a 0 in this section for the entire group.

Example: Suppose you got 5.5+0.4=5.9 marks in parts 1 and 2 and bids for marks were: 1, 2, 3, 4, then the group member with 4 as a bet would get 0.5 mark extra, and the rest will get 0 for this section.

If the bets were 1,1,2,3, everyone gets a 0. (same bids)

If the marks obtained were 7.5 and bets were 1,2,7,8, everyone gets a 0 since the absolute difference of 3rd and 4th students from marks obtained is the same, i.e. 0.5.

In case your group is of n members and you don't submit n number of bids, this will also result in everyone getting a 0 for this section.

**For Part 2 and 3:**

You will be required to submit this form: https://forms.gle/ZJsE5cPE6hbX9y5r5 which will be used to tally your submission time and record the bids for marks guesses. In case of a mismatch in submission time on this form and Moodle, it will result in 0 marks in Part 2 of the assignment (only one submission is allowed on Moodle), so remember to submit the form as soon as u finish submission on Moodle. Fill the form as per the file name like if the file name is EntryNumber1_EntryNumber2_EntryNumber3_EntryNumber4_A2.py then EntryNumber1 corresponds to student 1, EntryNumber2 corresponds to student 2, and so on.