Bài tập điểm danh:

Họ và Tên: Hà Trung Chiến -20225794

Mã nguồn:

```
% Define the problem data
c = [6; 1; 3; 3; 1; -7];  % Coefficients of the objective function: c' * x
A = [-1, 1, 0, -1, 0, 1;   % Coefficients of the constraints: A * x <= b
     -2, 0, 1, 0, 0, -2;
      4, 0, 0, 1, 1, -3];
b = [15; 9; 2];             % Right-hand side of the constraints: A * x <= b

% Initial basis matrix (identity matrix as a simple choice)
% An identity matrix will start the algorithm with a basic feasible solution
% where the slack variables are the initial basic variables.
B = eye(size(A, 1));

% Solve the LP problem
[x_opt, f_opt] = solve_LP_inverse_simplex(c, A, b, B);

% Display the results
disp('Optimal solution:');
disp(x_opt);
disp('Optimal objective value:');
disp(f_opt);

function [x_opt, f_opt] = solve_LP_inverse_simplex(c, A, b, B)
    % Solve LP problem using inverse simplex method

    % Check dimensions
    [m, n] = size(A); % m: number of constraints, n: number of variables
    if length(c) ~= n || length(b) ~= m || size(B, 1) ~= m || size(B, 2) ~= m
        error('Dimension mismatch in inputs');
    end

    % Solve the initial basic feasible solution
    % xB represents the values of the basic variables
    xB = B \ b;  % Solve B * xB = b for xB, more efficient than inv(B)*b
    x_opt = zeros(n, 1); % Initialize the optimal solution vector with zeros
    basic_indices = 1:m; % Initially, the basic variables are the slack variables
    x_opt(basic_indices) = xB; % Assign the values of xB to the basic variables

    % Compute initial cost
    f_opt = c' * x_opt;

    % Set up the inverse of the basis matrix
    B_inv = inv(B);

    % Simplex iterations
    while true
        % Compute the reduced costs
        cB = c(basic_indices);  % Cost coefficients of basic variables
        lambda = cB' * B_inv;  % Simplex multipliers (dual variables)
```

```
        reduced_costs = c' - lambda * A;  % Reduced costs for all variables

        % Check optimality
        if all(reduced_costs >= 0)
            break;  % All reduced costs are non-negative, optimal solution found
        end

        % Determine entering variable (most negative reduced cost)
        [~, entering_index] = min(reduced_costs);

        % Determine leaving variable (minimum ratio test)
        d = B_inv * A(:, entering_index);  % Pivot column
        if all(d <= 0)
            error('Problem is unbounded');  % Unbounded solution
        end
        ratios = xB ./ d;  % Ratios for minimum ratio test
        ratios(d <= 0) = inf;  % Ignore negative or zero pivot elements
        [~, leaving_index] = min(ratios);  % Index of leaving variable

        % Update the basis
        basic_indices(leaving_index) = entering_index;  % Swap basic variable
        B(:, leaving_index) = A(:, entering_index);  % Update basis matrix
        B_inv = inv(B);  % Recalculate inverse of basis matrix
        xB = B_inv * b;  % Solve for new basic feasible solution
        x_opt = zeros(n, 1);
        x_opt(basic_indices) = xB;  % Update optimal solution vector
        f_opt = c' * x_opt;  % Calculate new optimal objective value
    end
end
```

bai3.m

/MATLAB Drive/bai3.m

```
57          [~, entering_index] = min(reduced_costs);
58
59          % Determine leaving variable (minimum ratio test)
60          d = B_inv * A(:, entering_index);  % Pivot column
61          if all(d <= 0)
62              error('Problem is unbounded');  % Unbounded solution
63          end
64          ratios = xB ./ d;  % Ratios for minimum ratio test
65          ratios(d <= 0) = inf;  % Ignore negative or zero pivot elements
66          [~, leaving_index] = min(ratios);  % Index of leaving variable
67
68          % Update the basis
69          basic_indices(leaving_index) = entering_index;  % Swap basic variable
```

Command Window

New to MATLAB? See resources for Getting Started.

```
>> bai3
Optimal solution:
     0
    39
     0
     0
    47
    15

Optimal objective value:
   -19

>>
```