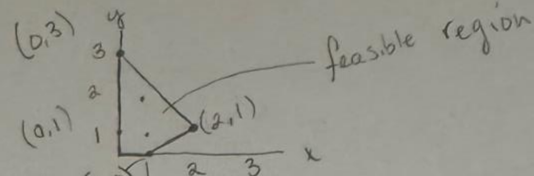


1a.) One feasible solution is (1,1) with value of 5. It is feasible because it is within the feasible region created by constraints.

1b.) $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{3-1}{0-2} = \frac{2}{-2} = -1$

$y - y_1 = m(x - x_1)$
$y - 3 = -1(x - 0)$
$y - 3 = -x$
$x + y = 3$ – test (0,0) for inequality
$0 \leq 3$
$x + y \leq 3$

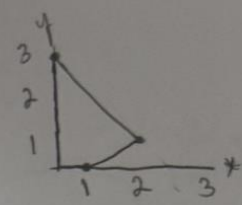
$m = \frac{0-1}{1-2} = \frac{-1}{-1} = 1$

$y - y_1 = 1(x - x_1)$
$y - 0 = 1(x - 1)$
$y = x - 1$
$x - y = 1$ – test (0,0)
$0 \leq 1$
$x - y \leq 1$

$y - y_1 = 0(x - x_1)$ $y \geq 0$
$y - 0 = y = 0$

4 constraints are:

$$\boxed{\begin{aligned} x + y &\leq 3 \\ x - y &\leq 1 \\ y &\geq 0 \\ x &\geq 0 \end{aligned}}$$



1c.) $Z = 3x + 2y$ Optimal solutions are the "vertices"

$(2,1)$ - optimal Solution

$Z = 3(2) + 2(1)$

$6 + 2$

$\boxed{= 8 \text{ optimal value for } (2,1)}$

2.) $\max_{x_1, x_2} \; -20x_1 - 5x_2$ C: $x + 2x_2 = 2$

$x_1 - 3x_2 \leq 3$

$x_1 \geq 0$

Step 1: Minimization to Maximization

Already a Max problem no change $\max_{x_1, x_2} \; -20x_1 - 5x_2$

Step 2: All variables have non-negativity constraint

Replace any variable not restricted to non-negative values, with subtraction of 2 positive

$x_j \rightarrow x_j' - x_j'' \quad x_j', x_j'' \geq 0$

$\max_{x_1, x_2} \; -20x_1 - 5x_2' + 5x_2''$

Subject to $x_1 + 2x_2' - 2x_2'' = 2$

$x_1 - 3x_2' + 3x_2'' \leq 3$

$x_1, x_2', x_2'' \geq 0$

Step 3: Equality to inequality

Replace each equality constraint with a pair of inequalities

$x_1 + 2x_2' - 2x_2'' = 2$

$x_1 + 2x_2' - 2x_2'' \leq 2 \qquad x_1 + 2x_2' - 2x_2'' \geq 2$

Step 4: Greater than equalities to less than

Negate the GE inequalities

$x_1 + 2x_2' - 2x_2'' \geq 2$

$\hookrightarrow -x_1 - 2x_2' + 2x_2'' \leq -2$

3.) Aggregate $= T(n)/n$

Multipush = linear with respect to A (k objects)

Sequence of n operations $= n \cdot O(A) = O(nA)$

Aggregate $= \frac{T(n)}{n} = \frac{O(nA)}{n} = O(A)$ where $A = k$ objects

4.) function FIND-SET(x)

// follow pointer from x back to its set object return Head (First)

return x.Set.first

```
function Union(x, y)
  // weighted hueristic is Set.Size
  Setx = x.Set
  Sety = y.Set
 if Setx.size ≥ Sety.size
  // Set last member of x to representative of y
  Setx.last.next = Sety.first
  // update all of y members set property to point at x
  While Sety.first ≠ NIL
        Sety.first.Set = Setx
        Sety.first = Sety.first.next

  Setx.last = Sety.last
  Setx.Size = Setx.Size + Sety.size
  // Destroy set object for ys list. Details for this Do not Matt
 Else // handle appending x to y
  Sety.last.next = Setx.first
  While Setx.first ≠ NIL
        Setx.first.Set = Sety
        Setx.first = Sety.first.next
 Sety.last = Setx.last
 Sety.size = Setx.Size + Sety.size
```