1. **Line representation**

   \[ y = Ax + B \quad (x_0 \neq x_1) \]

   1. **Slope:** \[ A = \frac{dy}{dx} = \frac{y_1 - y_0}{x_1 - x_0} \]
   2. **Y Intercept:** \[ B = y_0 - Ax_0 \]
   3. Alternatively, \[ x = Cy + D \quad (dy \neq 0) \]

2. **Incremental updates**

   \[ y_{i+1} = Ax_{i+1} + B \]
   \[ = A(x_i + 1) + B \]
   \[ = Ax_i + B + A \]
   \[ = y_i + A \]

3. **Implicit line representation**

   \[ f[x, y] = ax + by + c = 0 \]
   \[ a = dy, \ b = -dx, \ c = B \cdot dx \]

4. **Line drawing by decision variable**

   1. Decision variable at pixel \((x,y)\): (next pixel is NE if \(h\geq0\), otherwise E)

      \[ h[x, y] = f[x + 1, y + 1/2] \]

   2. Incremental update

      1. **Going East:** \[ h[x + 1, y] = h[x, y] + a \]
      2. **Going North-East:** \[ h[x + 1, y + 1] = h[x, y] + a + b \]
      3. **Initially:**

         \[ h[x_0, y_0] = f[x_0 + 1, y_0 + 1/2] = f[x_0, y_0] + a + b/2 = a + b/2 \]
5. Incremental update of decision variable in ellipses
   1. Implicit Function of an ellipse (circle if \(a=b\))
      \[
      f(x, y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - R^2
      \]
   2. Decision variable at a pixel \(\{x,y\}\)
      (going East/SouthEast)
      \[
      h(x, y) = \frac{(x + 1)^2}{a^2} + \frac{(y - \frac{1}{2})^2}{b^2} - R^2
      \]
   1. First differences
      1. If going East:
         \[
         \Delta_E h(x, y) = h(x + 1, y) - h(x, y) = \frac{2x + 3}{a^2}
         \]
      2. If going South-East:
         \[
         \Delta_{SE} h(x, y) = h(x + 1, y - 1) - h(x, y) = \frac{2x + 3}{a^2} - \frac{2y - 2}{b^2}
         \]
   2. Second differences
      1. Updating \(\Delta_E h(x, y)\)
         1. If going East:
            \[
            \Delta_{E,E} h(x, y) = \Delta_E h(x + 1, y) - \Delta_E h(x, y) = \frac{2}{a^2}
            \]
         2. If going South-East:
            \[
            \Delta_{E,SE} h(x, y) = \Delta_E h(x + 1, y - 1) - \Delta_E h(x, y) = \frac{2}{a^2}
            \]
      2. Updating \(\Delta_{SE} h(x, y)\)
         1. If going East:
            \[
            \Delta_{SE,E} h(x, y) = \Delta_{SE} h(x + 1, y) - \Delta_{SE} h(x, y) = \frac{2}{a^2}
            \]
         2. If going South-East:
            \[
            \Delta_{SE,SE} h(x, y) = \Delta_{SE} h(x + 1, y - 1) - \Delta_{SE} h(x, y) = \frac{2}{a^2} + \frac{2}{b^2}
            \]
6. Scan-converting polygons (or Polygon filling)

1. Basic Idea: Intersect the scan-line with each polygon edge, and draw pixels between intersections.
   1. Apply “on-off” walk: The pixels that are inside the polygon are between each odd-even intersection pair

2. Algorithm sketch
   1. Find out min, max y coordinate of the polygon
   2. Increment y from ymin to ymax
      1. Update the list of edges that intersect with the scan line
      2. Use line-scan-conversion to draw pixels on each edge with that y-coordinate
      3. Sort the edges by the x coordinates of the first pixel on the scanline.
      4. Draw pixels between the pixels of odd-even edges

3. Problems and solutions
   1. What happens when scan-line passes through a polygon vertex, how many intersections should be counted?
      1. For each edge, only count the vertex with larger y coordinate than the other vertex.
      2. Ignore both vertices of a horizontal line.