Computer Graphics

Line clipping Algorithm

Md. Biplob Hosen
Assistant Professor, IIT-JU.
Email: biplob.hosen@juniv.edu

Lecture Outlines

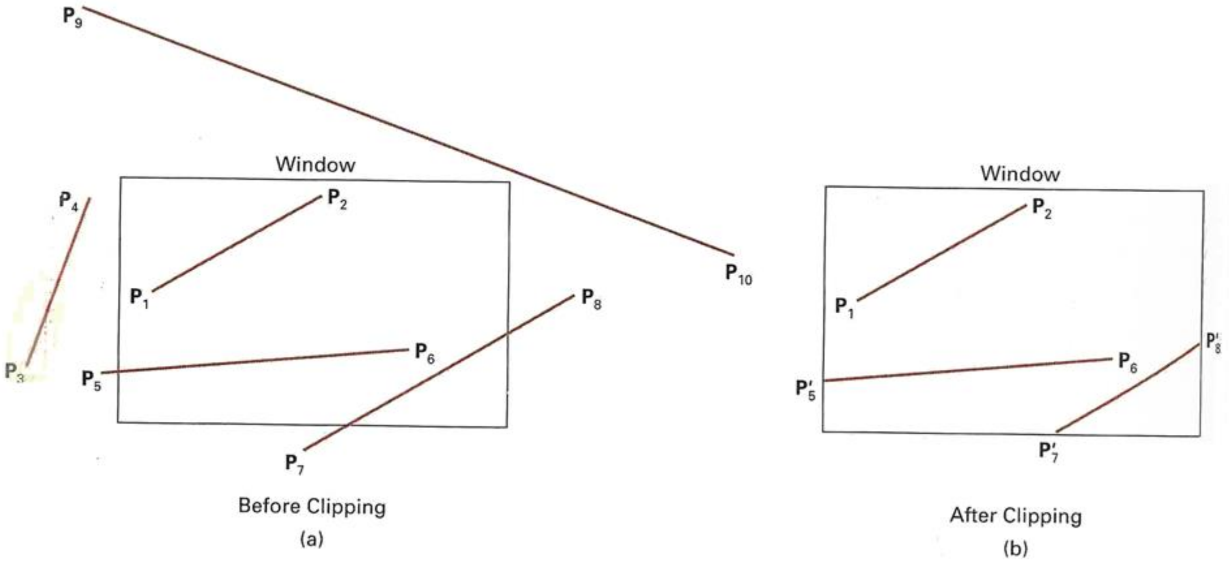
- Line Clipping Algorithms -
  - ✓ Cohen-Sutherland Algorithm
  - ✓ Midpoint Subdivision Algorithm

Line Clipping

- **Line clipping procedure -**
  - Test a given line segment to determine whether it lies completely inside the clipping window.
  - If it doesn’t, we try to determine whether it lies completely outside the window.
  - If we can’t identify a line as completely inside or completely outside, we must perform intersection calculations with one or more clipping boundaries.

Continue…

- Checking the line endpoints ⇒ inside-outside test.



Before Clipping
(a)

After Clipping
(b)

- Line clipping Algorithm:
  - Cohen-Sutherland Algorithm;
  - Midpoint Subdivision Algorithm;
  - Liang-Barsky Algorithm.

Cohen-Sutherland Algorithm

- Divide the line clipping process into two phases:

  1) Identify those lines which intersect the clipping window and so need to be clipped;

  2) Perform the clipping.

- All lines fall into one of the following clipping categories:

  1) Visible: Both end points of the line lie within the window.

  2) Not visible: The line definitely lies outside the window. This will occur if the line from (x1,y1) to (x2,y2) satisfies any one of the following inequalities:

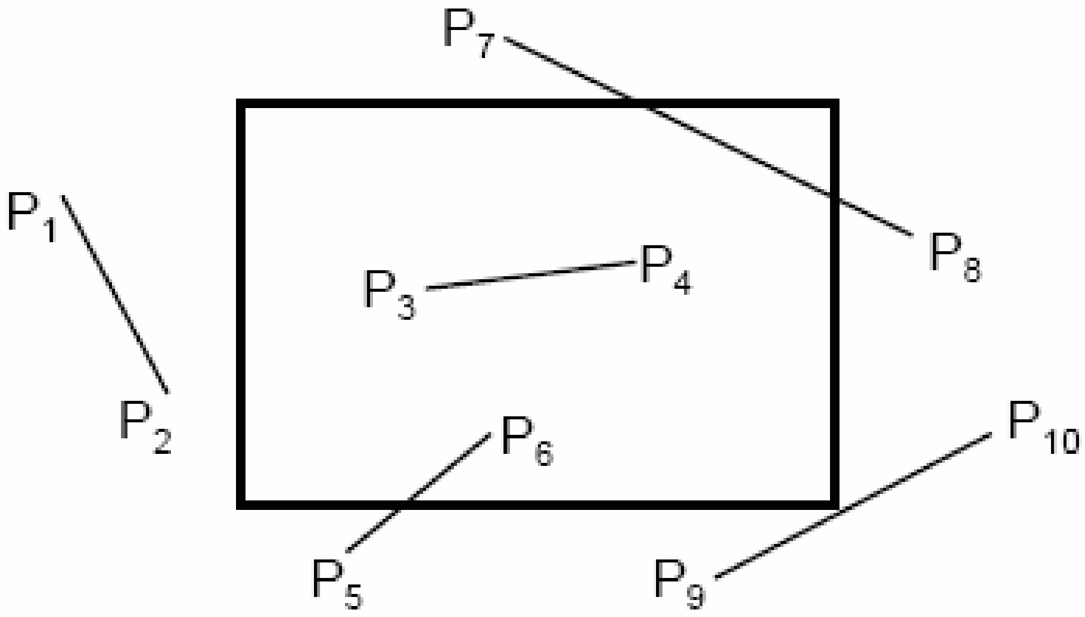
$$x_{1,}x_2 > x_{\max} \qquad y_1, y_2 > y_{\max}$$

$$x_{1,}x_2 < x_{\min} \qquad y_1, y_2 < y_{\min}$$

  3) Clipping candidate: the line is in neither category 1 nor 2.

Continue…

Find the part of a line inside the clip window

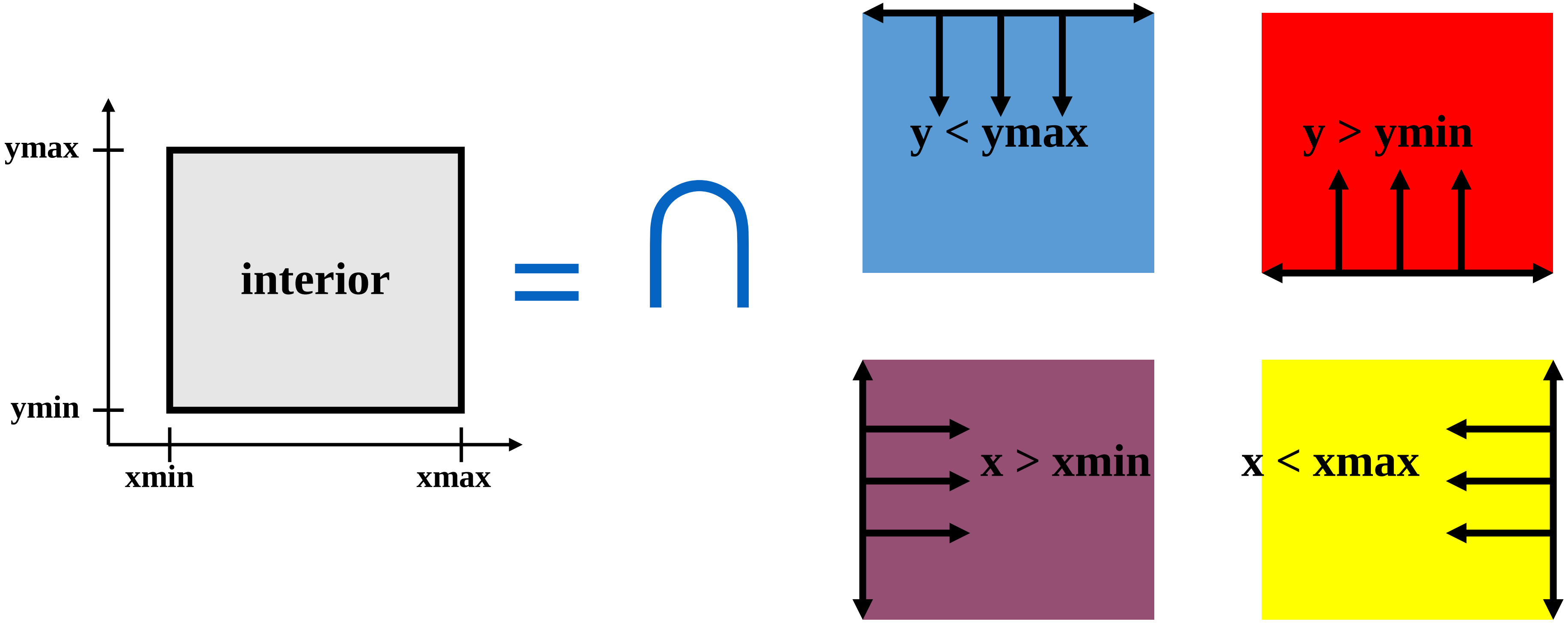


p_3p_4 is in category 1(Visible)

p_1p_2 is in category 2(Not Visible)

$p_5p_6, p_7p_8, p_9p_{10}$ is in category 3(Clipping candidate)

Continue...



ymax

ymin

xmin

xmax

interior

$= \cap$

$y < ymax$

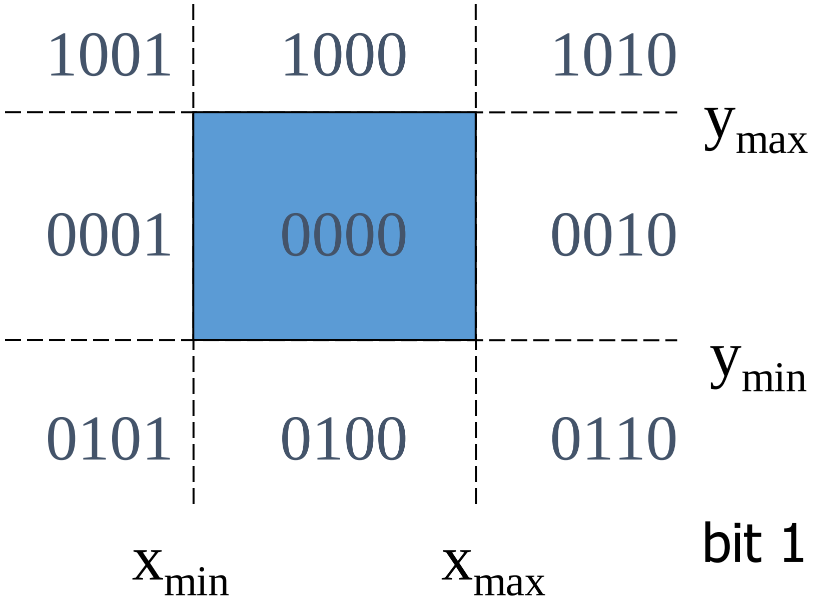
$y > ymin$

$x > xmin$

$x < xmax$

Continue…

- Assign a four-bit pattern (Region Code) to each endpoint of the given segment. The code is determined according to which of the following nine regions of the plane the endpoint lies in.

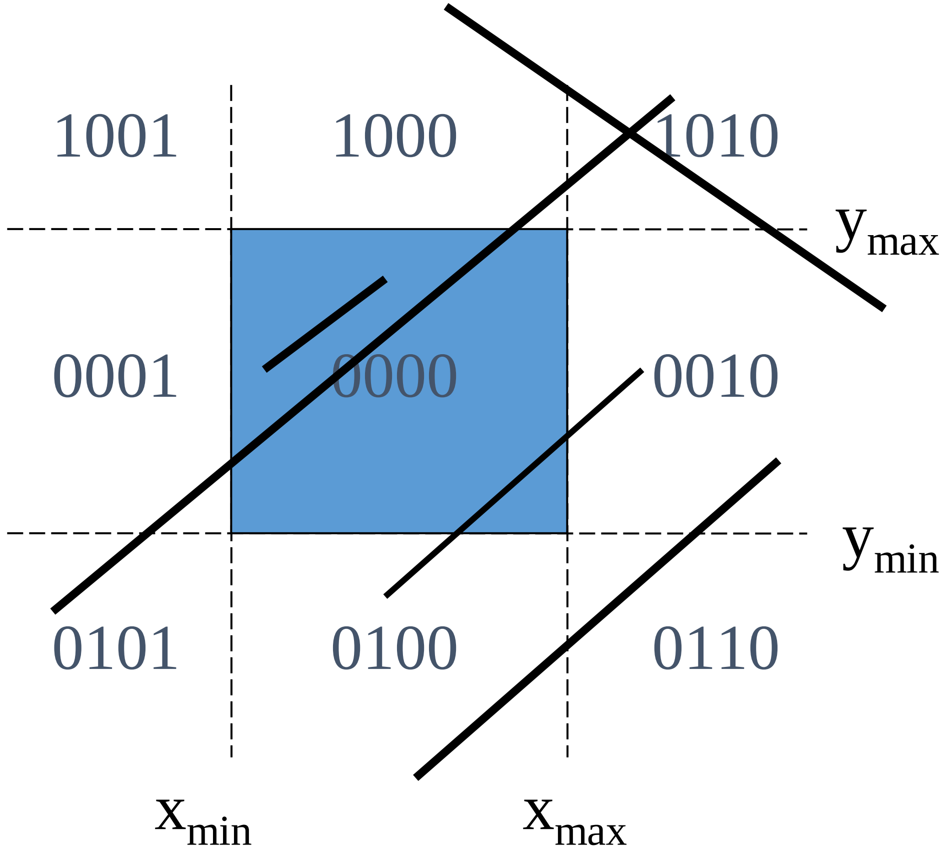


bit 1 : bit 2 : bit 3 : bit 4

Top : Bottom : Right : Left:

- Of course, a point with code 0000 is inside the window.

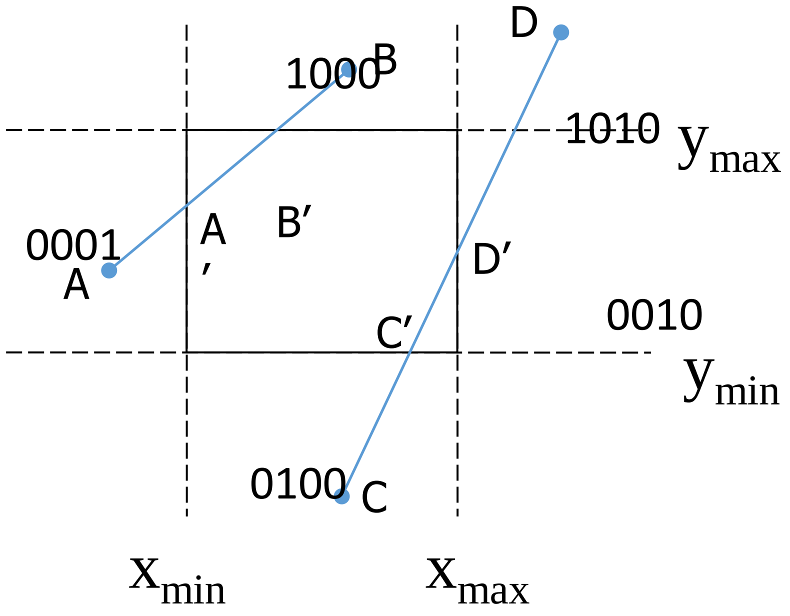
Continue…


1001
1000
1010
y_{max}
0001
0000
0010
y_{min}
0101
0100
0110
x_{min}
x_{max}

Continue…

- If both endpoint codes are 0000, the line segment is visible (inside).
- The logical AND of the two endpoint codes -
    - not completely 0000 , the line segment is not visible (outside).
    - completely 0000, the line segment maybe inside (and outside).
- Lines that cannot be identified as being completely inside or completely outside, a clipping window are then checked for intersection with the window border lines.

Continue…



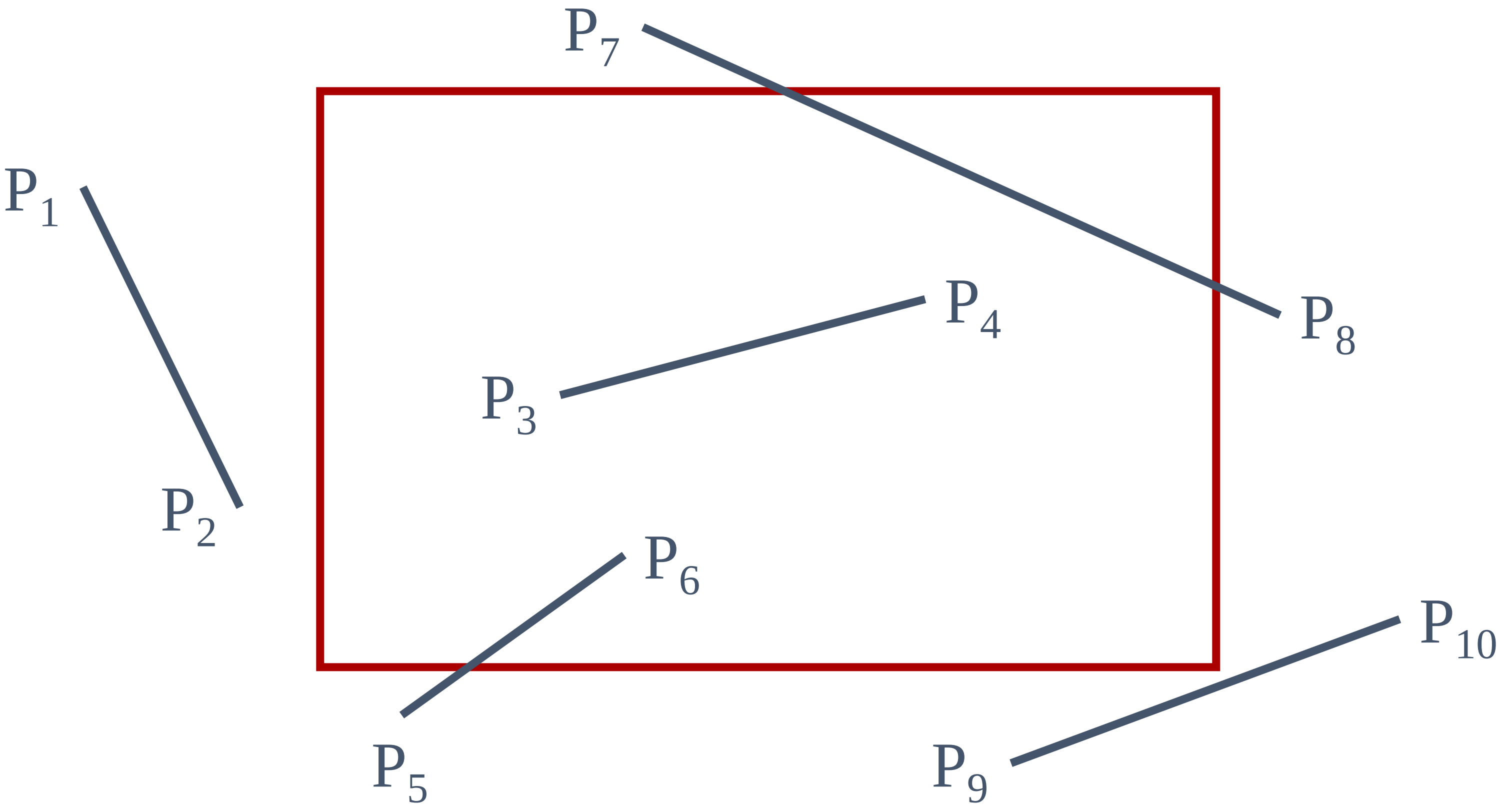
- Consider code of an end point
  - if bit 1 is 1, intersect with line y = Ymax
  - if bit 2 is 1, intersect with line y = Ymin
  - if bit 3 is 1, intersect with line x = Xmax
  - if bit 4 is 1, intersect with line x = Xmin
- Consider line CD.
  - If endpoint C is chosen, then the bottom boundary line Y=Ymin is selected for computing intersection
  - If endpoint D is chosen, then either the top boundary line Y=Ymax or the right boundary line X=Xmax is used.
  - The coordinates of the intersection point are:
    - $\begin{cases} x_i = x_{min} \text{ or } x_{max} \\ y_i = y_1 + m(x_i - x_1) \end{cases}$ if the boundary line is vertical
    - $\begin{cases} x_i = x_1 + (y_i - y_1)/m \\ y_i = y_{min} \text{ or } y_{max} \end{cases}$ if the boundary line is horizontal
    - where $m = \dfrac{y_{end} - y_0}{x_{end} - x_0}$

Continue…

- Replace endpoint (x1,y1) with the intersection point(xi,yi), effectively eliminating the portion of the original line that is on the outside of the selected window boundary.
- The new endpoint is then assigned an updated region code and the clipped line re-categoriged and handled in the same way.
- This iterative process terminates when we finally reach a clipped line that belongs to either category 1(visible) or category 2(not visible).

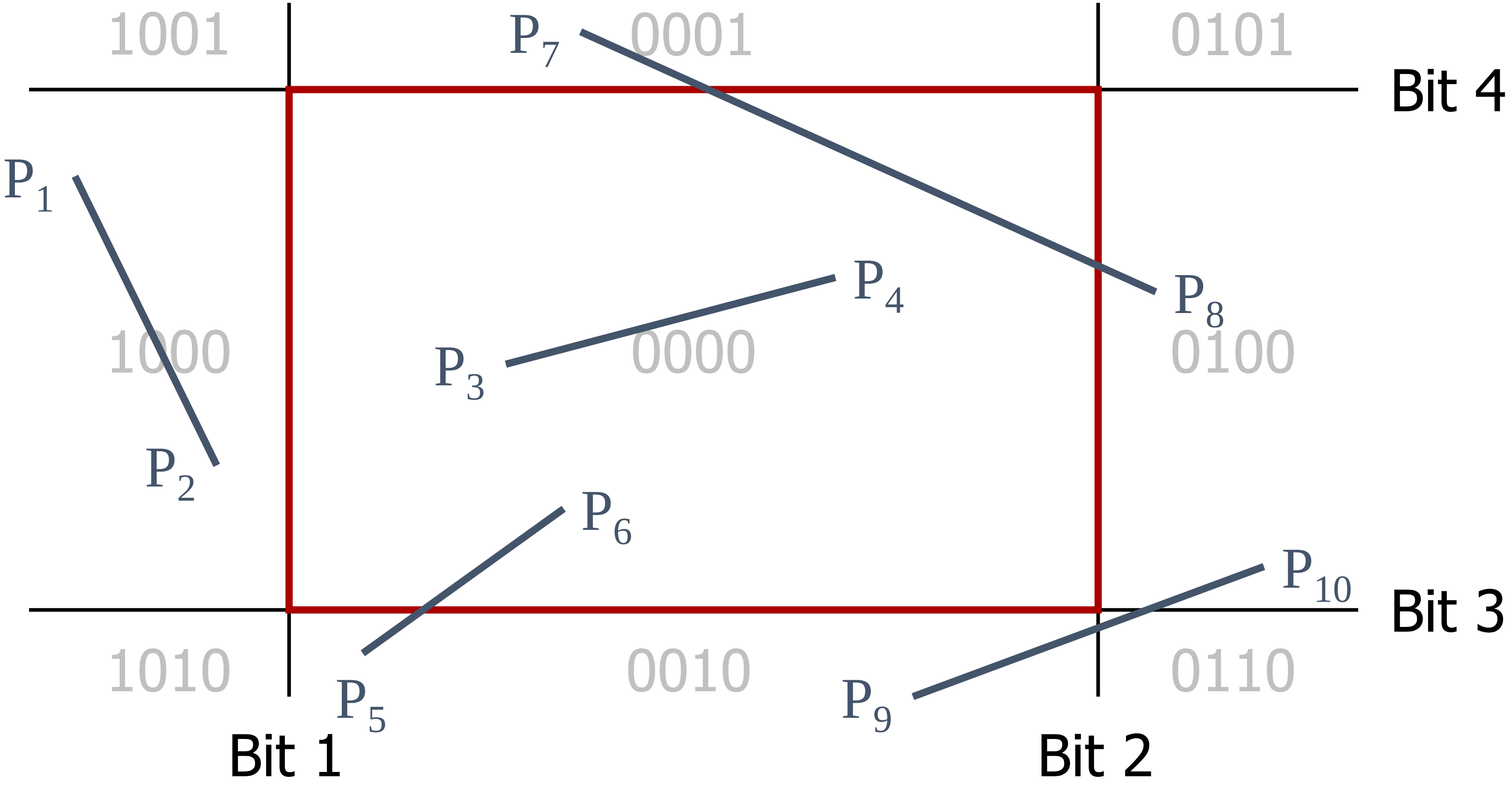
Continue…

- Use simple tests to classify easy cases first:

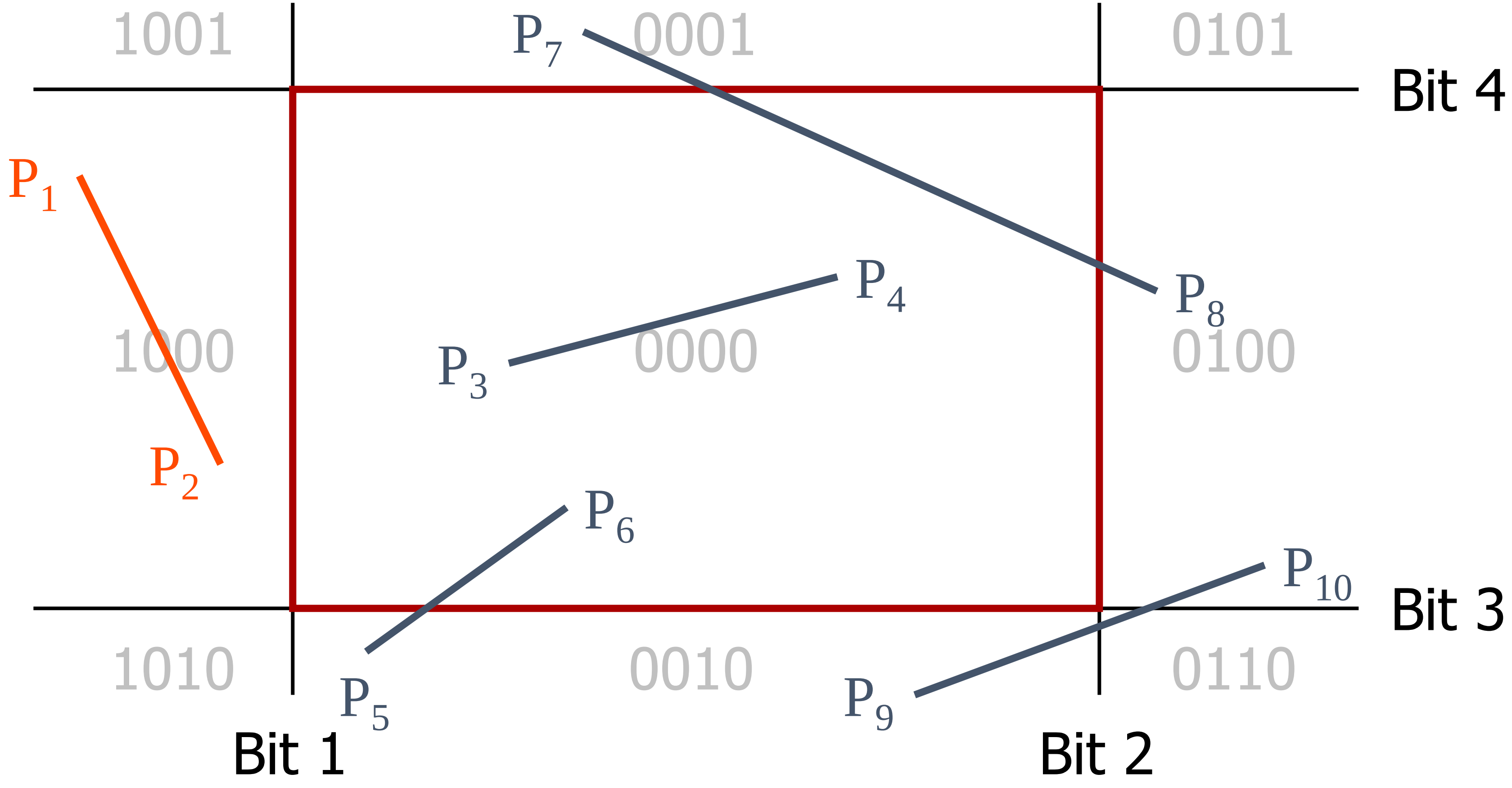


Continue…

- Classify some lines quickly by AND of bit-codes representing regions of two endpoints (must be 0).

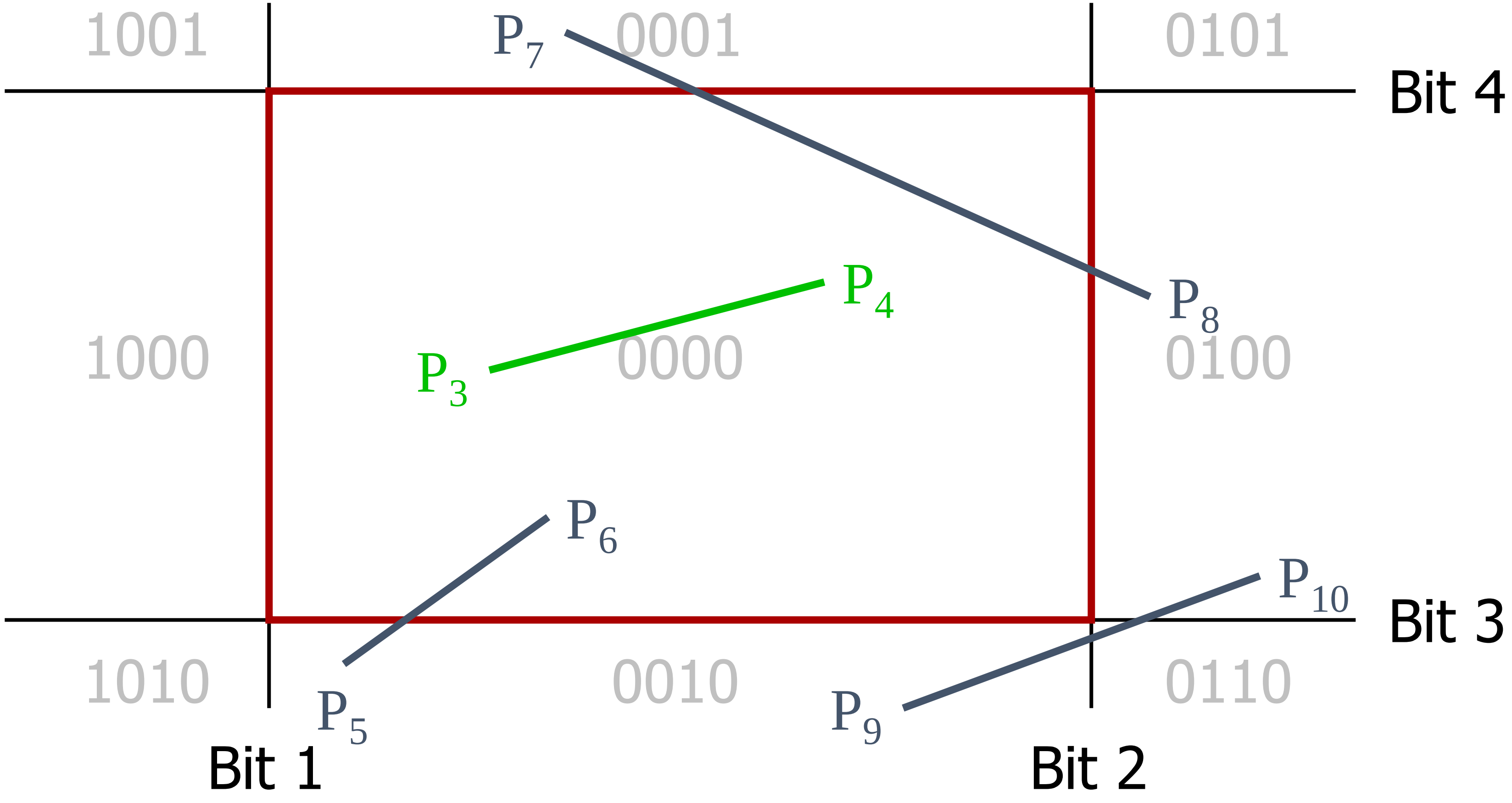


Continue…



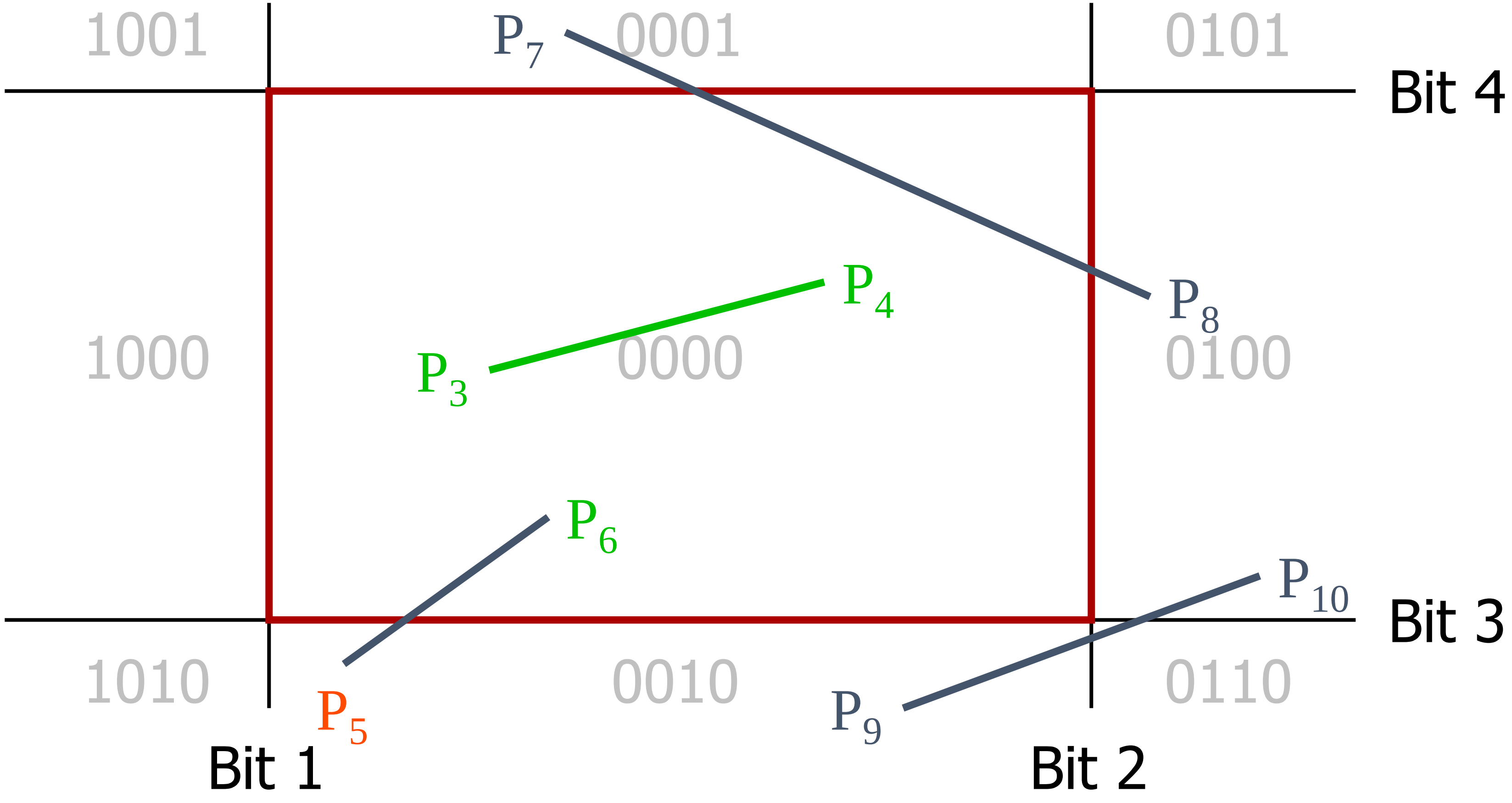
1001
P$_7$
0001
0101
Bit 4
P$_1$
P$_4$
P$_8$
1000
P$_3$
0000
0100
P$_2$
P$_6$
P$_{10}$
Bit 3
1010
0010
0110
P$_5$
P$_9$
Bit 1
Bit 2

Continue…

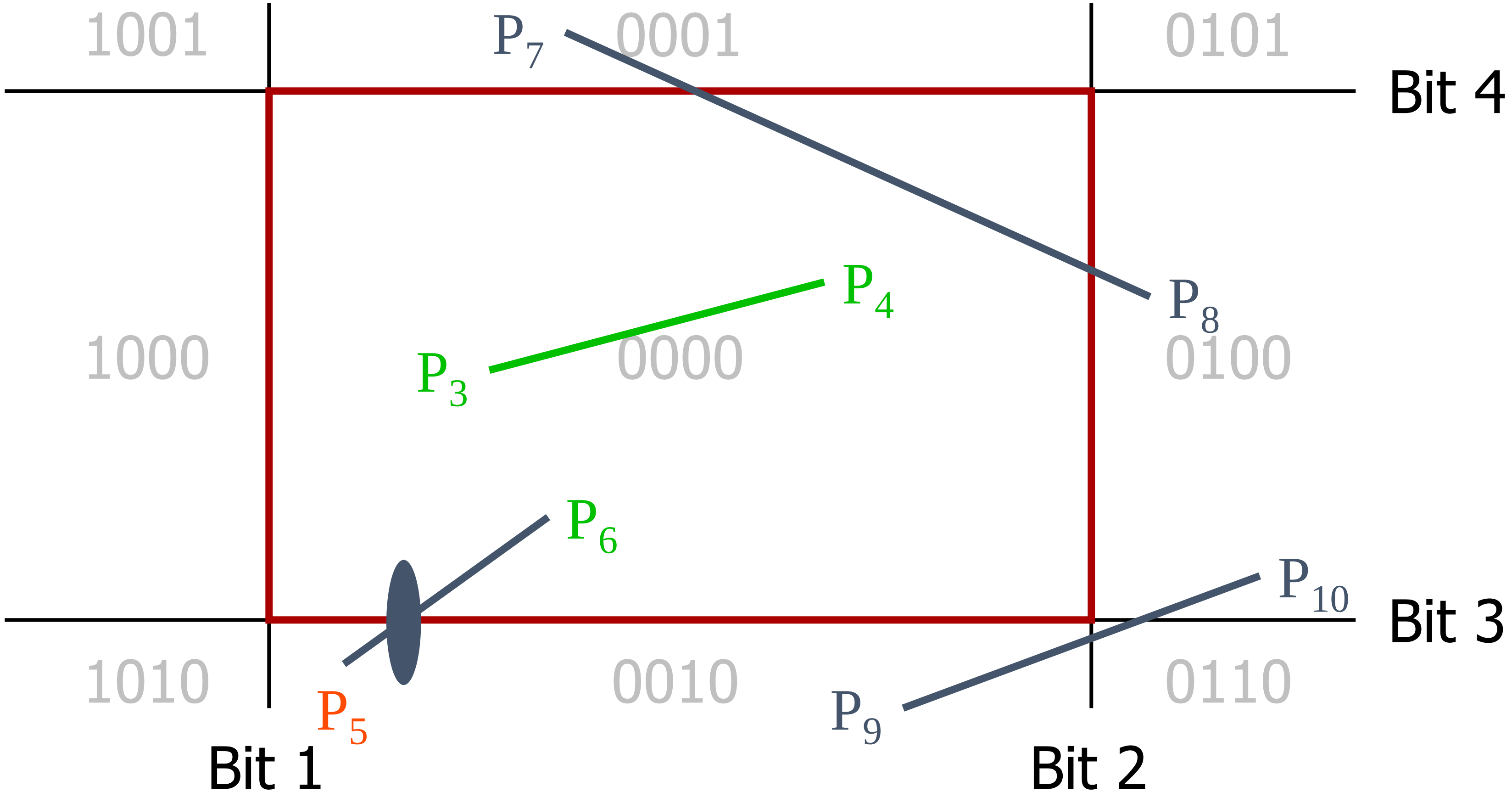

1001
0001
0101
Bit 4
1000
0000
0100
1010
0010
0110
Bit 3
Bit 1
Bit 2
P_3
P_4
P_5
P_6
P_7
P_8
P_9
P_{10}

Continue…

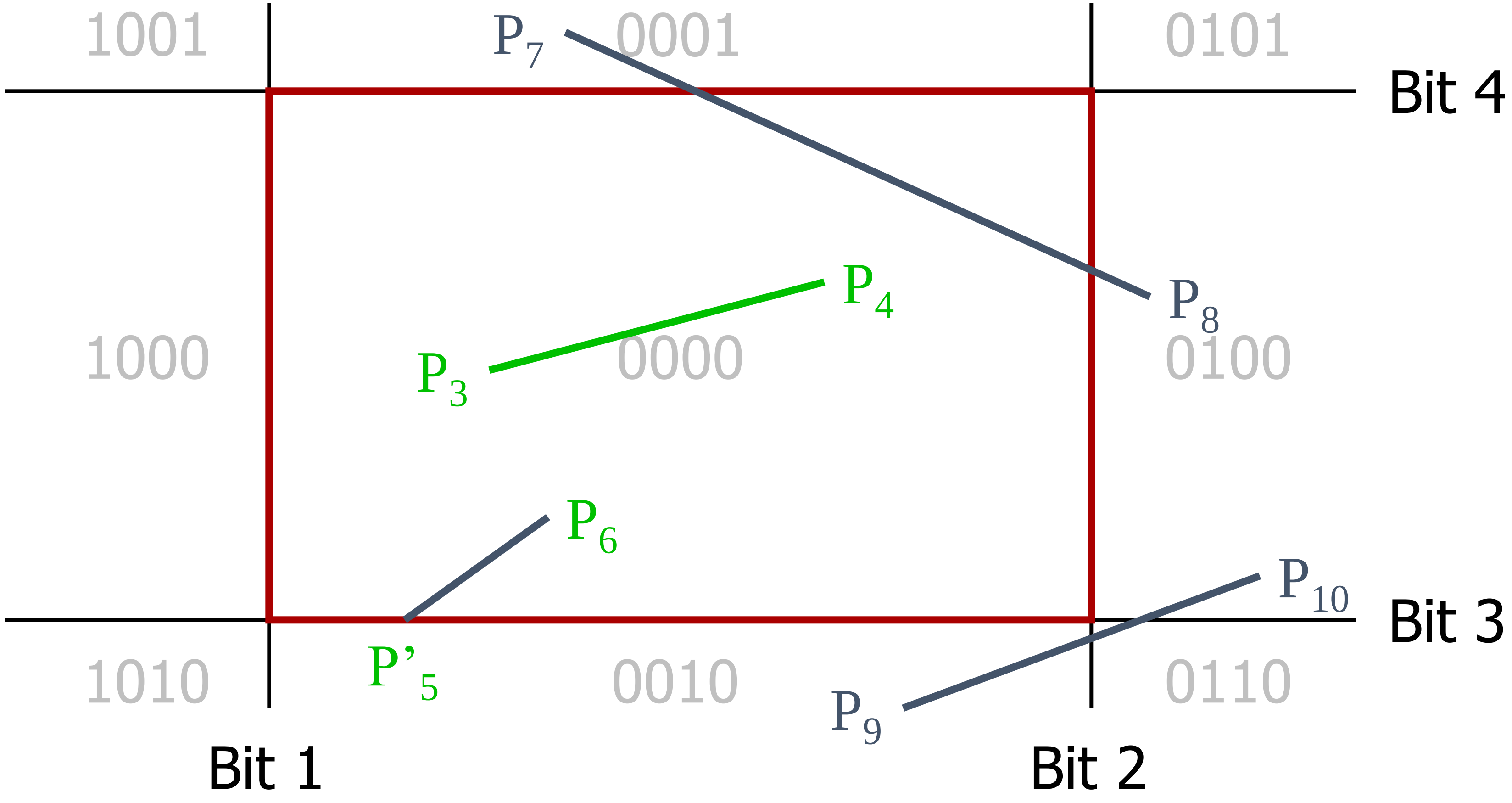
- Compute intersections with window boundary for lines that can't be classified quickly:



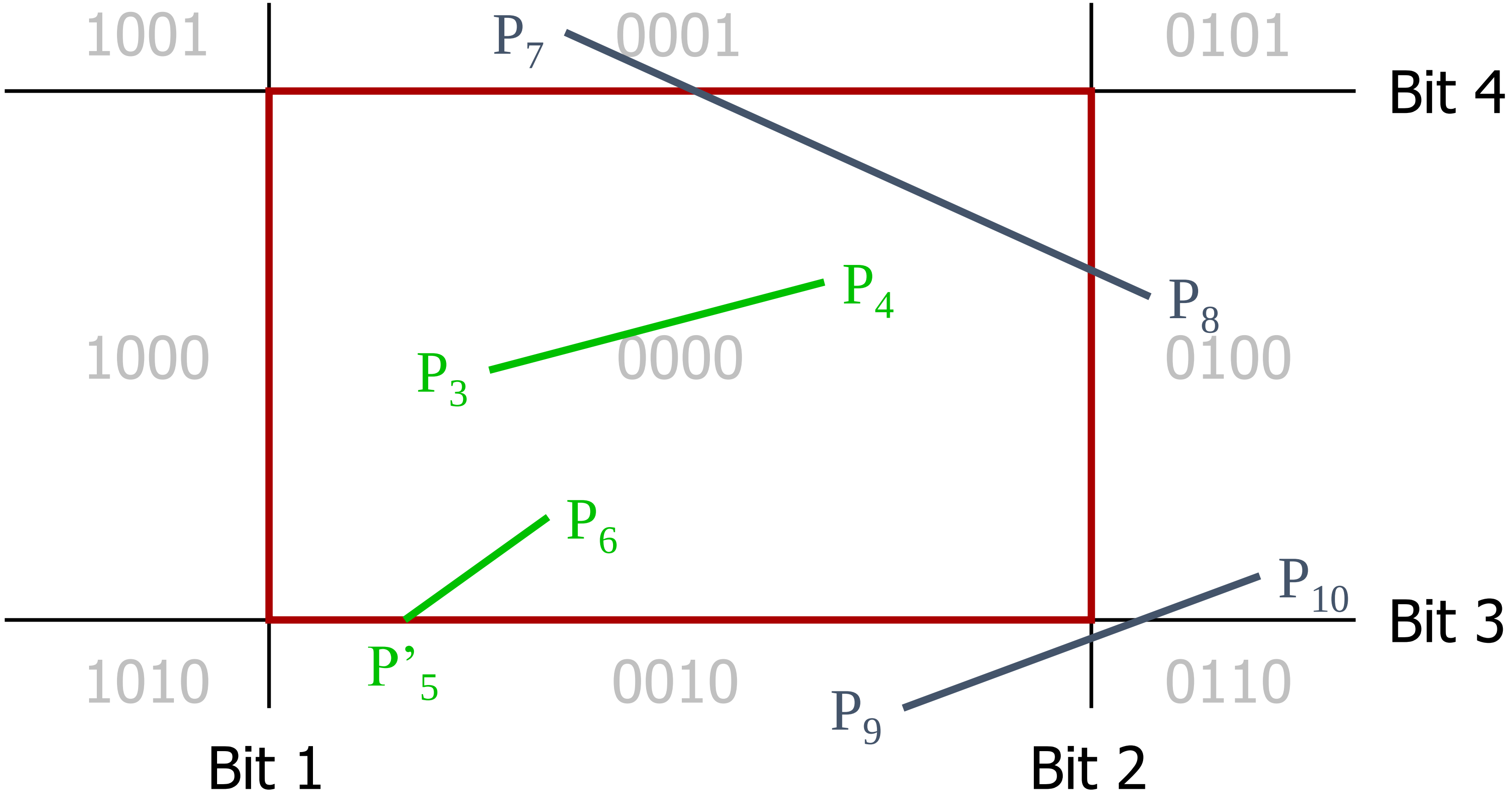
Continue…


1001
P7
0001
0101
Bit 4
P4
P8
1000
P3
0000
0100
P6
P10
Bit 3
1010
0010
0110
P5
P9
Bit 1
Bit 2

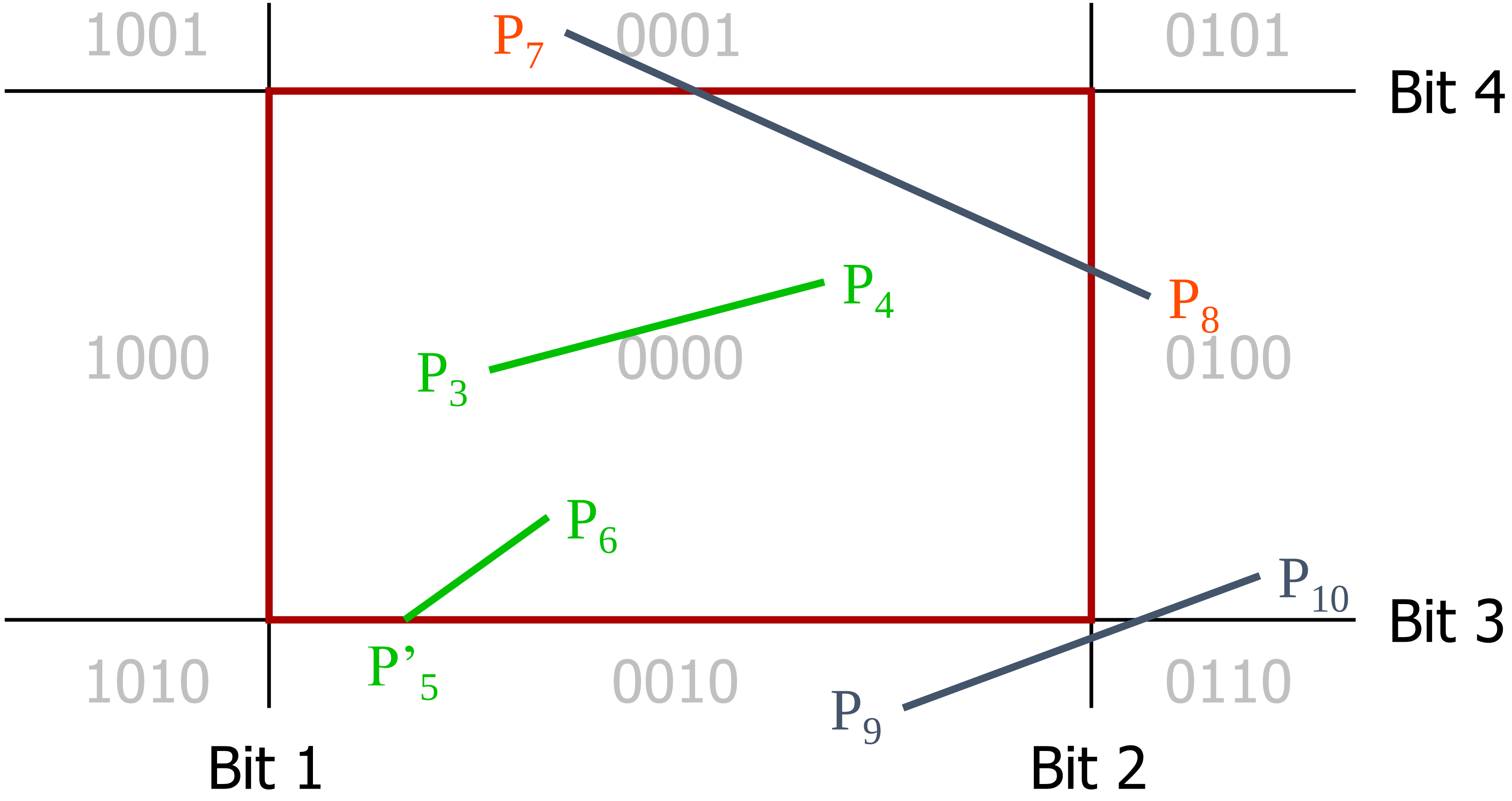
Continue…


1001
P7
0001
0101
Bit 4
P4
P8
1000
P3
0000
0100
P6
P10
Bit 3
P'5
1010
0010
P9
0110
Bit 1
Bit 2

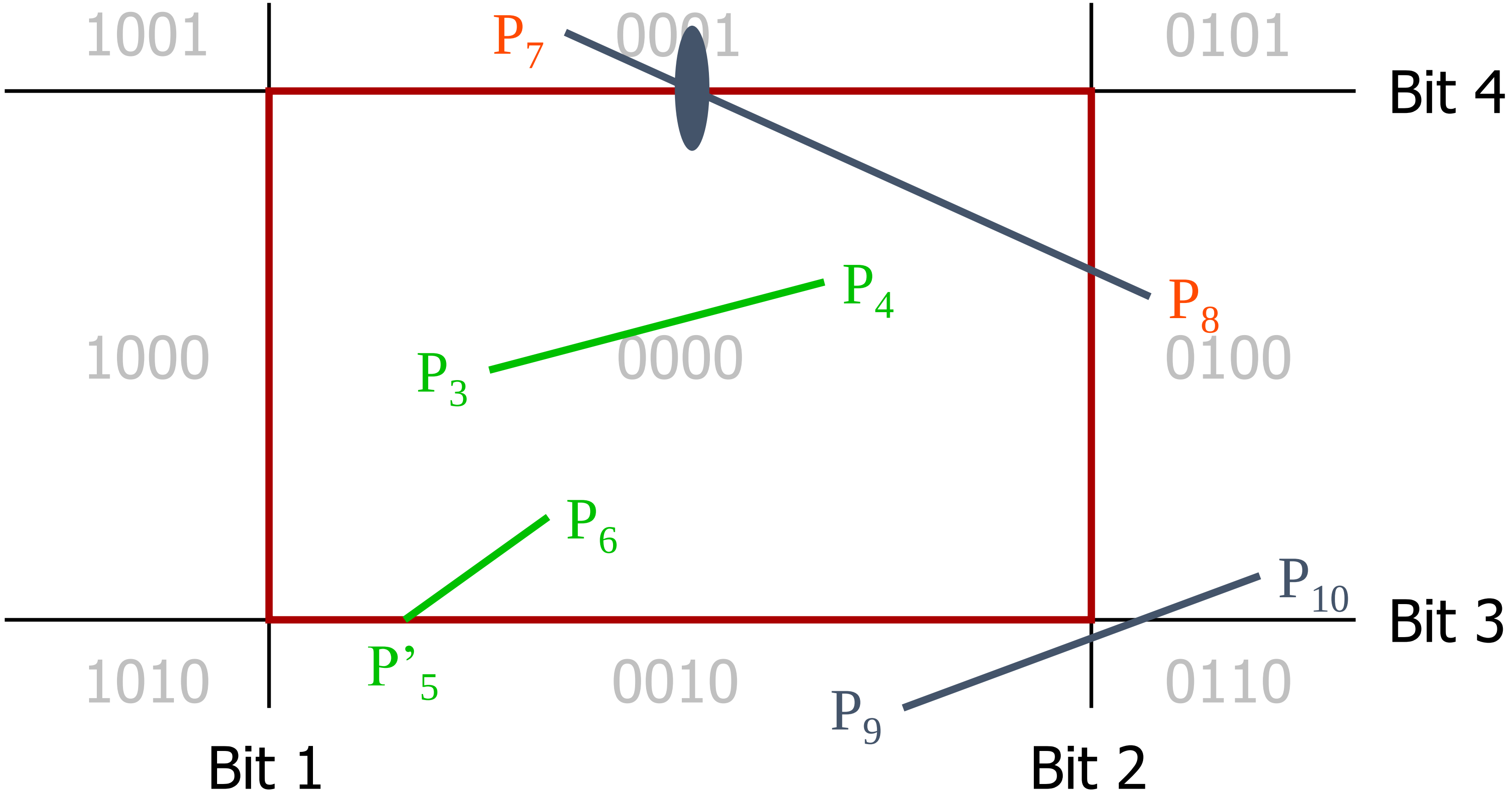
Continue…


1001
P7
0001
0101
Bit 4
P4
P8
P3
1000
0000
0100
P6
P10
Bit 3
P'5
1010
0010
P9
0110
Bit 1
Bit 2

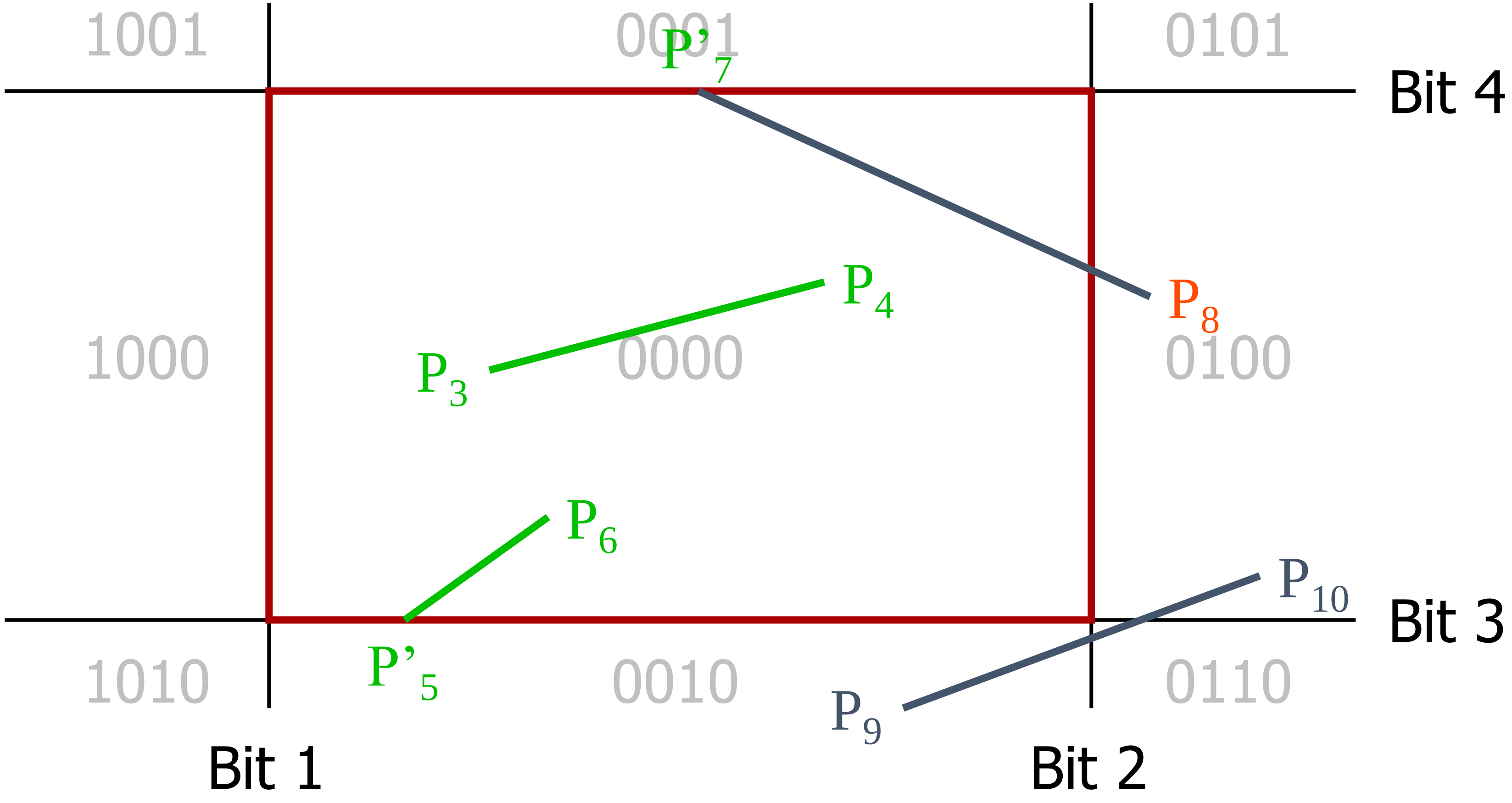
Continue…


1001
P7
0001
0101
Bit 4
P4
P8
1000
P3
0000
0100
P6
P10
Bit 3
P'5
1010
0010
P9
0110
Bit 1
Bit 2

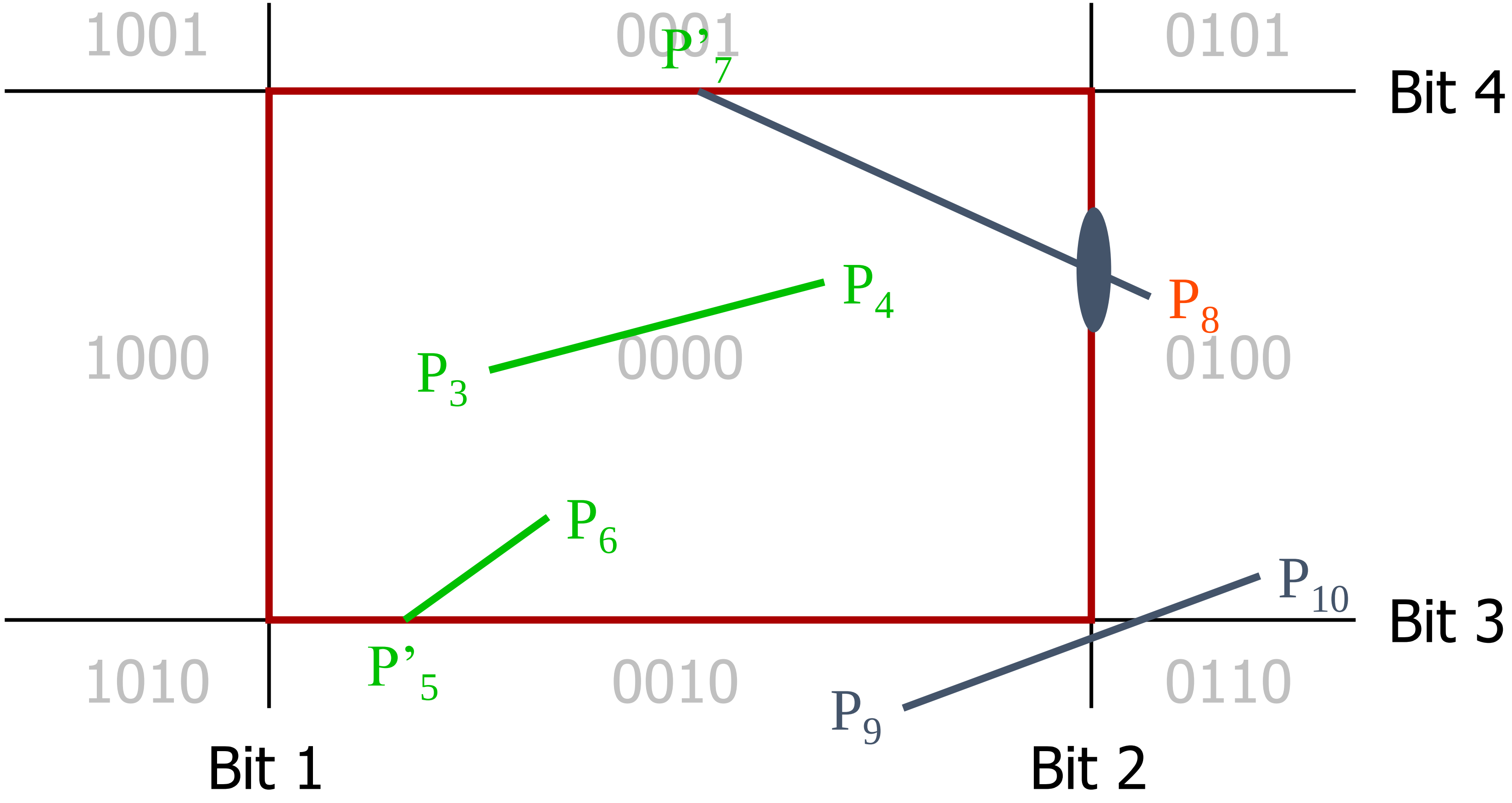
Continue…


1001
P7
0001
0101
Bit 4
P4
P8
1000
P3
0000
0100
P6
P10
Bit 3
P'5
1010
0010
P9
0110
Bit 1
Bit 2

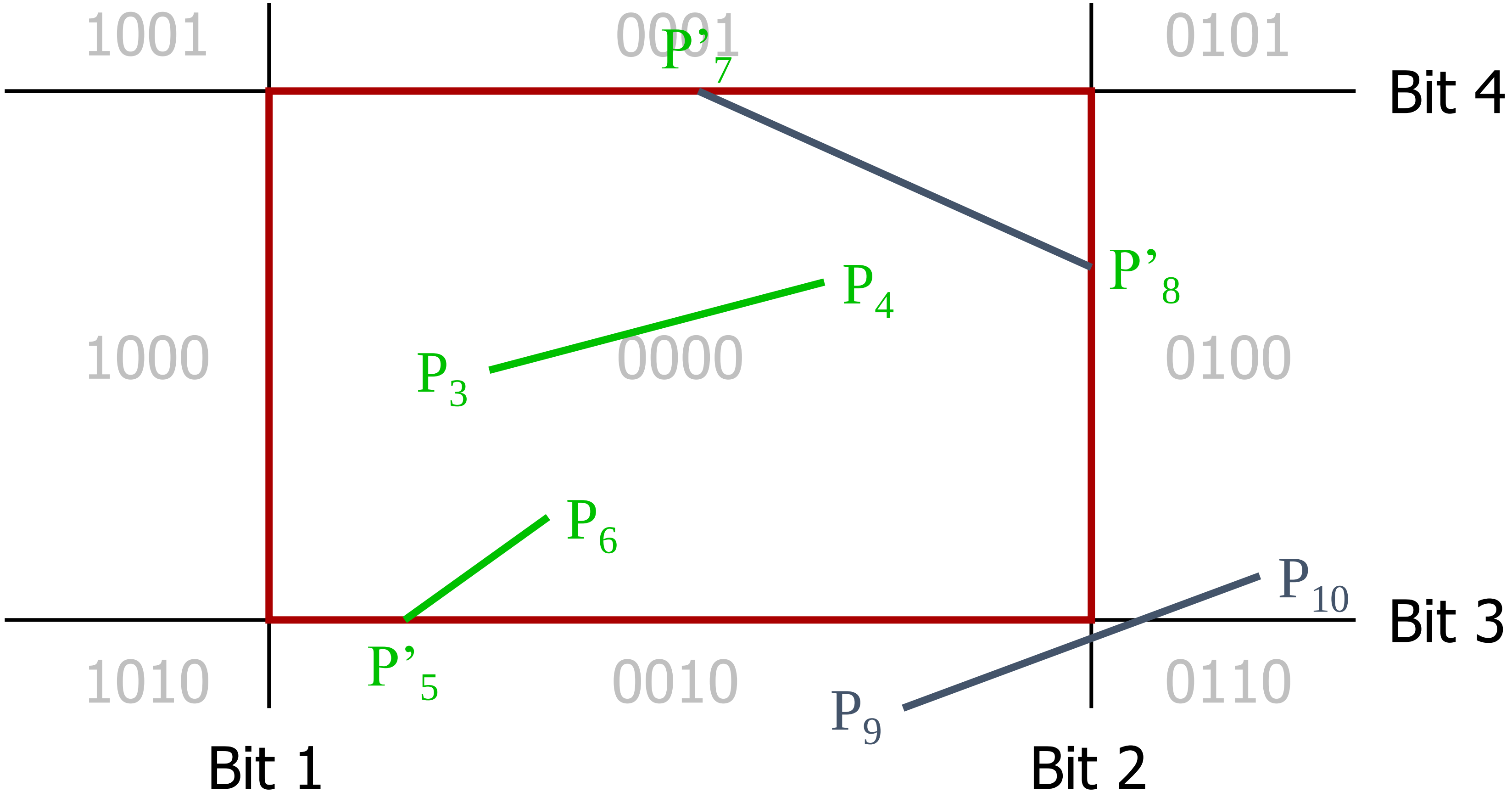
Continue…


1001
0001
P'_7
0101
Bit 4
P_4
P_8
1000
P_3
0000
0100
P_6
P_{10}
Bit 3
P'_5
1010
0010
P_9
0110
Bit 1
Bit 2

Continue…

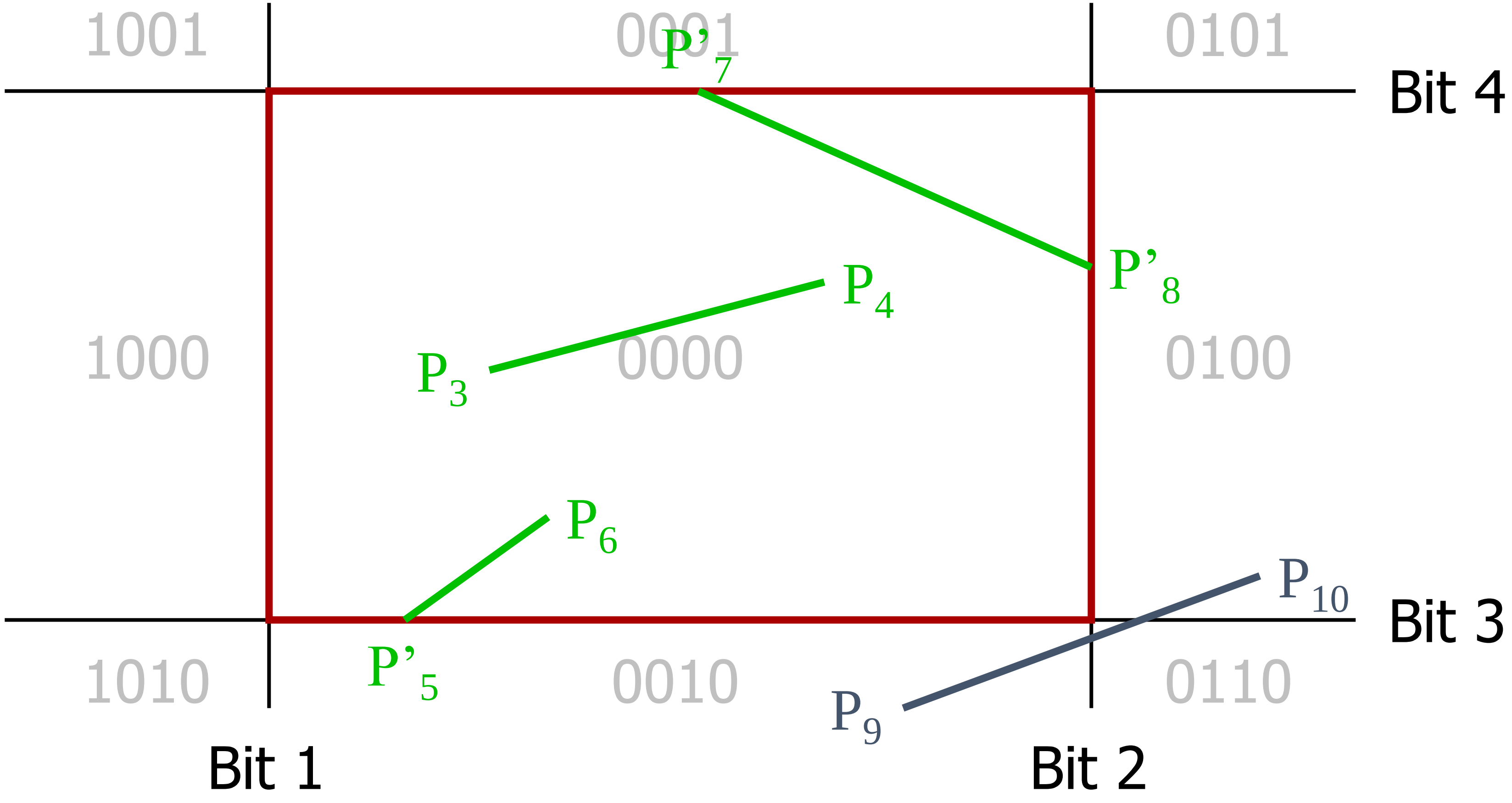

1001
0001
P'$_7$
0101
Bit 4
P$_4$
P$_8$
1000
P$_3$
0000
0100
P$_6$
P$_{10}$
Bit 3
1010
P'$_5$
0010
P$_9$
0110
Bit 1
Bit 2

Continue…

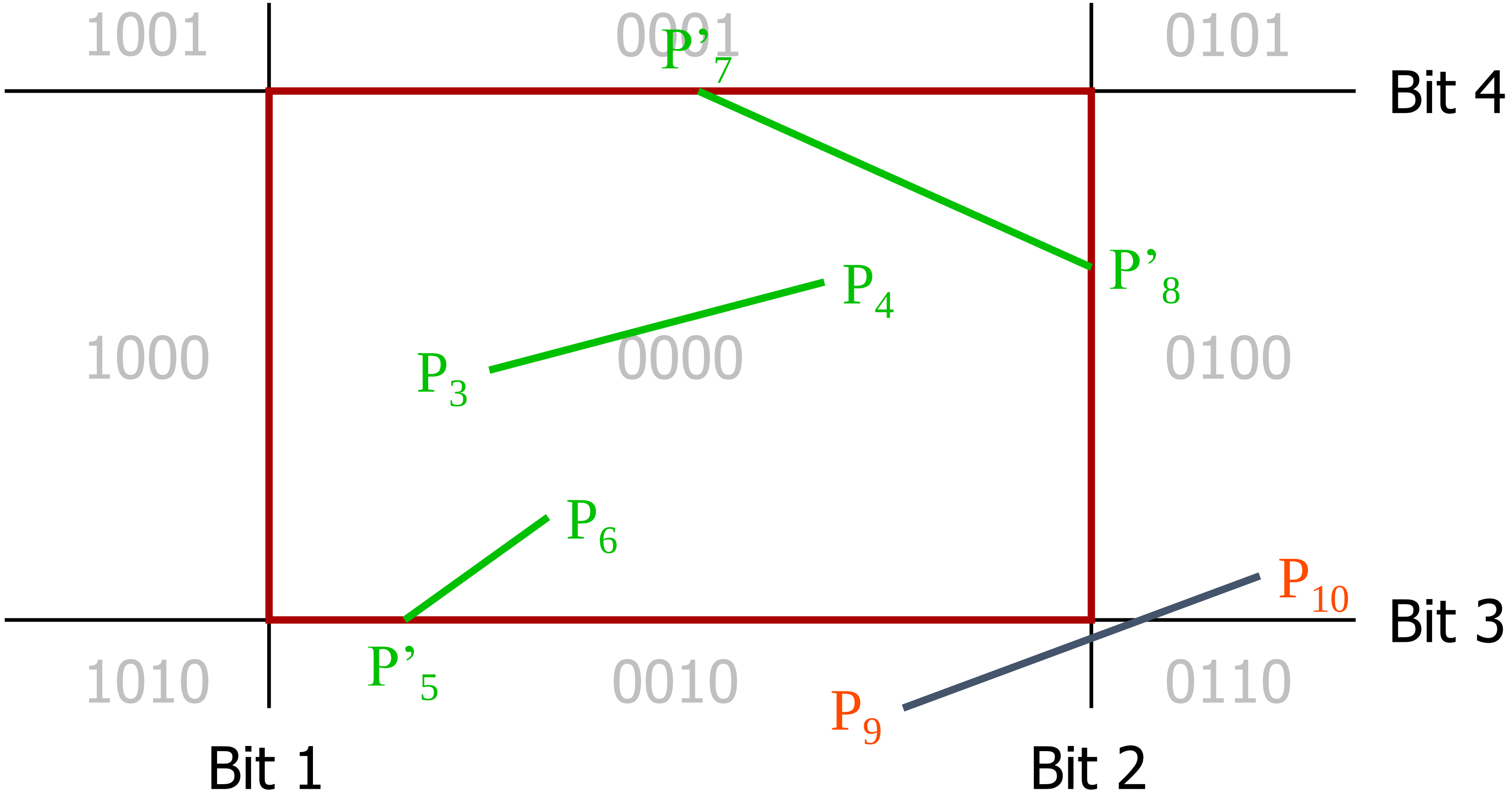


1001
0001
0101
P'$_7$
Bit 4
P'$_8$
P$_4$
1000
P$_3$
0000
0100
P$_6$
P$_{10}$
Bit 3
P'$_5$
1010
0010
P$_9$
0110
Bit 1
Bit 2

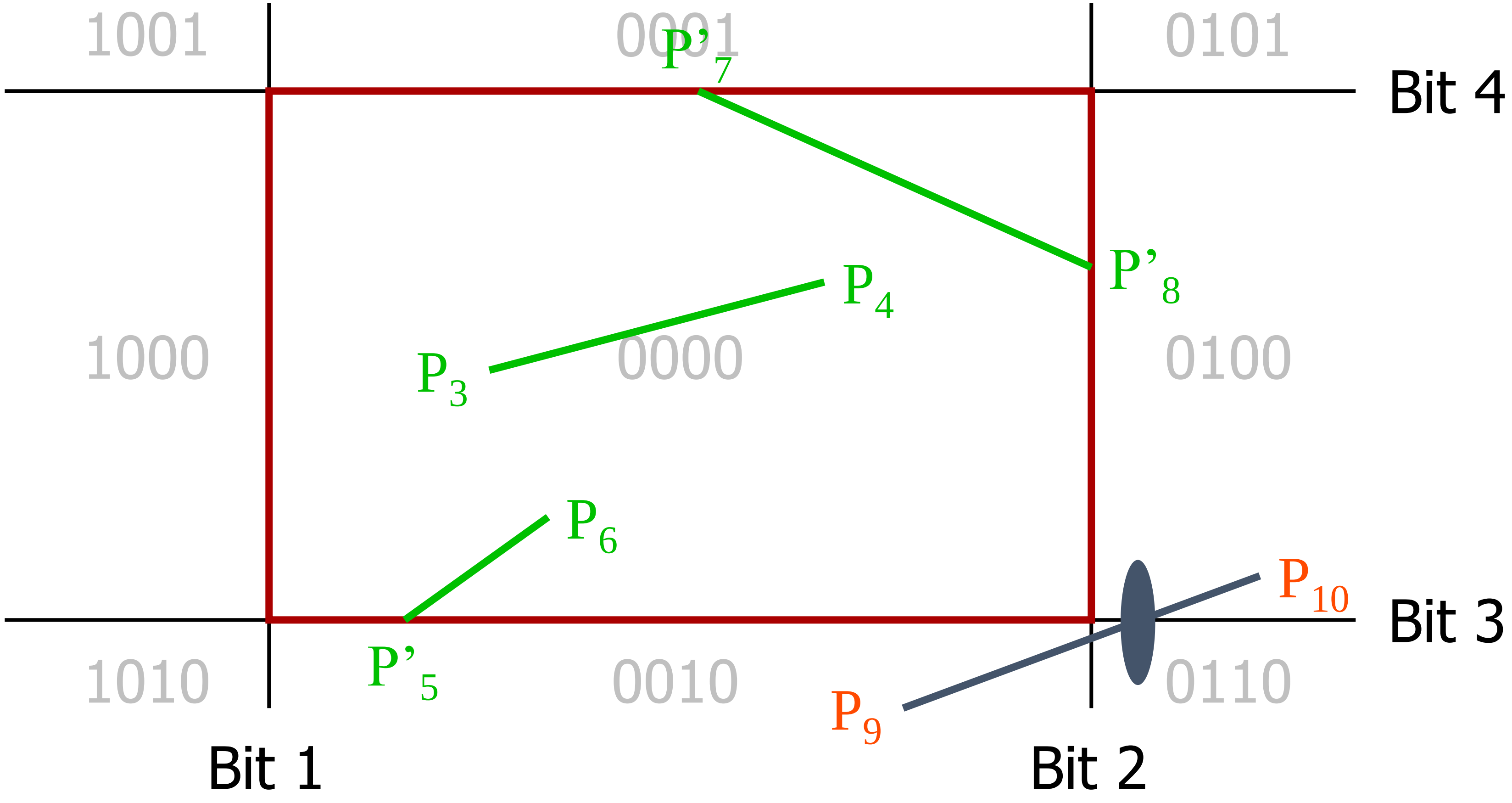
Continue…


1001
0001
0101
P'7
Bit 4
P'8
P4
1000
P3
0000
0100
P6
P10
Bit 3
P'5
1010
0010
0110
P9
Bit 1
Bit 2

Continue…

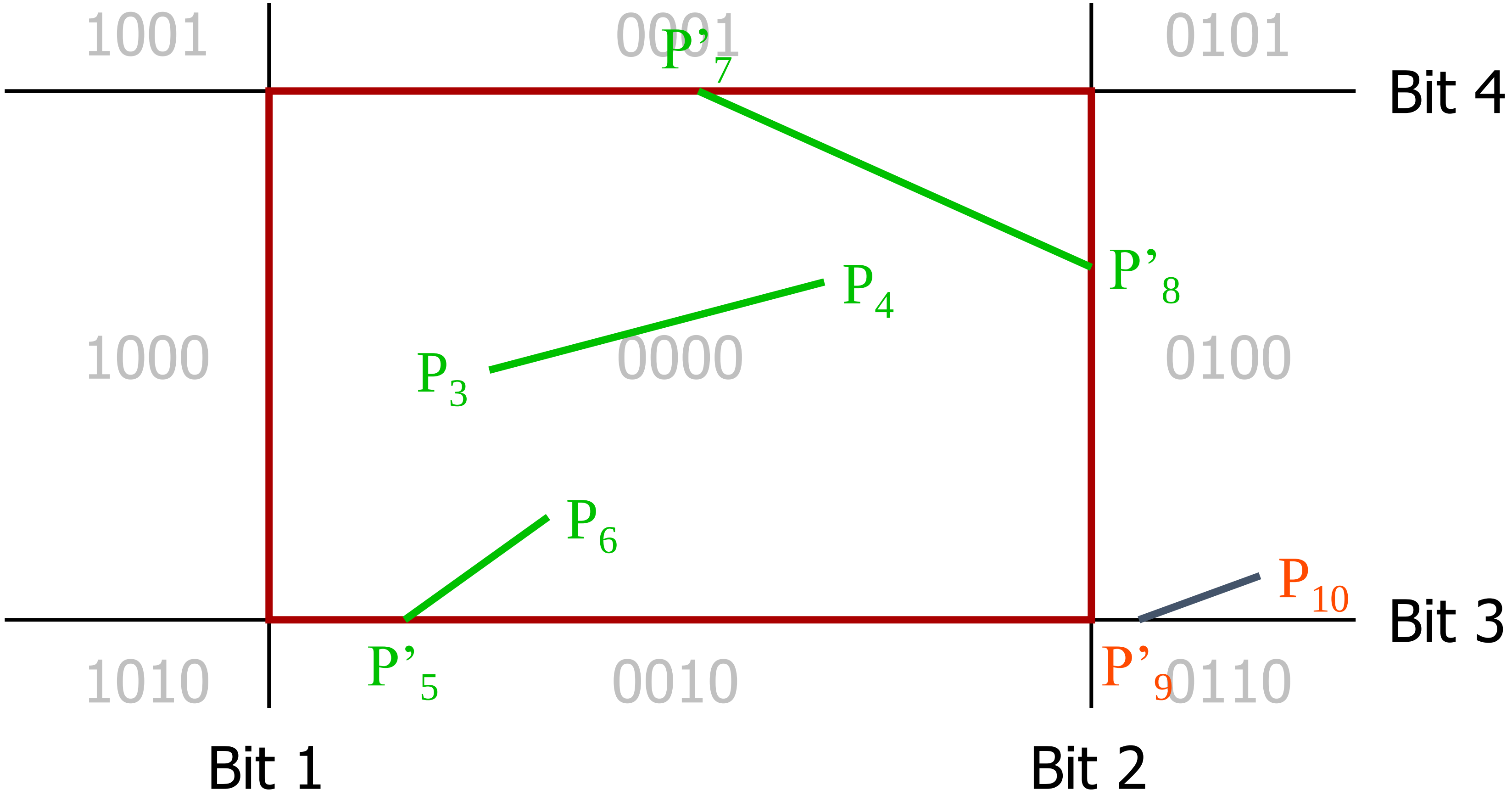

1001
0001
0101
P'7
Bit 4
P'8
P4
1000
P3
0000
0100
P6
P10
Bit 3
P'5
1010
0010
P9
0110
Bit 1
Bit 2

Continue…

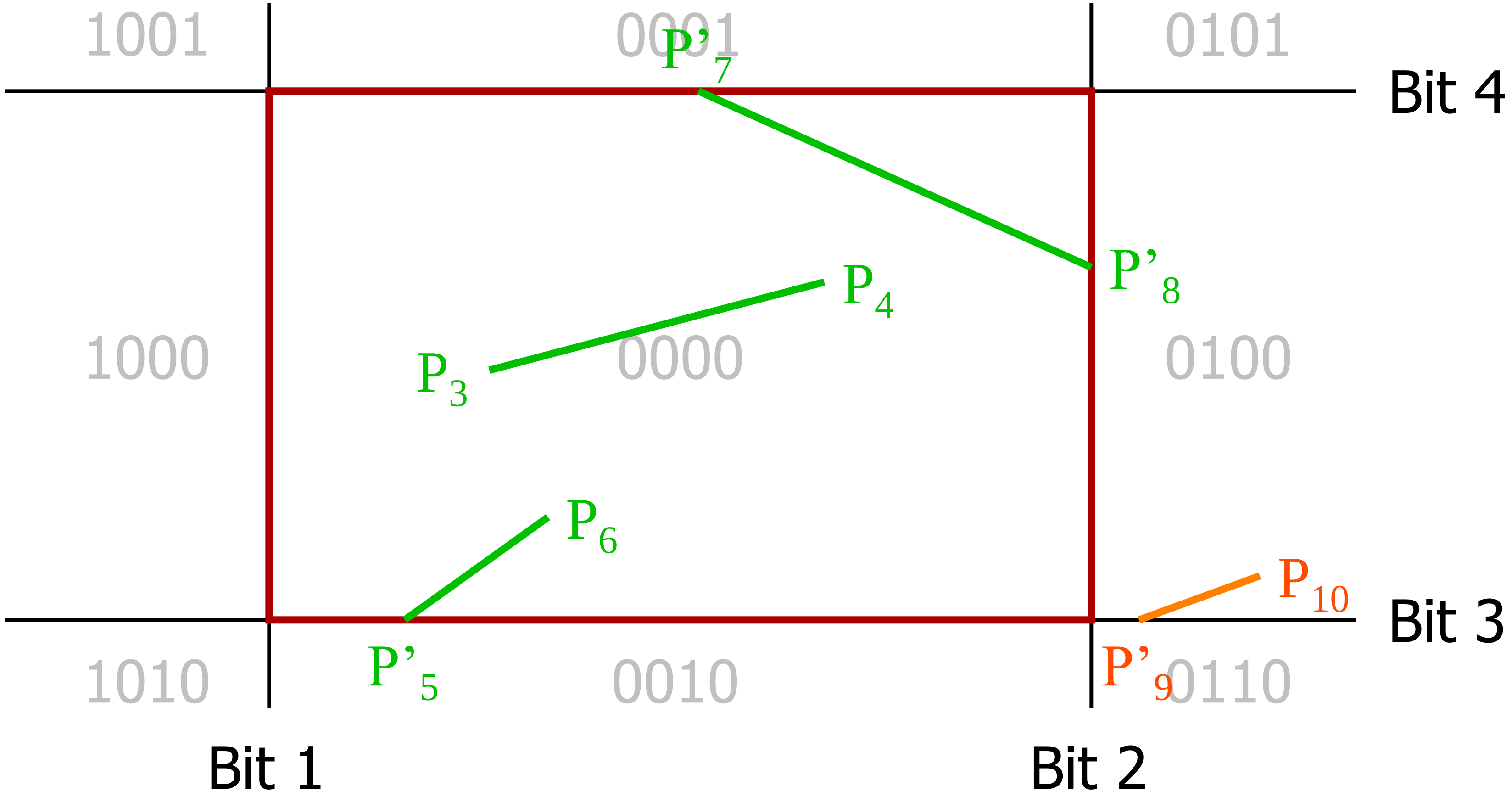


1001
0001
P'$_7$
0101
Bit 4
P'$_8$
P$_4$
1000
P$_3$
0000
0100
P$_6$
P$_{10}$
Bit 3
P'$_5$
1010
0010
P$_9$
0110
Bit 1
Bit 2

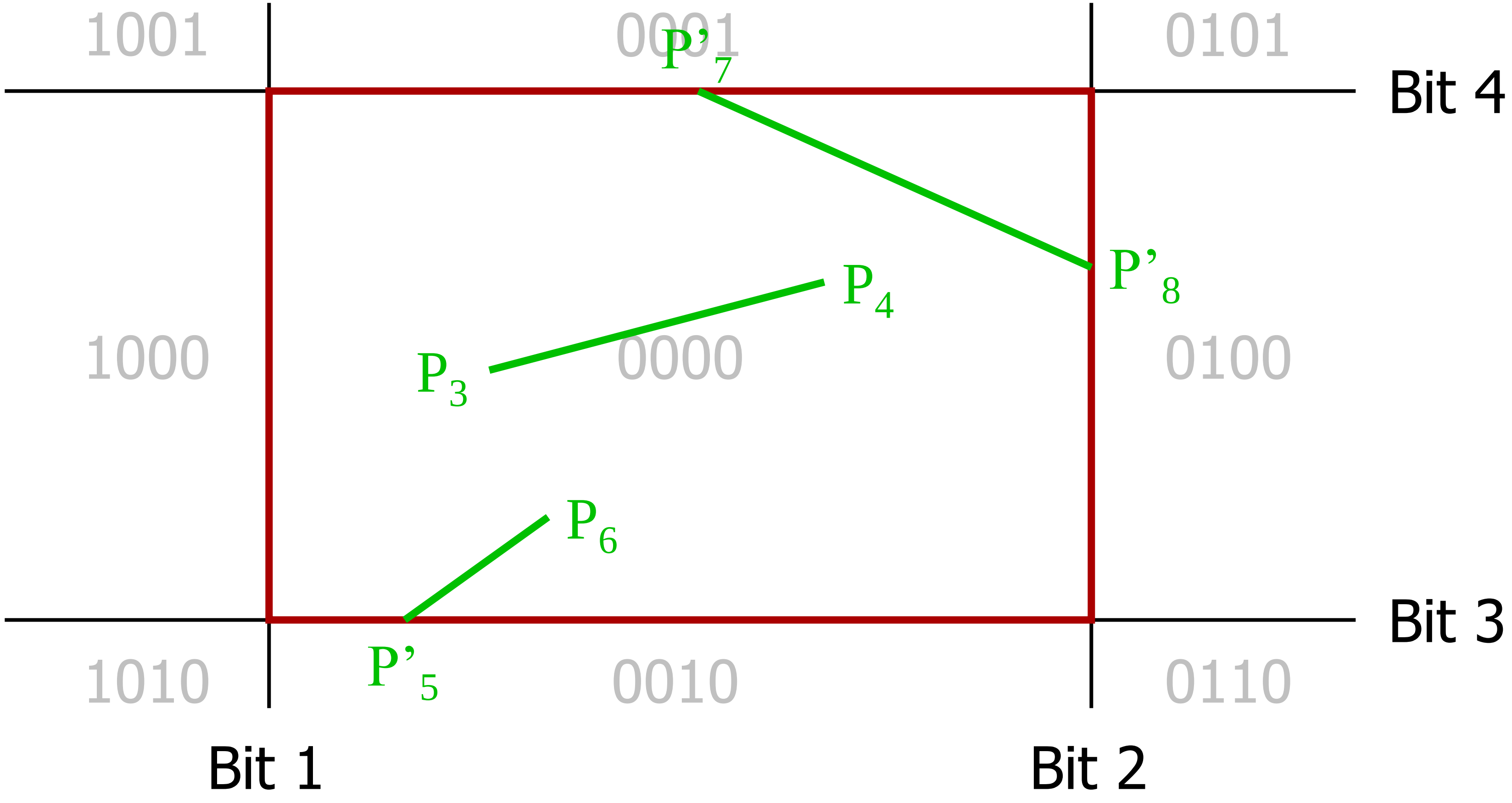
Continue…


1001
0001
0101
P'7
Bit 4
P'8
P4
1000
P3
0000
0100
P6
P10
Bit 3
P'5
1010
0010
P'9
0110
Bit 1
Bit 2

Continue…


1001
0001
0101
Bit 4
P'7
P'8
P4
1000
0000
0100
P3
P6
P10
Bit 3
1010
P'5
0010
P'9
0110
Bit 1
Bit 2

Continue…


1001
0001
0101
P'$_7$
Bit 4
P'$_8$
P$_4$
1000
P$_3$
0000
0100
P$_6$
Bit 3
P'$_5$
1010
0010
0110
Bit 1
Bit 2

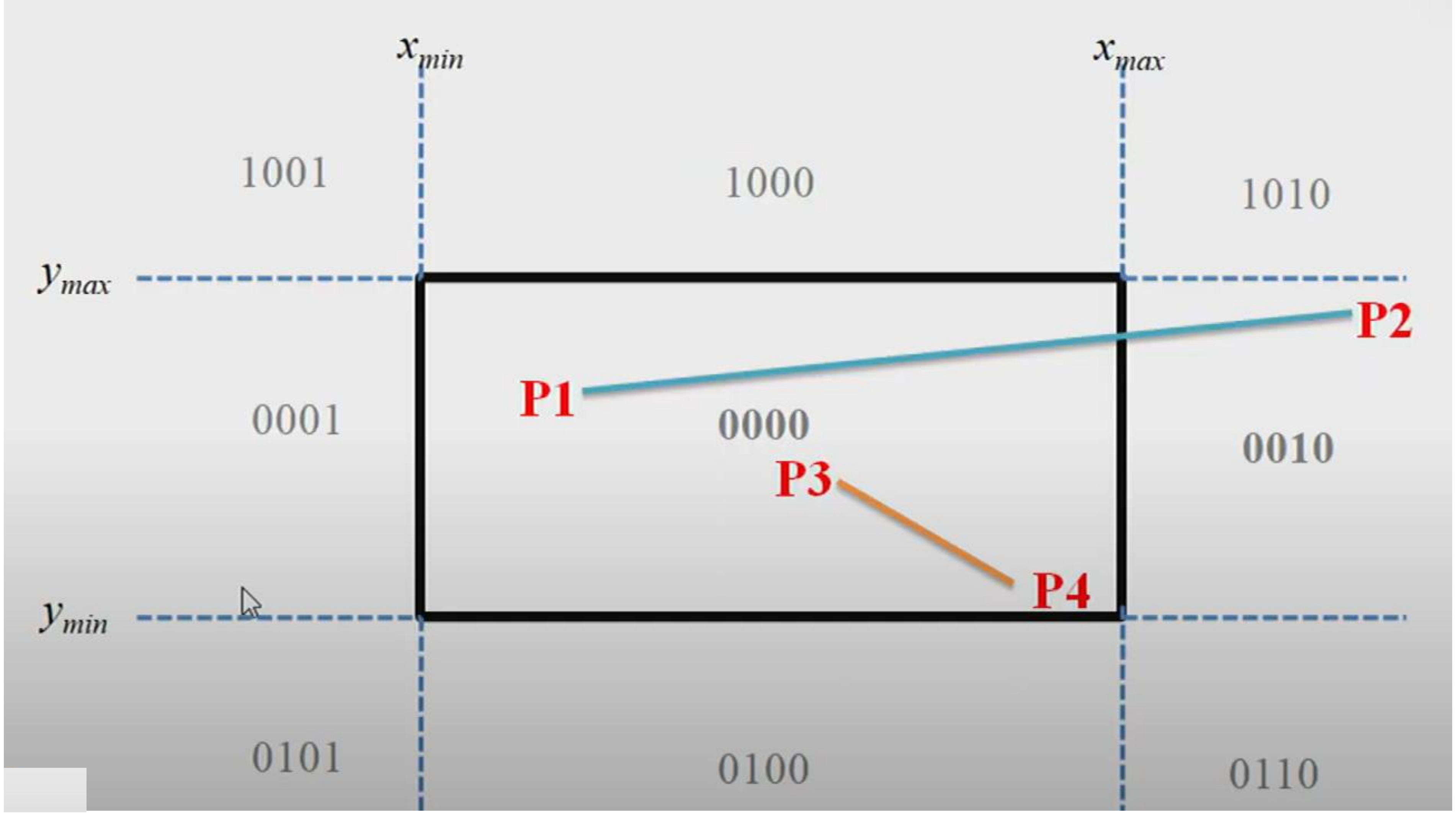
Midpoint Subdivision

- An alternative way to process a line in category 3 is based on binary search.
- The line is divided at its midpoint into two shorter line segments.
- The clipping categories of the two new line segments are determined by their region codes.
- Each segment in category 3 is divided again into shorter segments and categorized.
- This bisection & categorization process continues until each line segment that spans across a window boundary reaches a threshold for line size and all other segments are either in category 1 or in category 2.

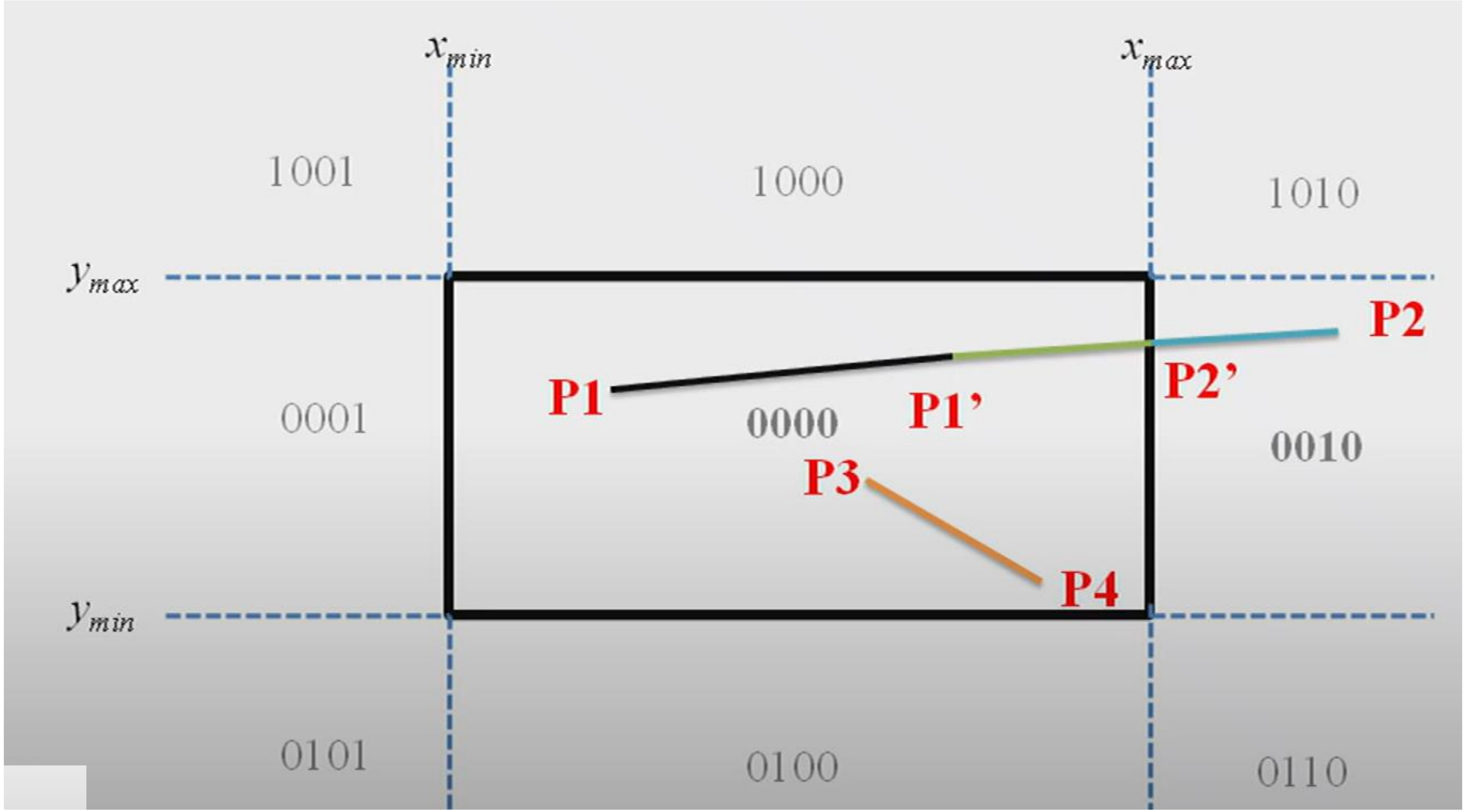
Midpoint Subdivision Algorithm

```
• Step-1: Calculate the position of both endpoints of the line.
• Step-2: Perform OR operation on both of these endpoints.
• Step-3: If the OR operation gives 0000:
      then-
          Line is guaranteed to be visible;
      else-
          Perform AND operation on both endpoints.
          If AND ≠ 0000-
              the line is invisible;
          else
              the line is clipped case;
• Step-4: For the line to be clipped. Find midpoint.
      $X_m=(x_1+x_2)/2$
      $Y_m=(y_1+y_2)/2$
• Step-5: Check each midpoint, whether it nearest to the boundary of a window or not.
• Step-6: If the line is totally visible or totally rejected not found:
      Repeat step 1 to 5.
• Step-7: Stop algorithm.
```

Example - 01


x_{min}
x_{max}
y_{max}
y_{min}
1001
1000
1010
0001
0000
0010
0101
0100
0110
P1
P2
P3
P4

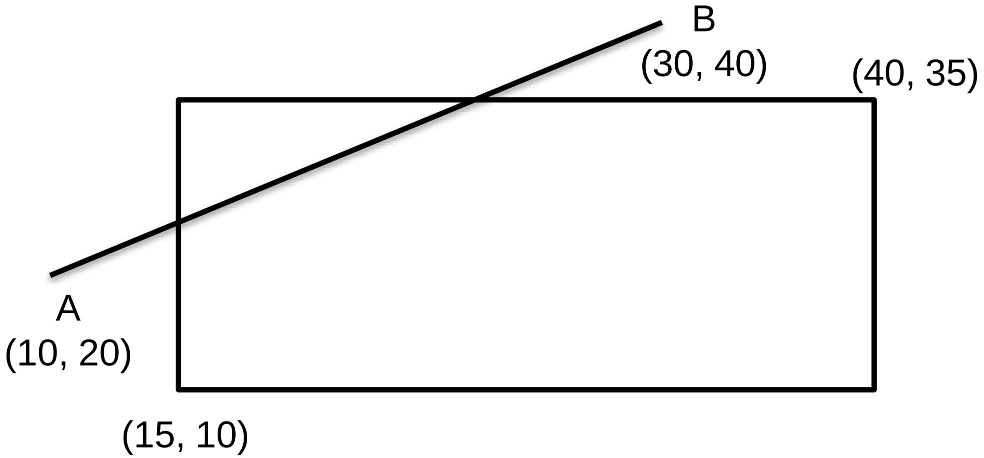
Continue…



x_{min}
x_{max}
1001
1000
1010
y_{max}
P2
P1
P2'
0001
0000
P1'
0010
P3
P4
y_{min}
0101
0100
0110

Example - 02

- Window size is (15, 10) to (40, 35). A line AB is given having co-ordinates of A (10, 20) and B (30, 40). Find the visible portion of the line using midpoint subdivision.



Practice

1) Use the Cohen Sutherland algorithm to clip line P1 (70,20) and p2(100,10) against a window lower left hand corner (50,10) and upper right hand corner (80,40).
2) Window size is (-3, 1) to (2, 6). A line AB is given having co-ordinates of A (-4, 2) and B (-1, 7). Find the visible portion of the line using midpoint subdivision and Cohen Sutherland algorithm.

Thank you!