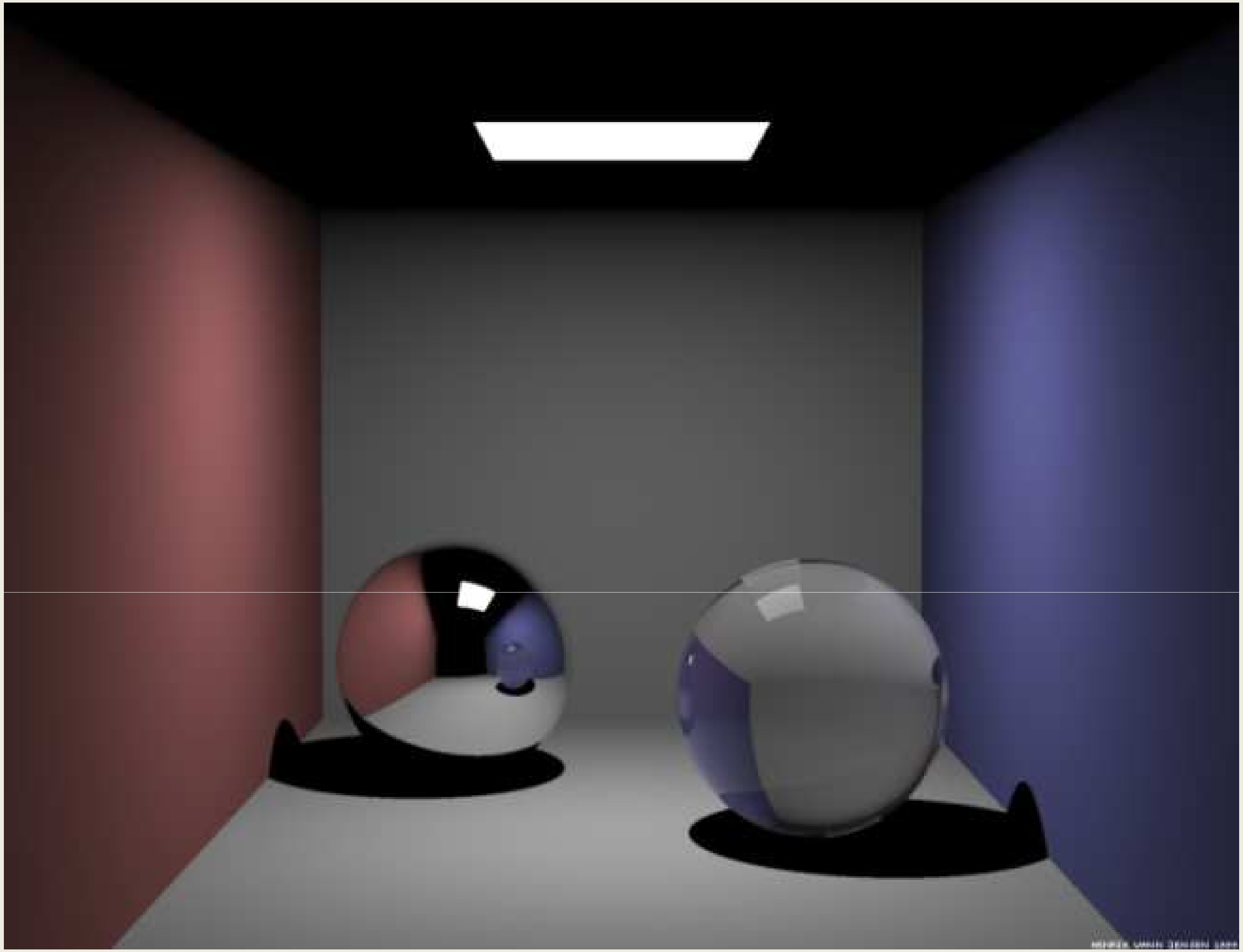


Department of Computer Science and Engineering,
University of Dhaka

# Computer Graphics (CSE 4103)

# Lecture 6

## Basic Raster Graphics Algorithm for Drawing 2D Primitives

## Clipping Line

# Last Classes

1. *Incremental Line-drawing Algorithm*
2. *Mid-point Algorithm for Line-drawing*
    - *i) In only one Octant*
    - *ii) In all 8 octants*
    - *iii) 8-way Symmetry*
    - *iv) Tangent Independent mid-point line drawing algorithm*
3. *Mid-point Algorithm for drawing Circle*
4. *Mid-point Algorithm for drawing Ellipse*

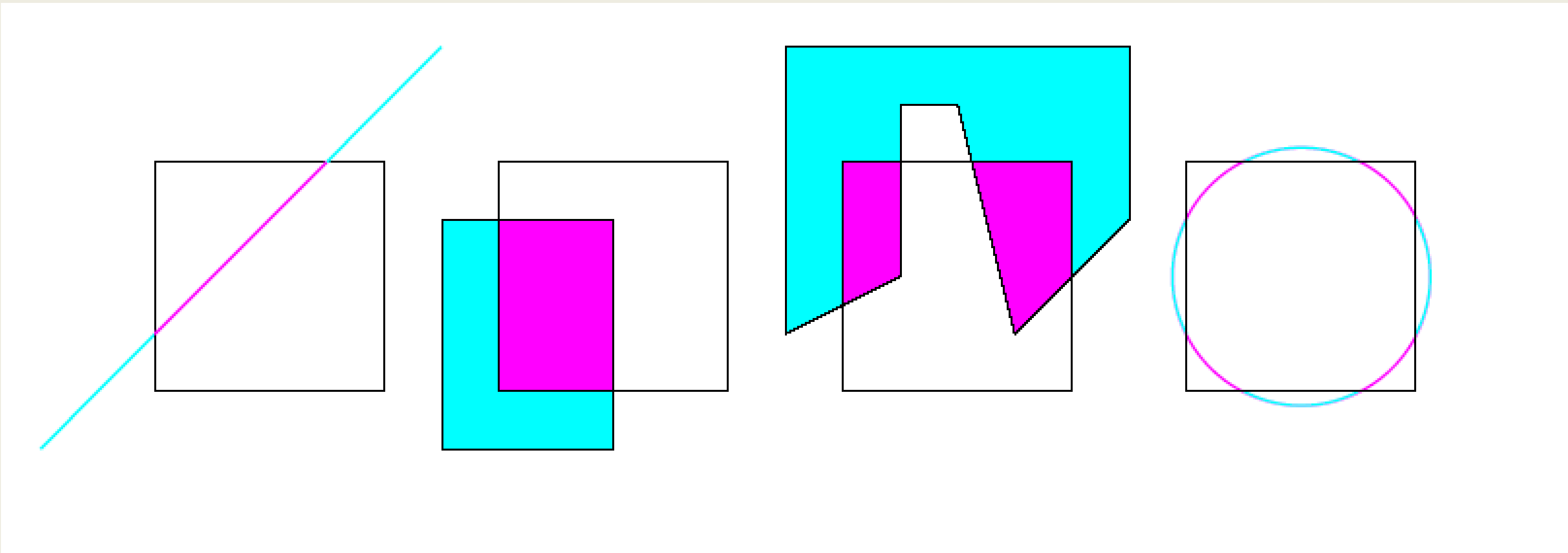
# This Class

1. *Clipping lines using*
   a) *Cohen-Sutherland Algorithm*
   b) *Cyrus-Bech Algorithm*

# What is Clipping

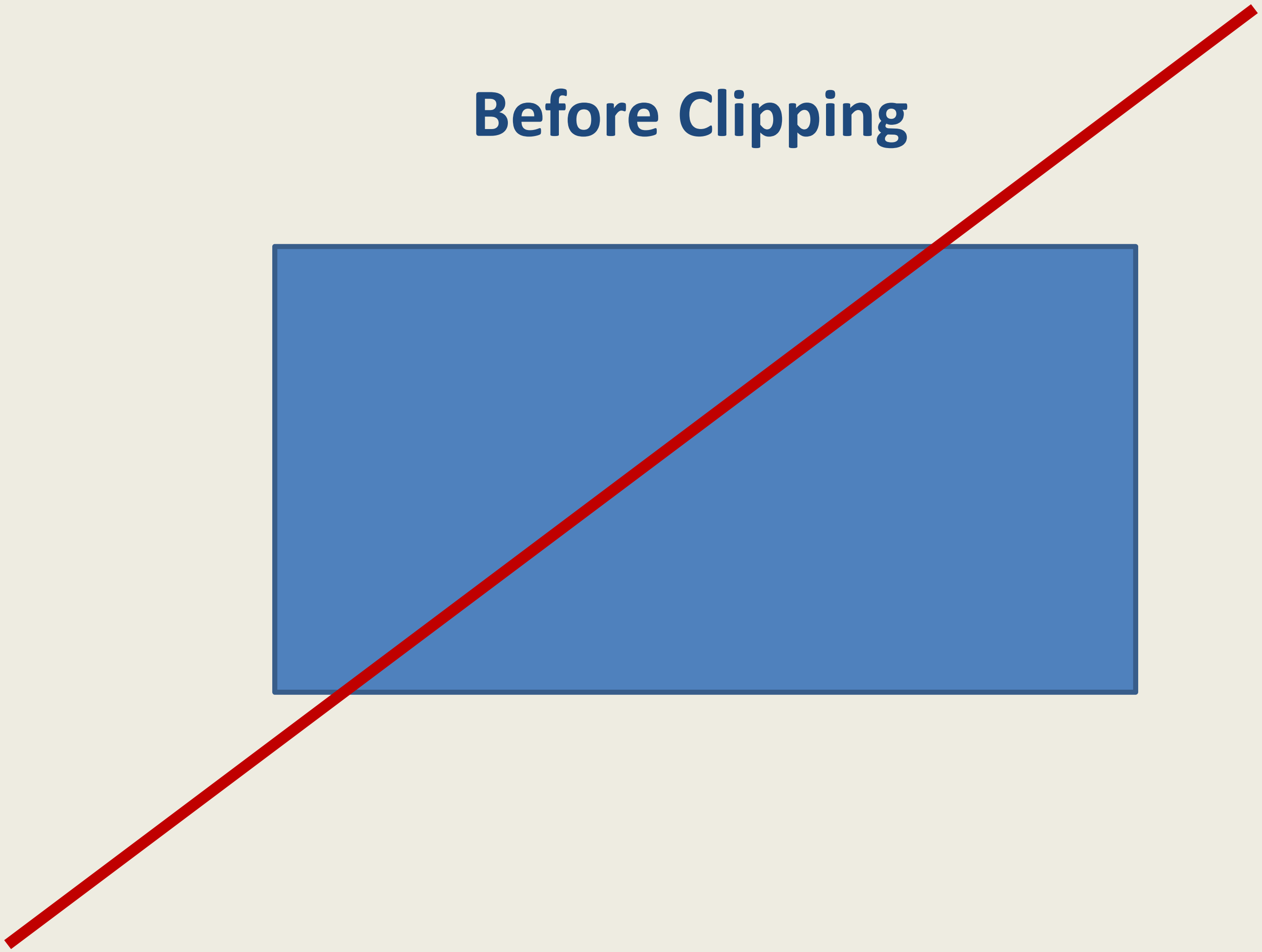
**From Computer Graphic's context, Clipping** "is a method to selectively enable or disable rendering operations within a defined region of interest"[Wiki].

In general, Clipping is a technique to trimming down a given primitive with respect to an object.



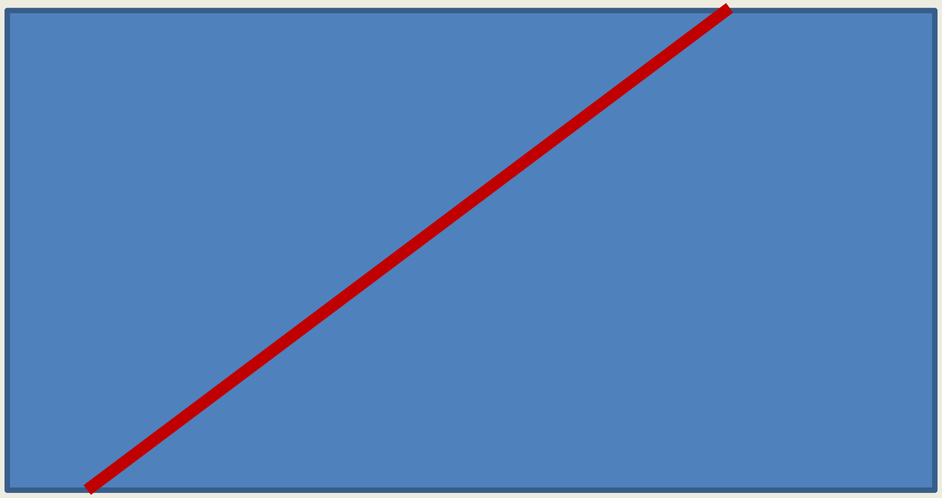
# What is Clipping..

**Before Clipping**

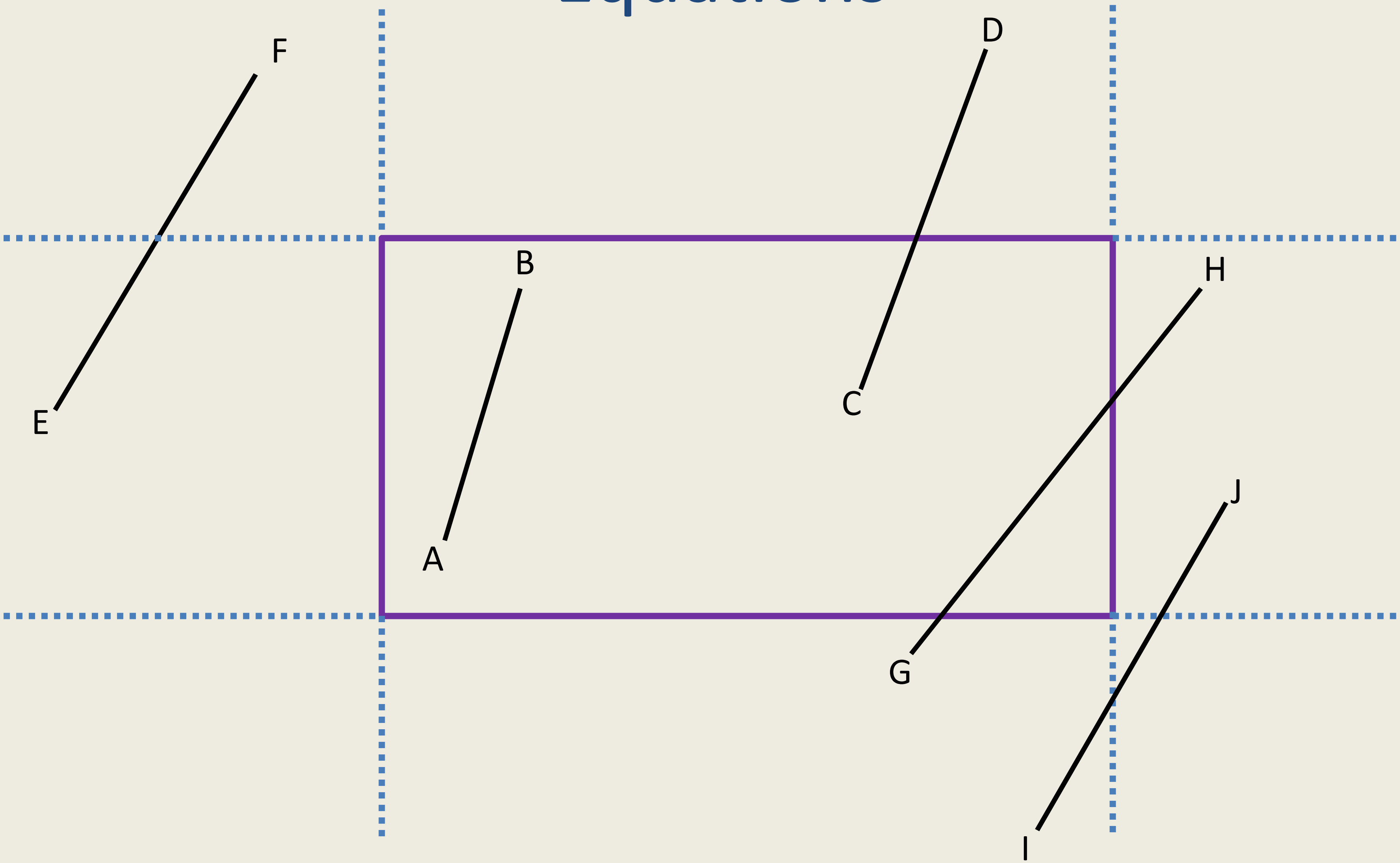


# What is Clipping..

**After Clipping**



# Clipping Lines by Solving Simultaneous Equations



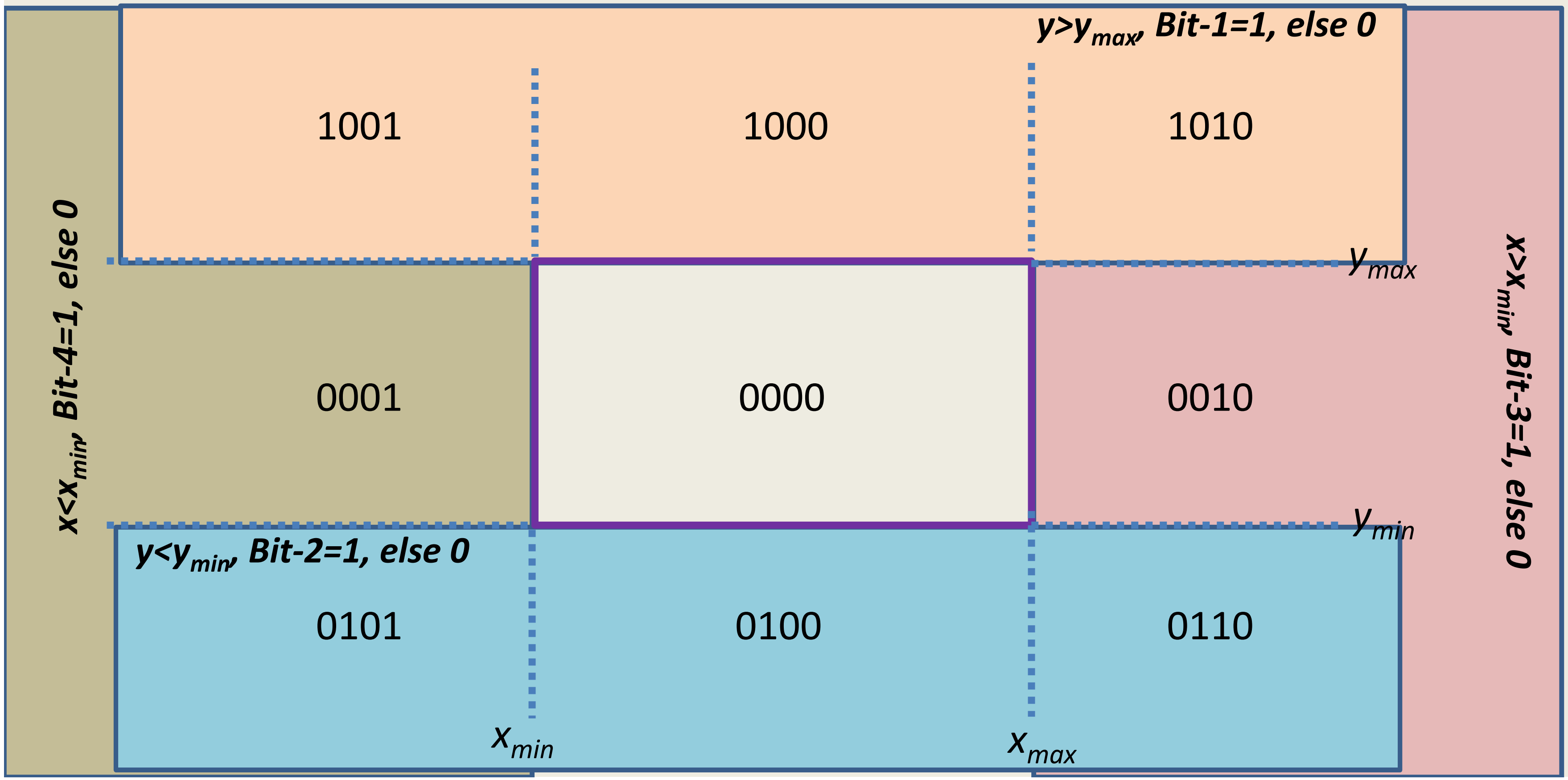
# Clipping Lines by Solving Simultaneous Equations..

**Input:** Coordinates of Lines, Four corners of rectangle $\{(x_{min}, y_{min}) (x_{max}, y_{max})\}$
**Output:** Clipped lines with respect to given rectangle

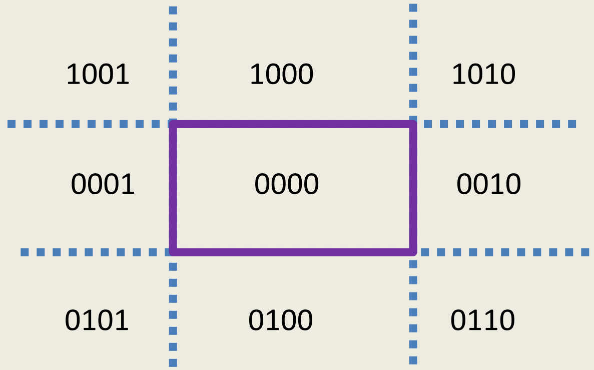
Step 1. **For** every line segment
Step 2: **If** the line is trivially accepted, do nothing
Step 3: **else** convert the line segment to equation of line (EL)
Step 4. consider the edges to equation of lines (EE)
Step 5. consider every pair or $(EL, EE_j)$
Step 6. find the intersecting point between $(EL, EE_j)$
Step 7. validate the intersecting point with respect to $\{(x_{min}, y_{min}) (x_{max}, y_{max})\}$

# Cohen-Sutherland Algorithm

- Divide viewplane into regions defined by viewport edges
- Assign each region a 4-bit outcode:

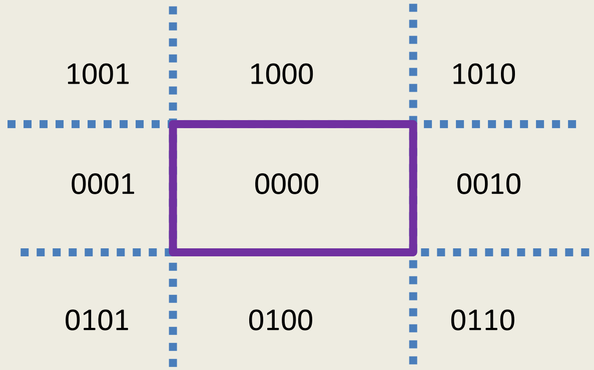


# Cohen-Sutherland Algorithm..



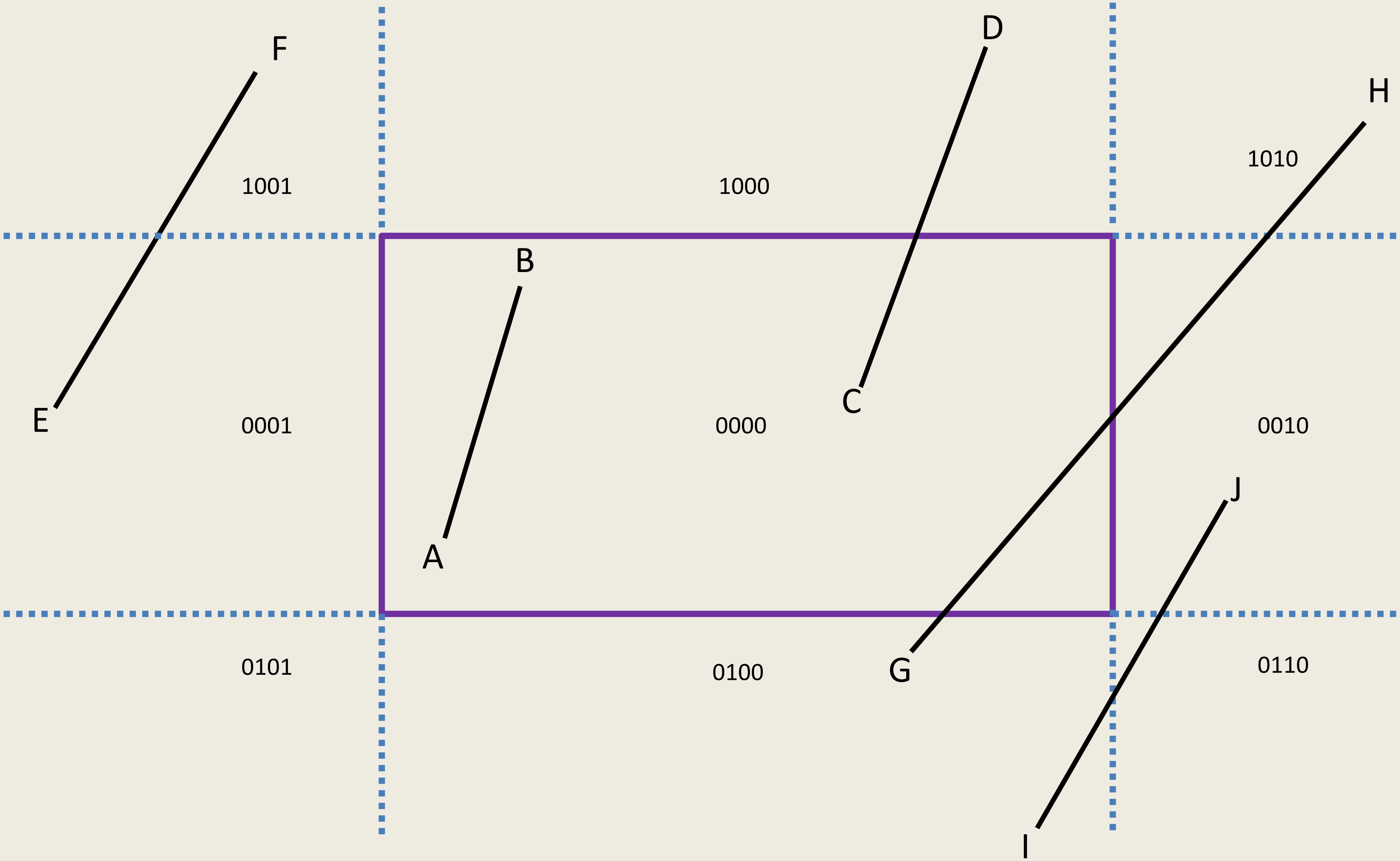

1. Check for trivial accept (both endpoints are in, i.e., outcode for both endpoints are 0000)
2. Check whether it line segment is trivially reject
   - (Trivially reject condition: Bit-wise AND of two outcode should be non-zero)
3. If line can be neither trivially accepted or trivially rejected, then split in two at a clip edge and iterate the same process.

# Cohen-Sutherland Algorithm..

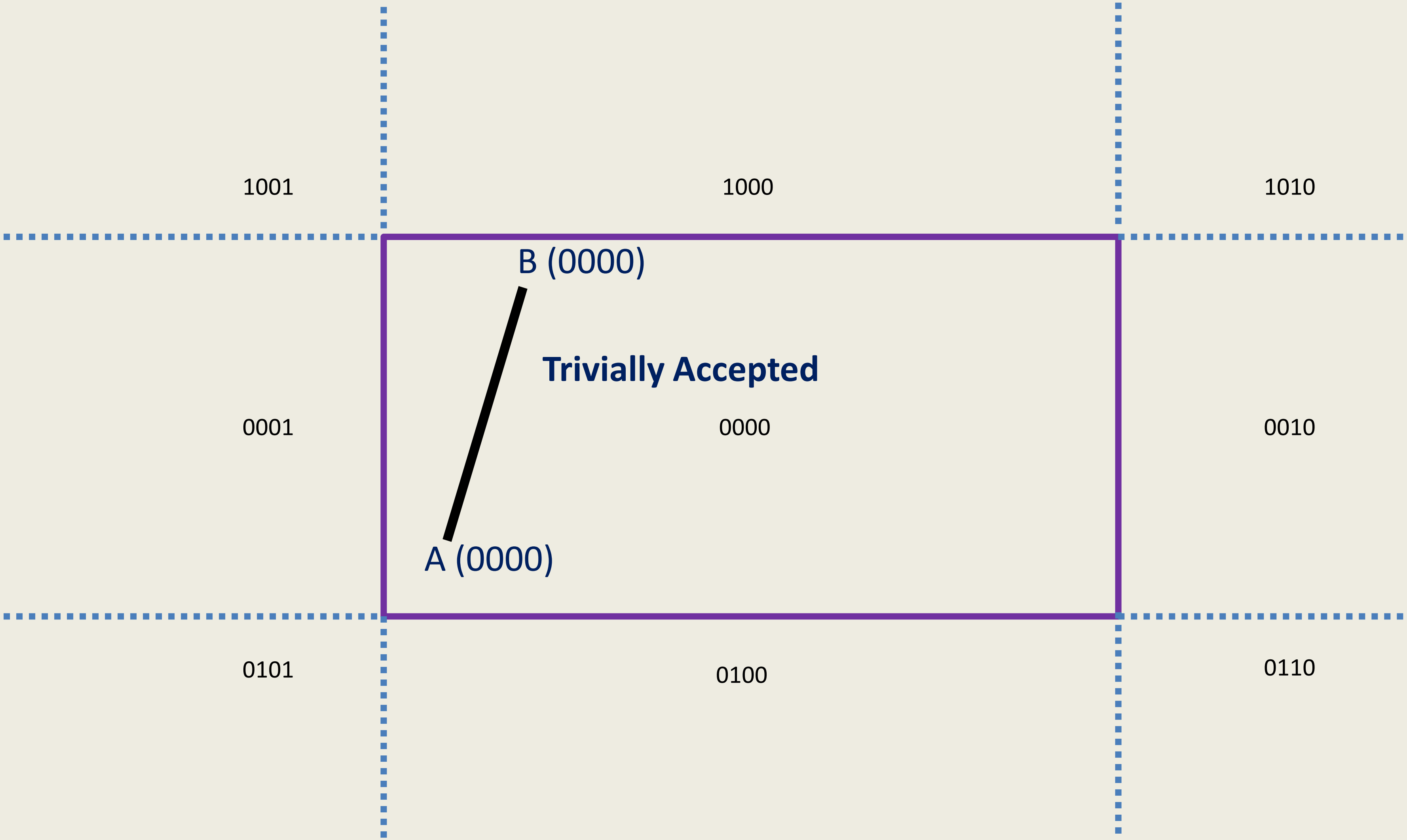



1. Check for trivial accept (both endpoints are in, i.e., outcode for both endpoints are 0000)
2. Check whether it line segment is trivially reject
   - (Trivially reject condition: Bit-wise AND of two outcode should be non-zero)
3. If line can be neither trivially accepted or trivially rejected, then split in two at a clip edge and iterate the same process.

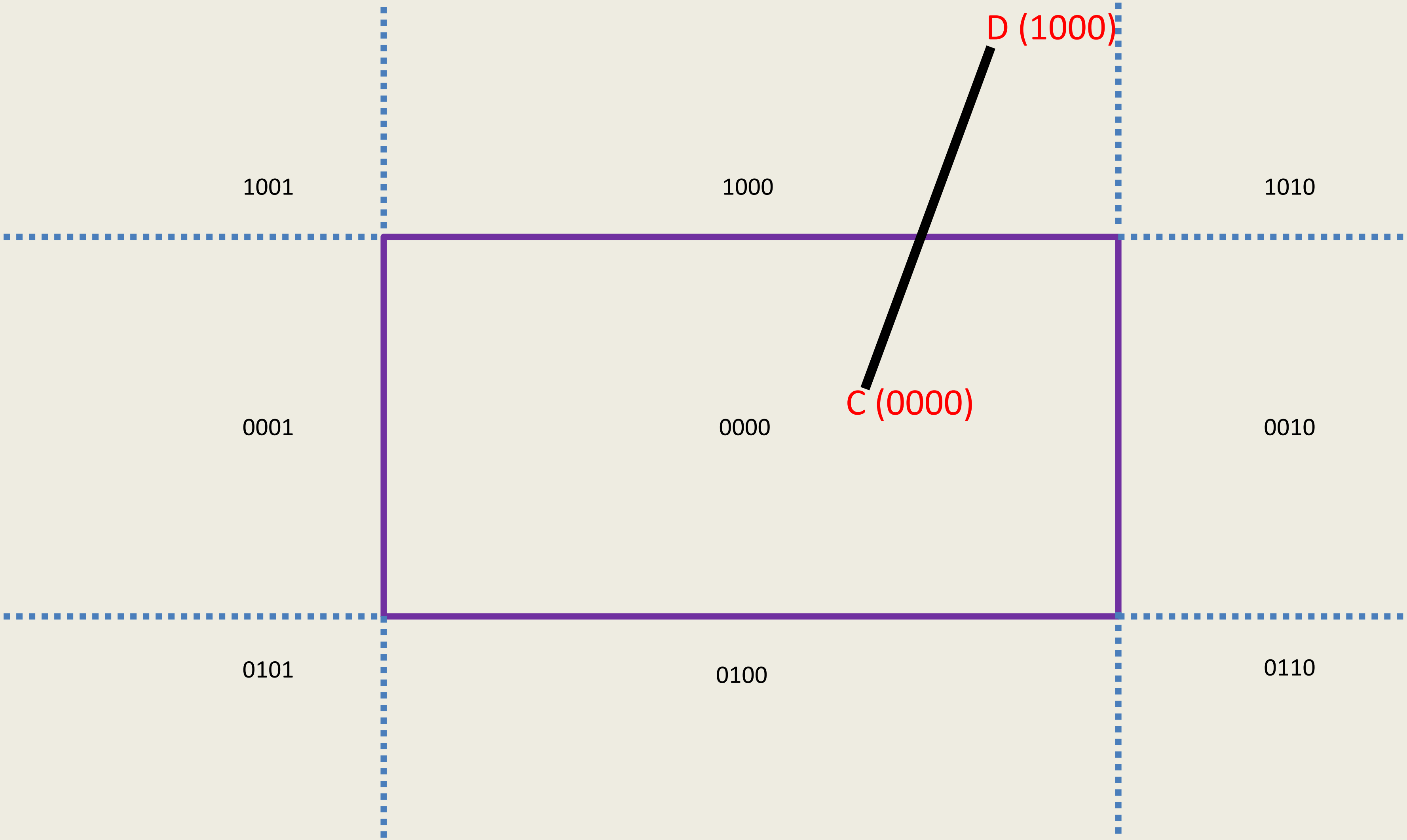
# Cohen-Sutherland Algorithm..



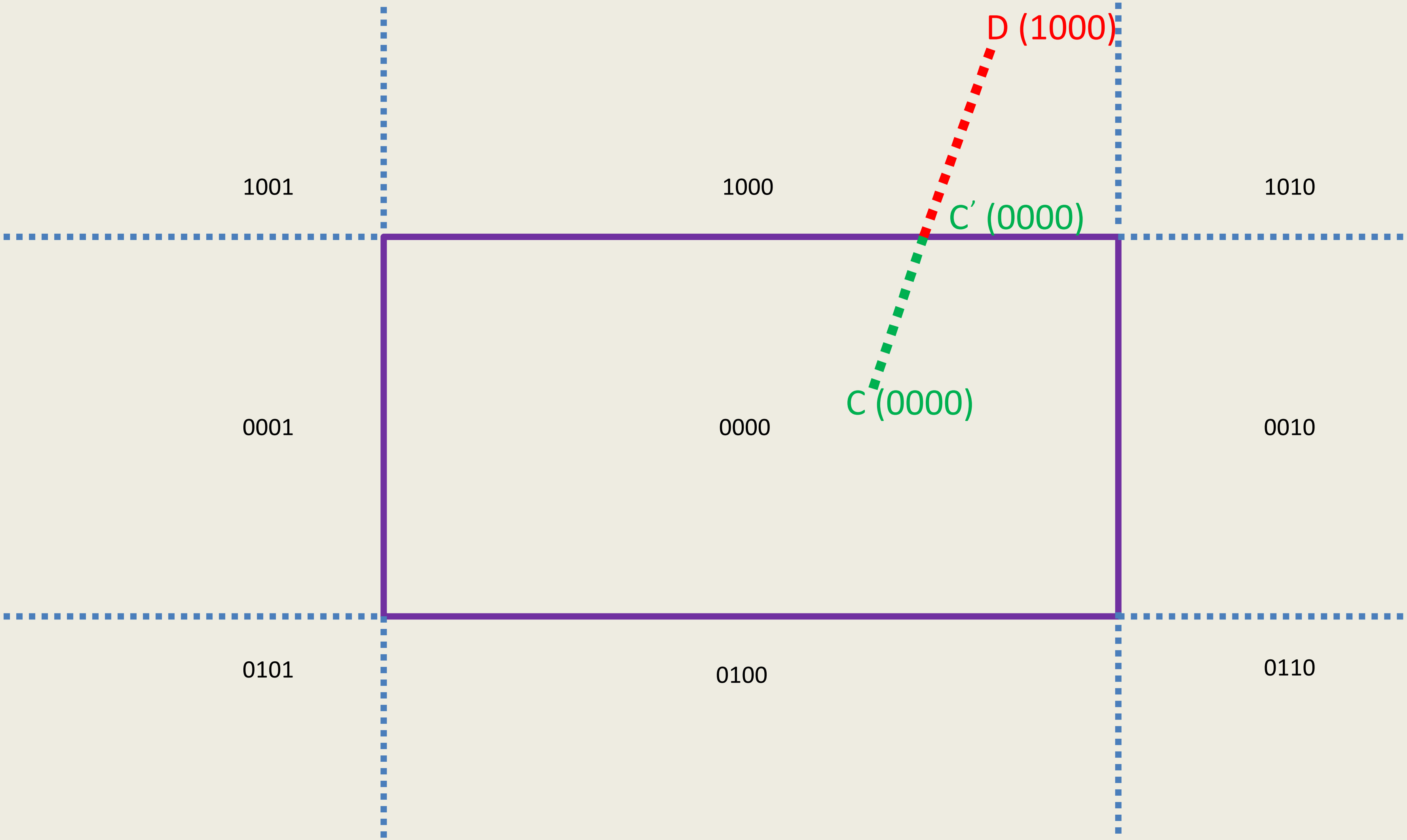
# Cohen-Sutherland Algorithm..



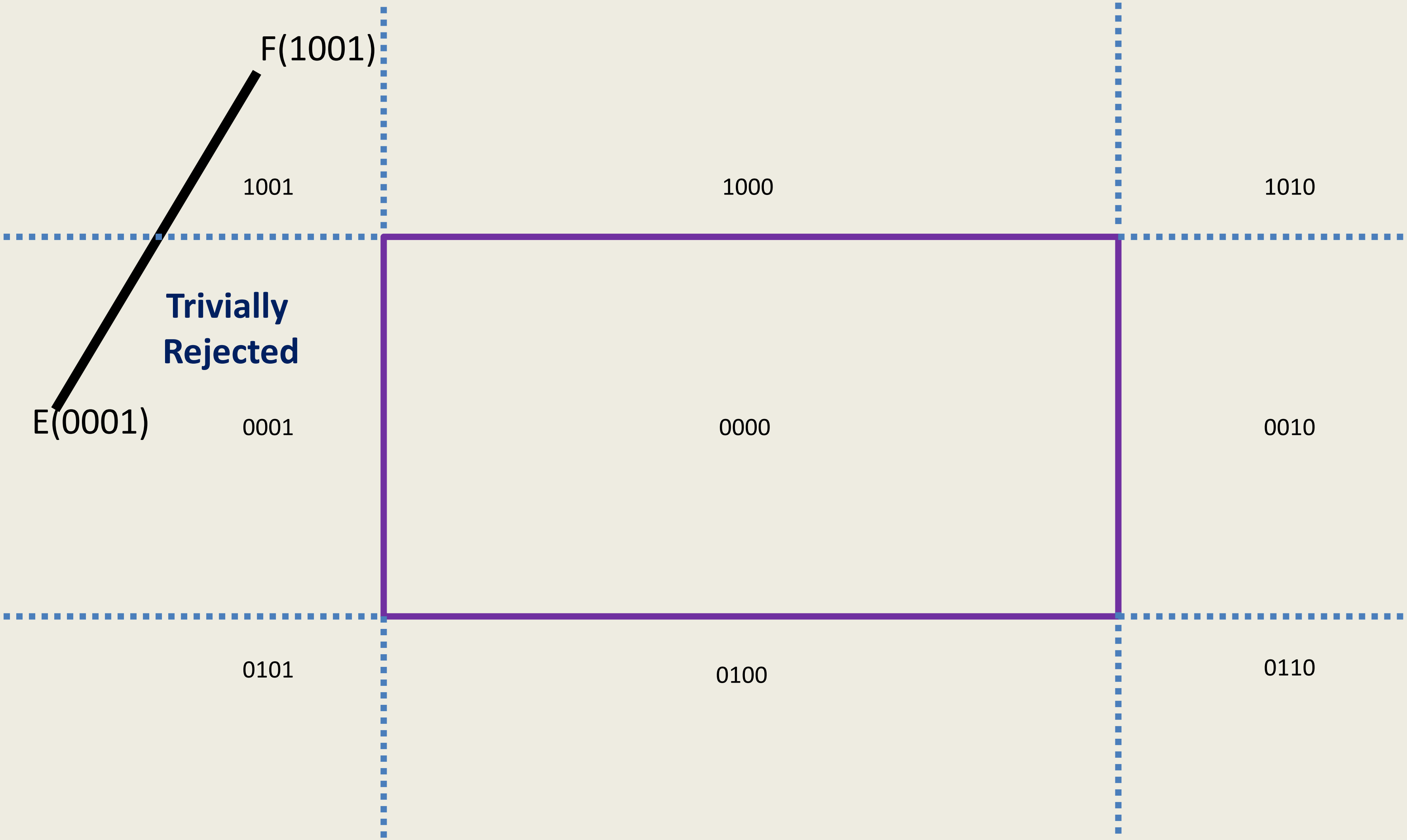
# Cohen-Sutherland Algorithm..



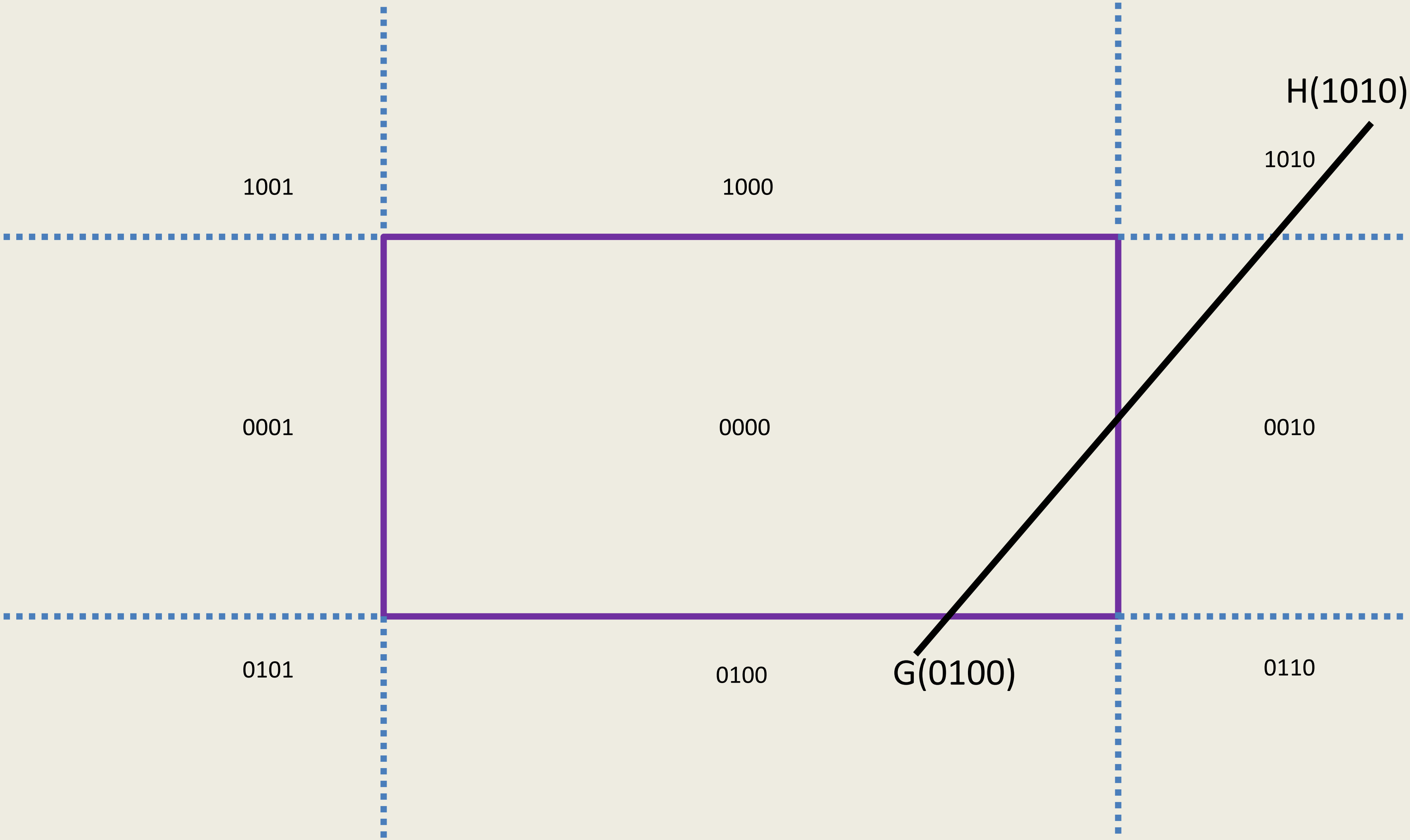
# Cohen-Sutherland Algorithm..



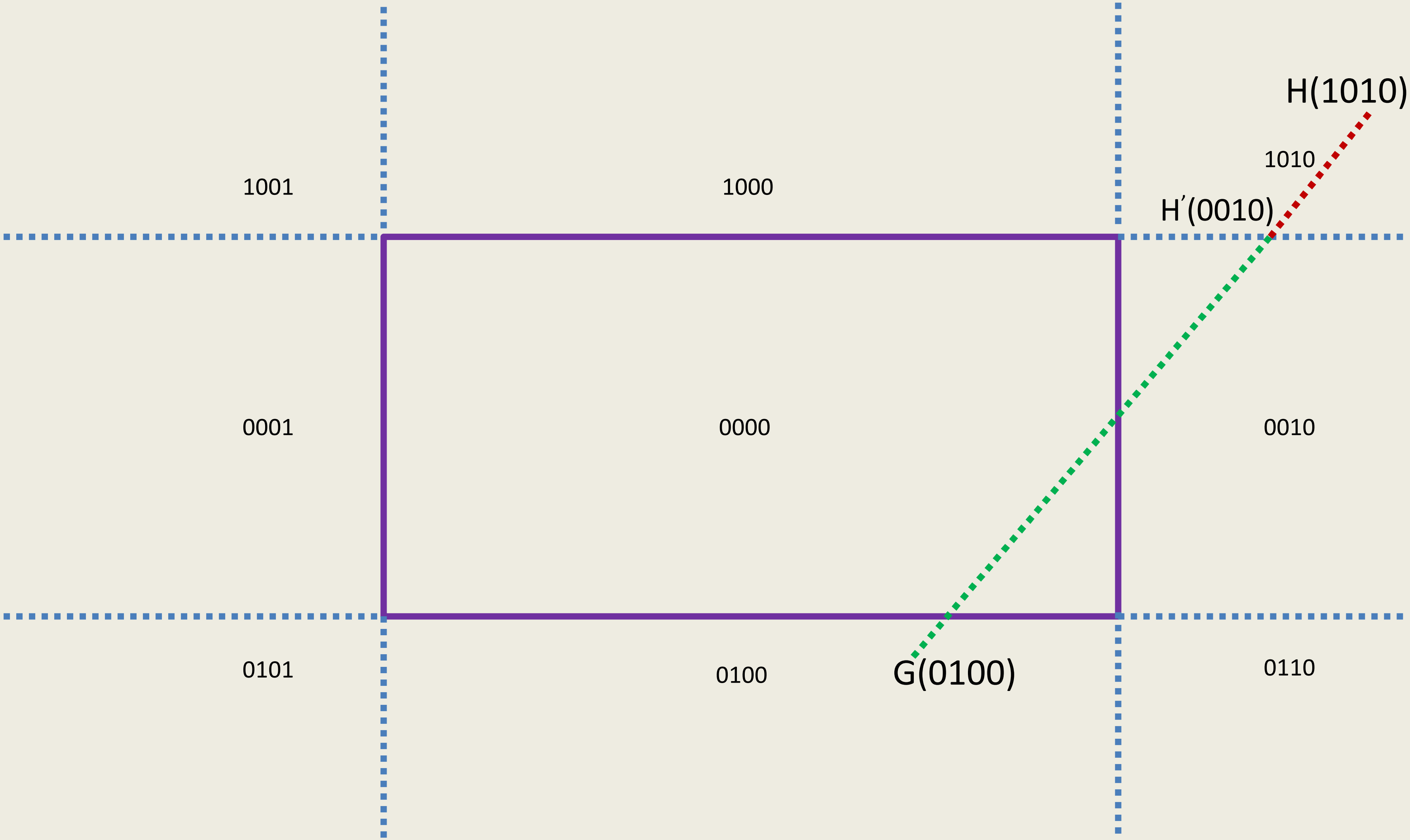
# Cohen-Sutherland Algorithm..



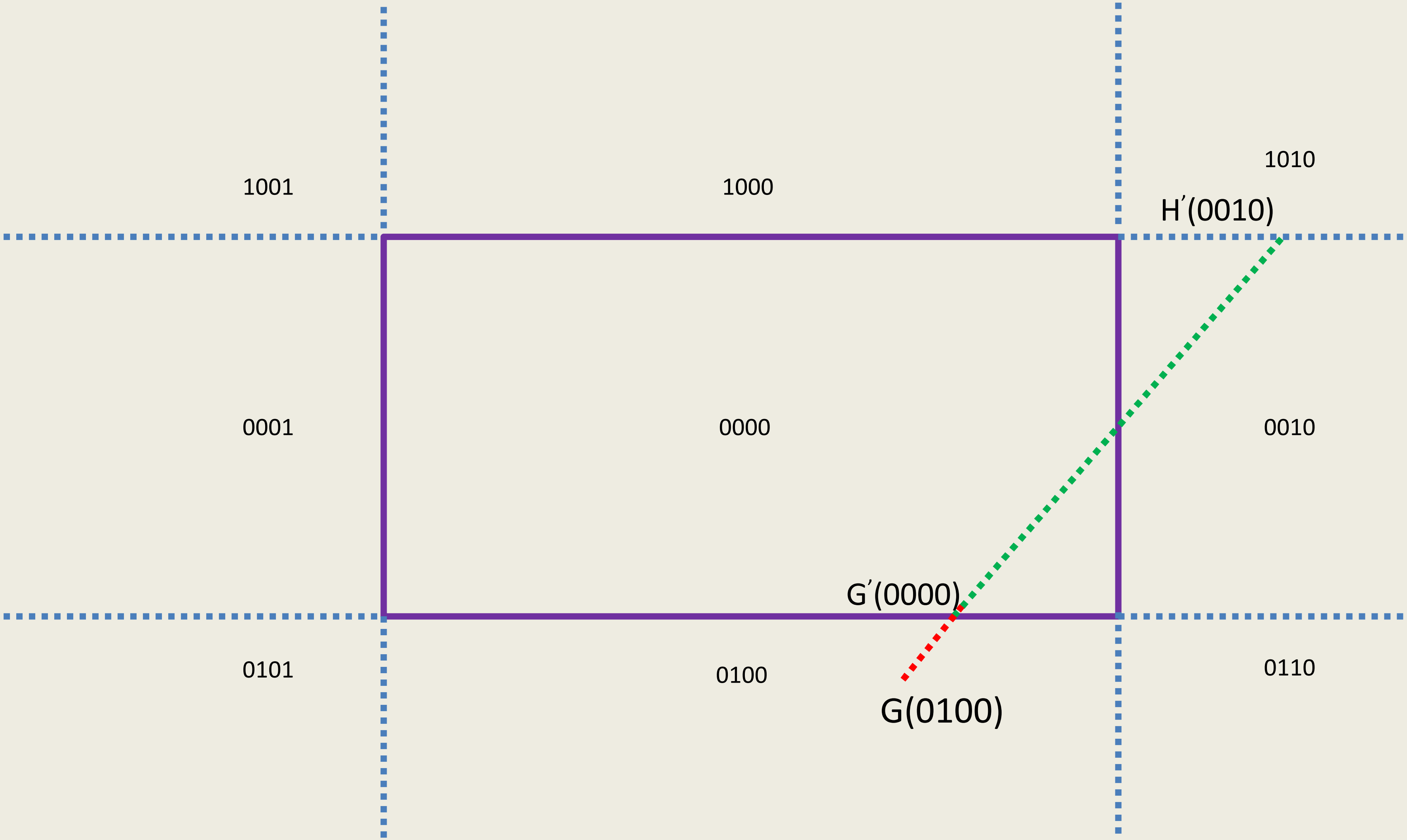
# Cohen-Sutherland Algorithm..



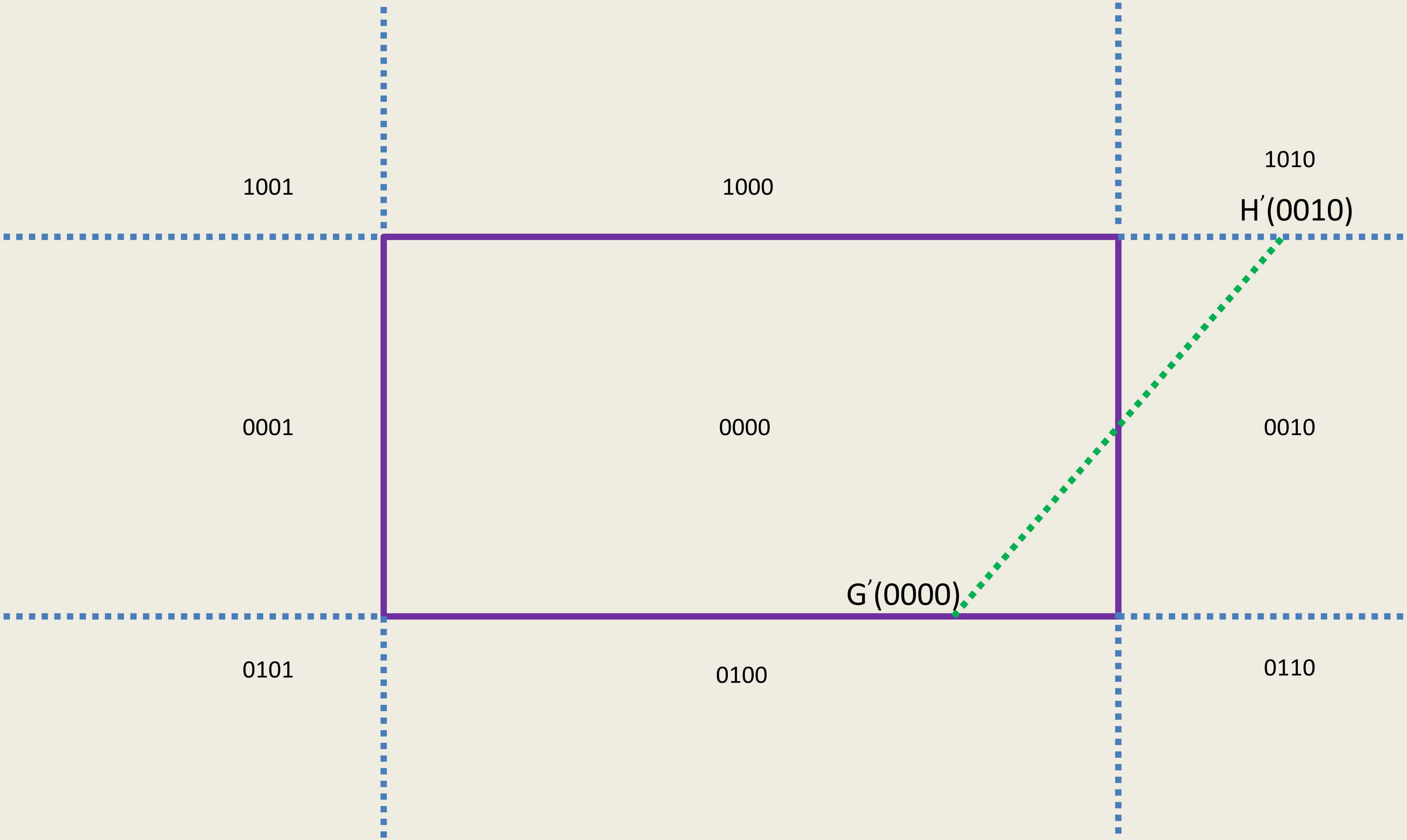
# Cohen-Sutherland Algorithm..



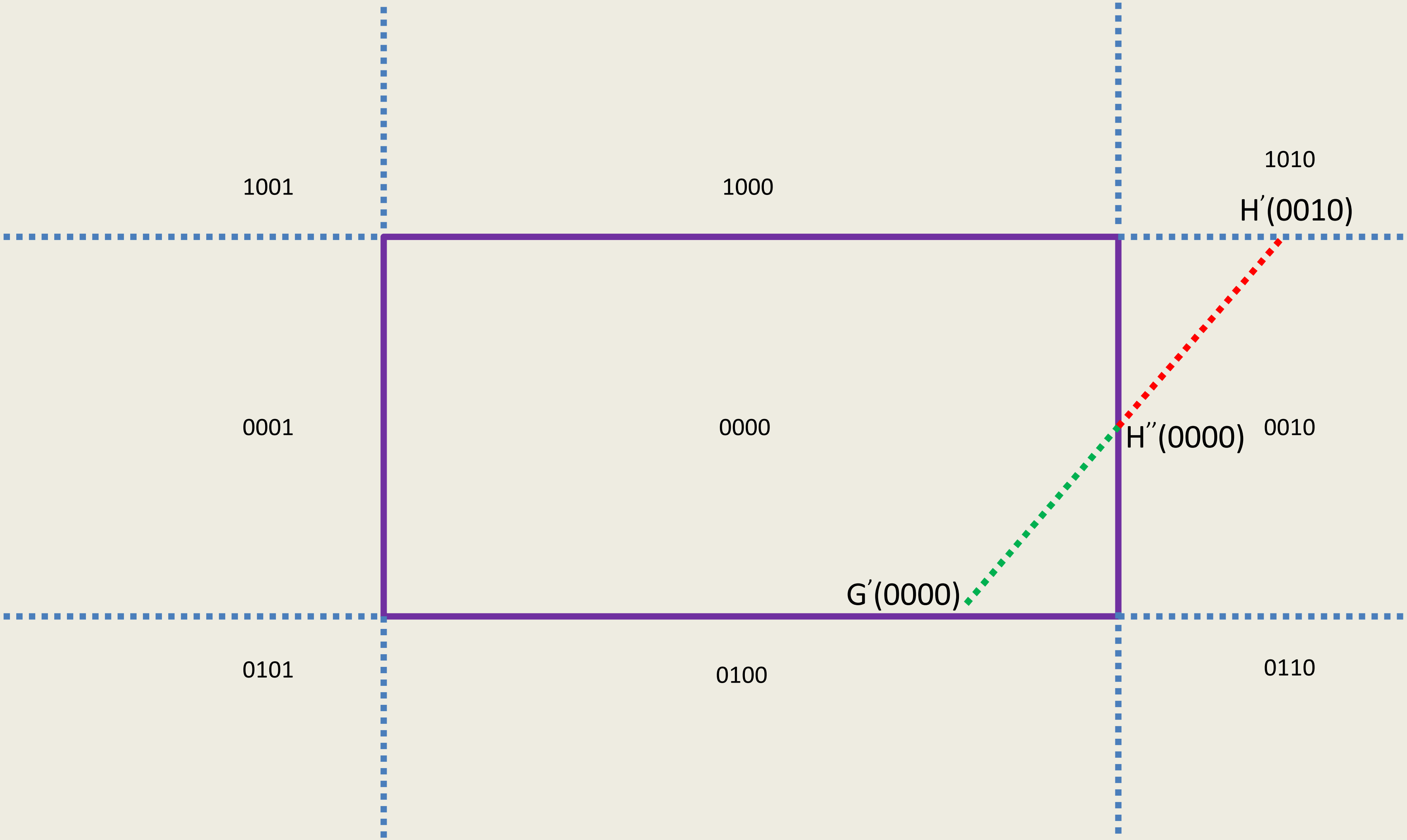
# Cohen-Sutherland Algorithm..



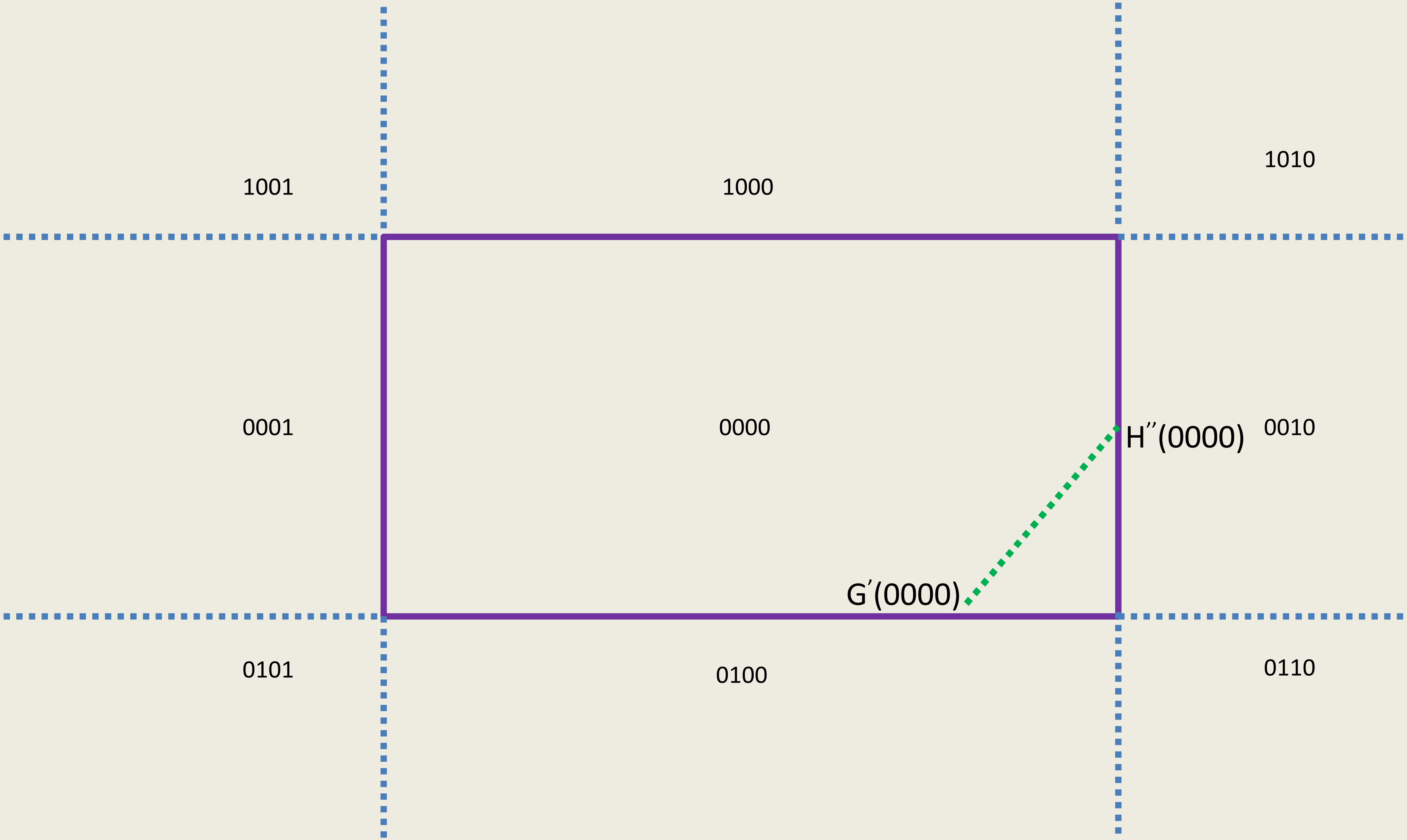
# Cohen-Sutherland Algorithm..



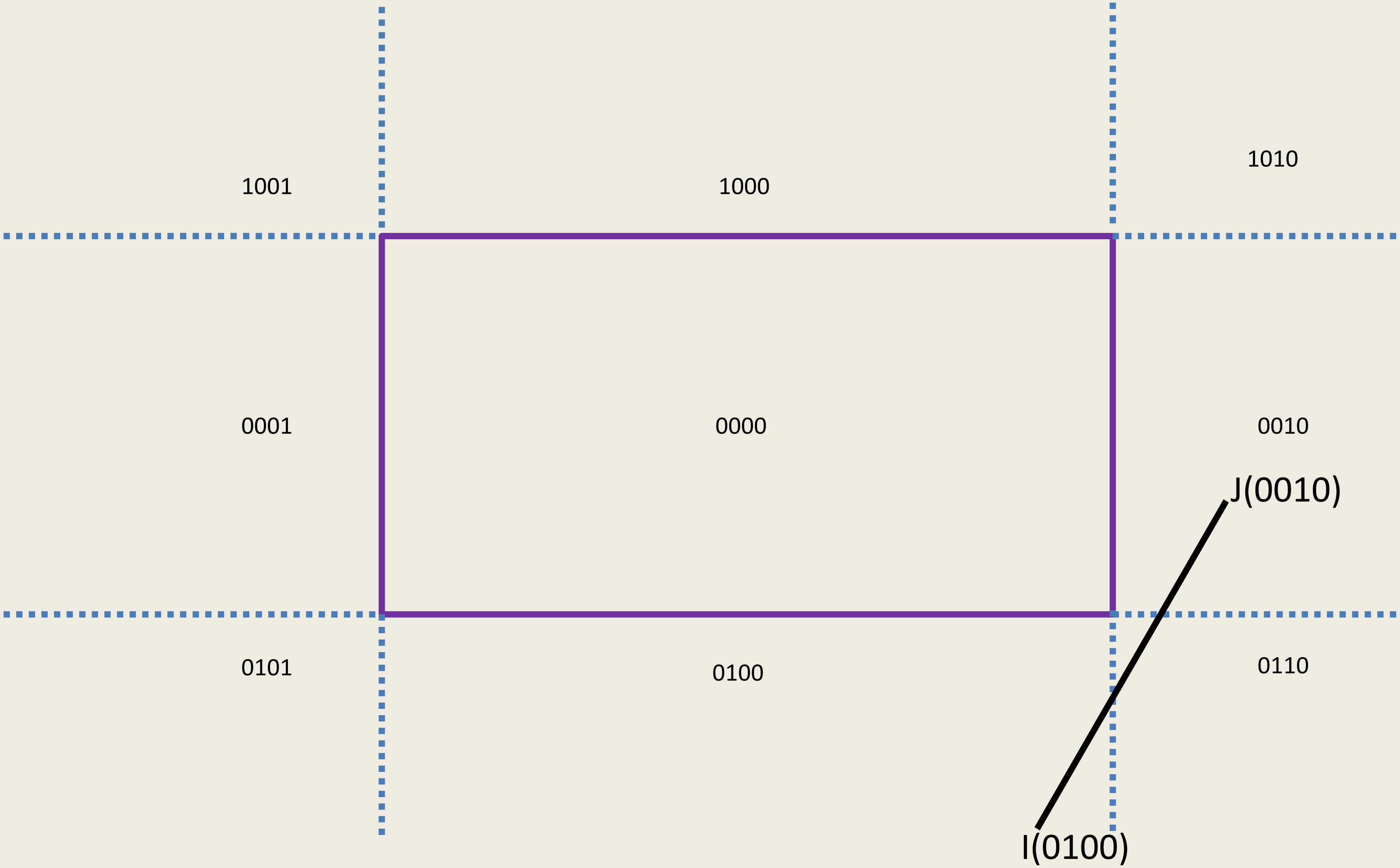
# Cohen-Sutherland Algorithm..



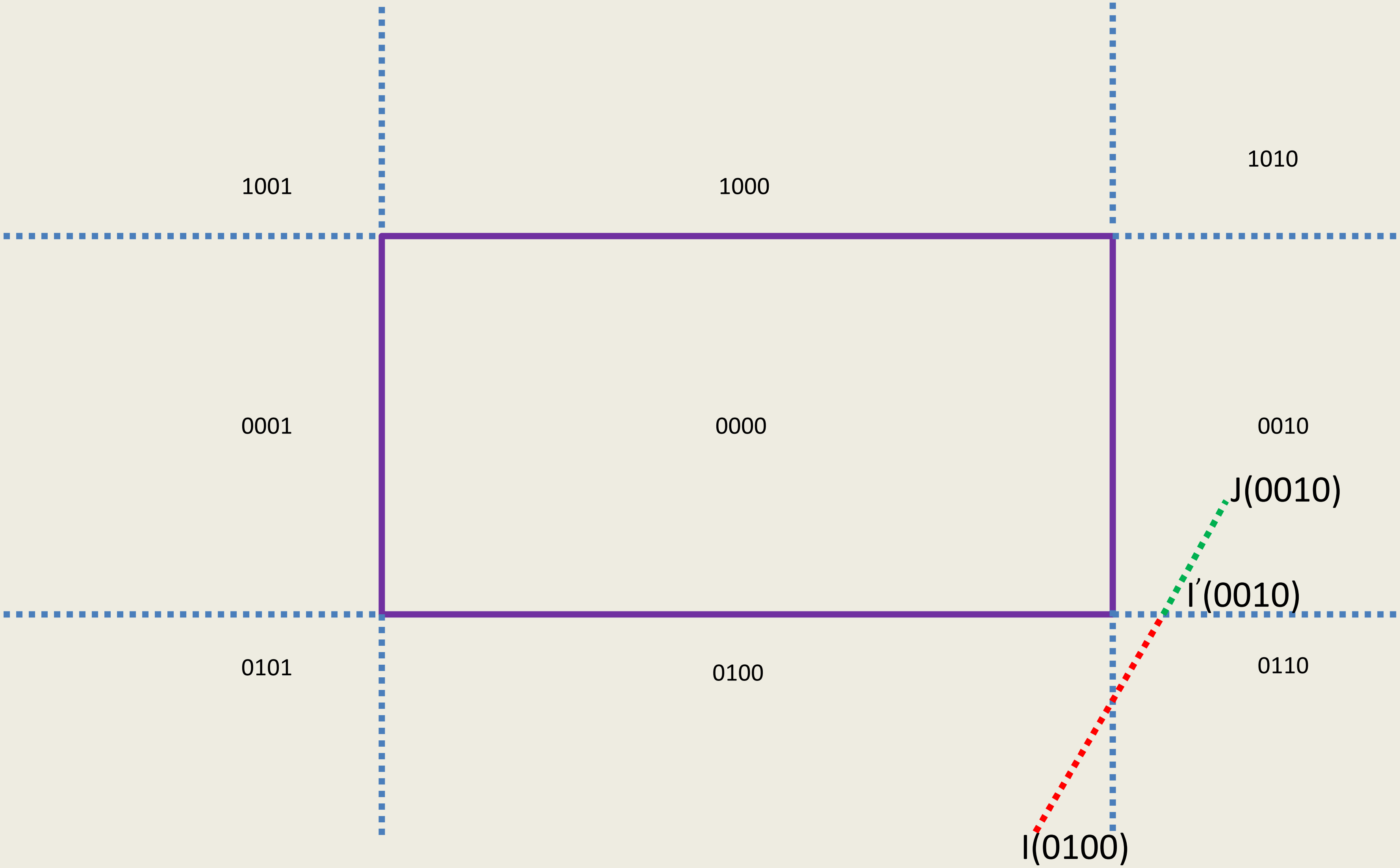
# Cohen-Sutherland Algorithm..



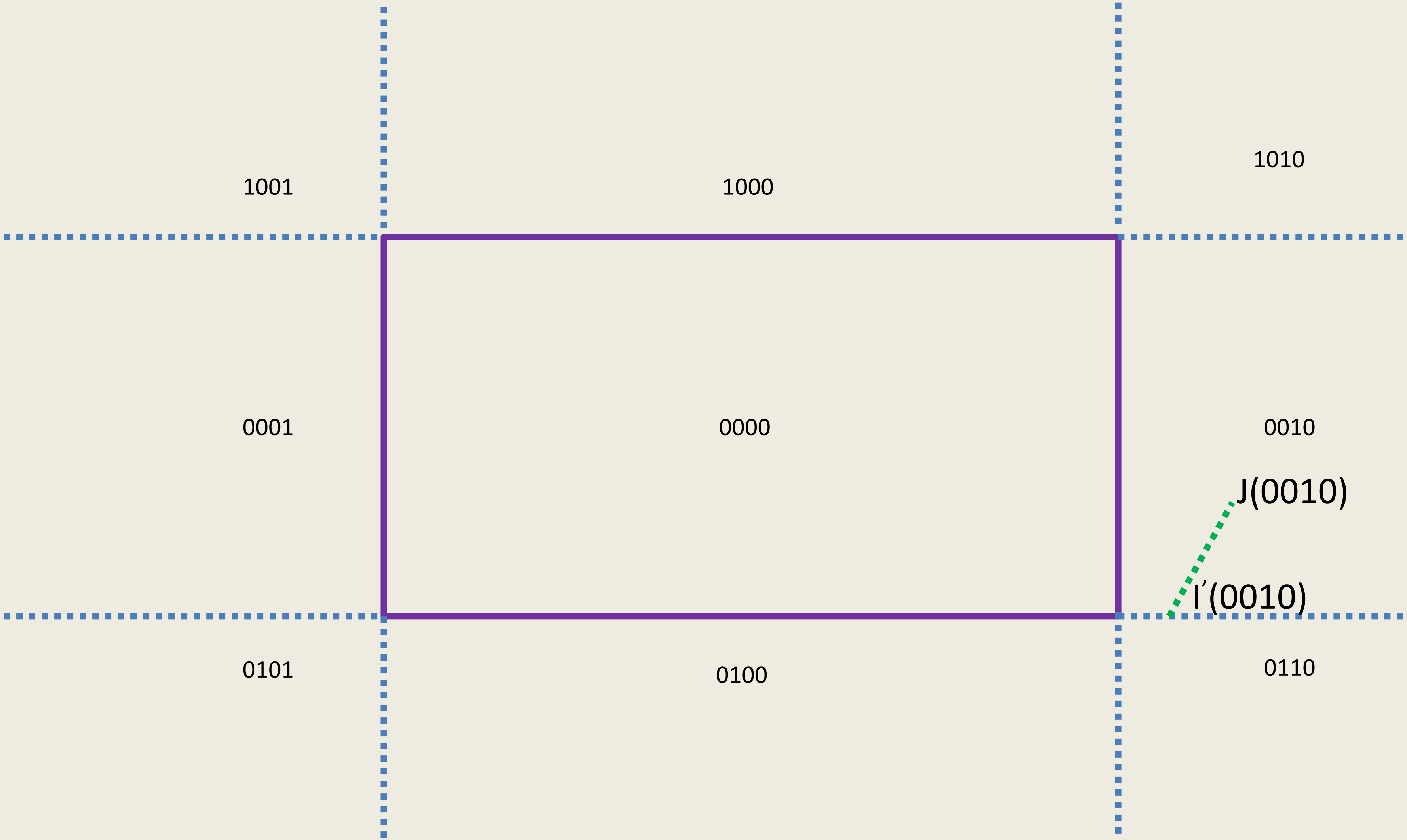
# Cohen-Sutherland Algorithm..



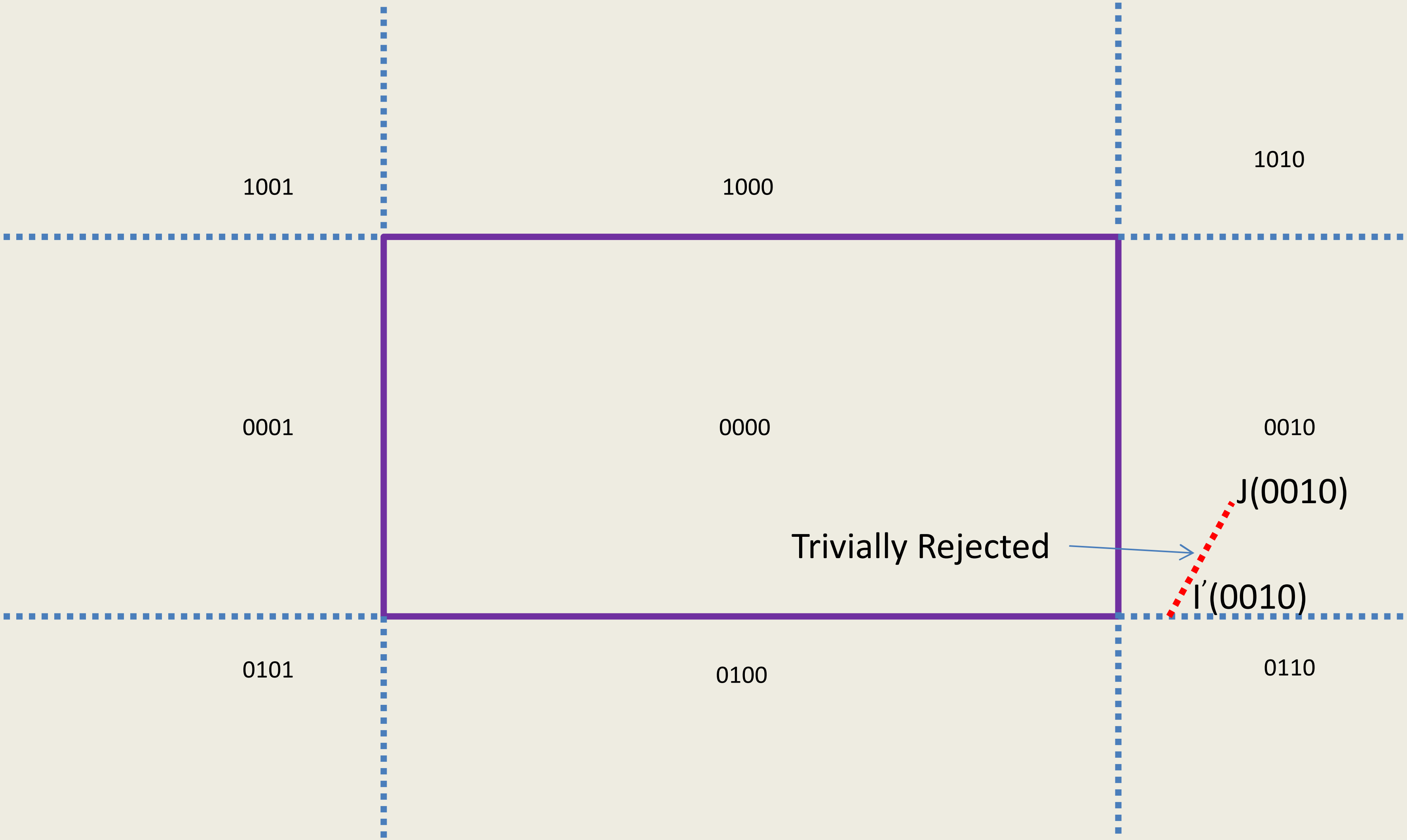
# Cohen-Sutherland Algorithm..



# Cohen-Sutherland Algorithm..



# Cohen-Sutherland Algorithm..



# Cyrus Beck Line Clipping Algorithm

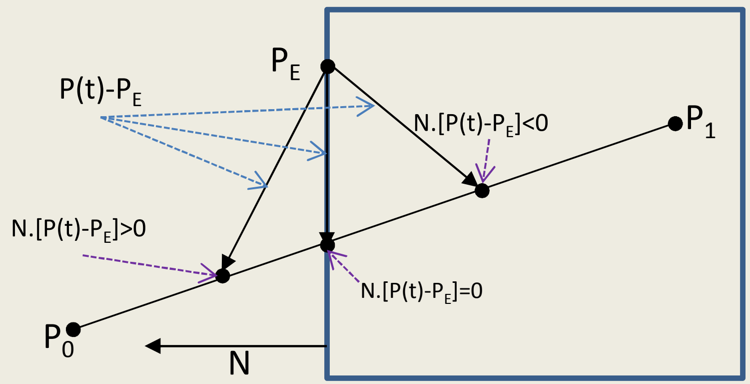
**Parametric Line Equation:**

$$P(t) = P_0 + (P_1 - P_0)t$$

$$P(0) = P_0$$

$$P(1) = P_1$$

$P_E$ is any arbitrary value on the edge




$N.[P(t)-P_E]=0$

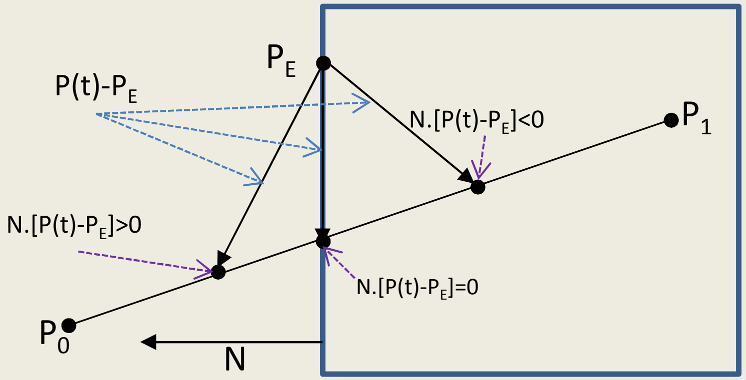
$=> N.[P_0 + (P_1 - P_0)t - P_E]=0$

$=> N.[P_0 - P_E] + N.[P_1 - P_0]t=0$

Let, $D = (P_1 - P_0)$

$$t = \frac{N_i.[P_0 - P_E]}{-N.D}$$

# Cyrus Beck Line Clipping Algorithm




$$t = \frac{N_i.[P_0 - P_E]}{-N.D}$$

For this to be true, the algorithm checks that

N!= 0, normal should not be zero

D!=0, ($P_1$ !=$P_0$)

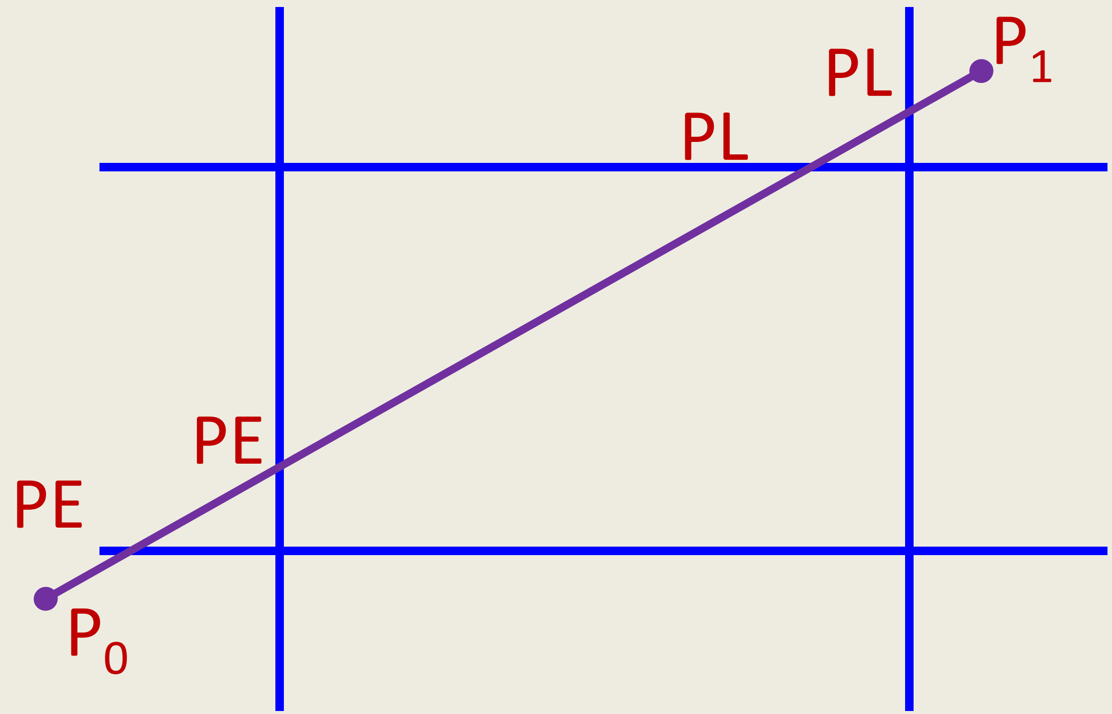
N.D !=0 (Edge and line should not be parallel)

# Cyrus Beck Line Clipping Algorithm

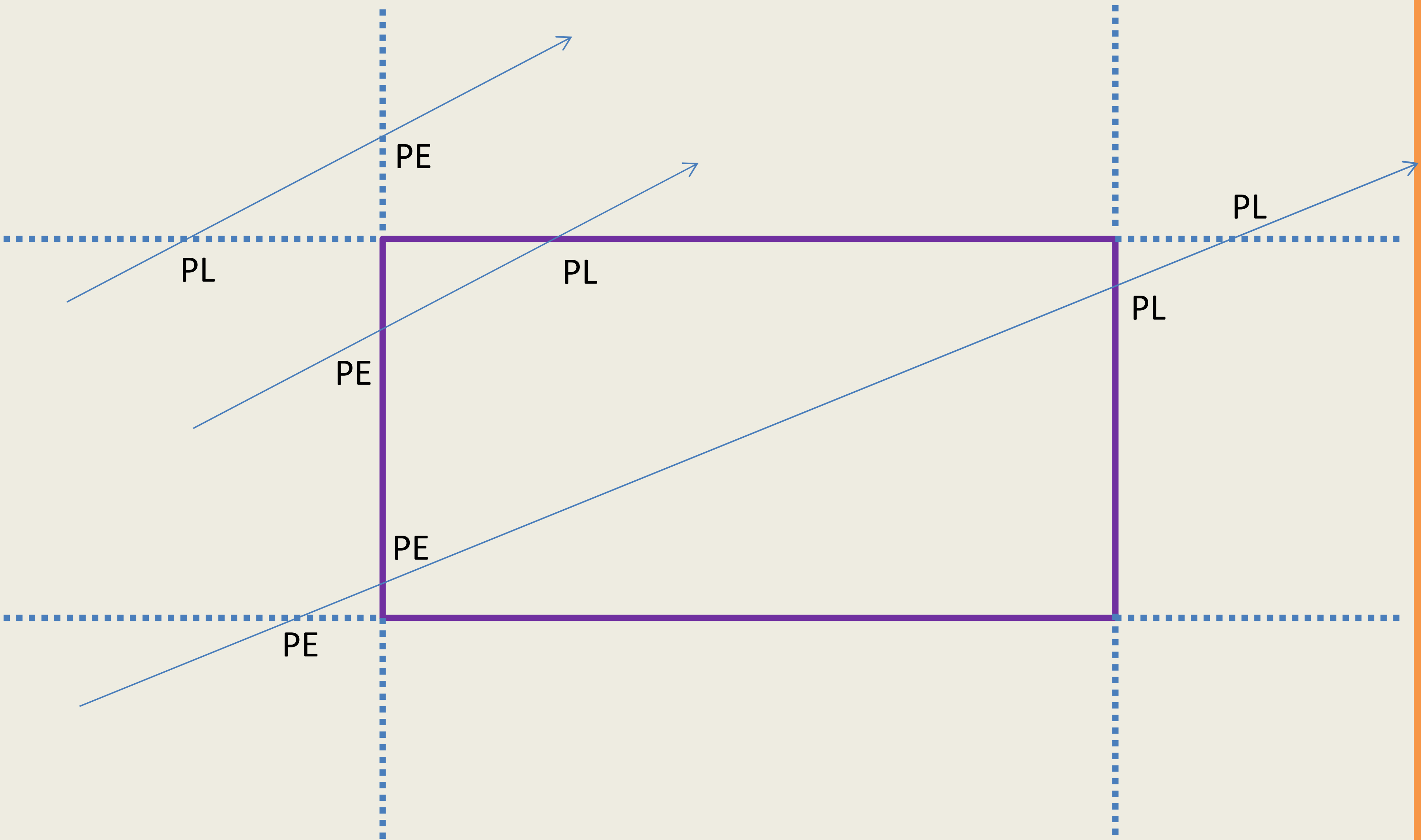
- Compute t for line intersection with all four edges by selecting in each of the four edges of clip rectangle
- Obtain the values for t
- Discard all ($t < 0$) and ($t > 1$)
- Classify each remaining intersection as
  - Potentially Entering (PE)
  - Potentially Leaving (PL)
- $N_L \ [P_1 - P_0] > 0$ implies PL
- $N_L \ [P_1 - P_0] < 0$ implies PE
  - Note that we computed this term when computing t

# Cyrus Beck Line Clipping Algorithm..

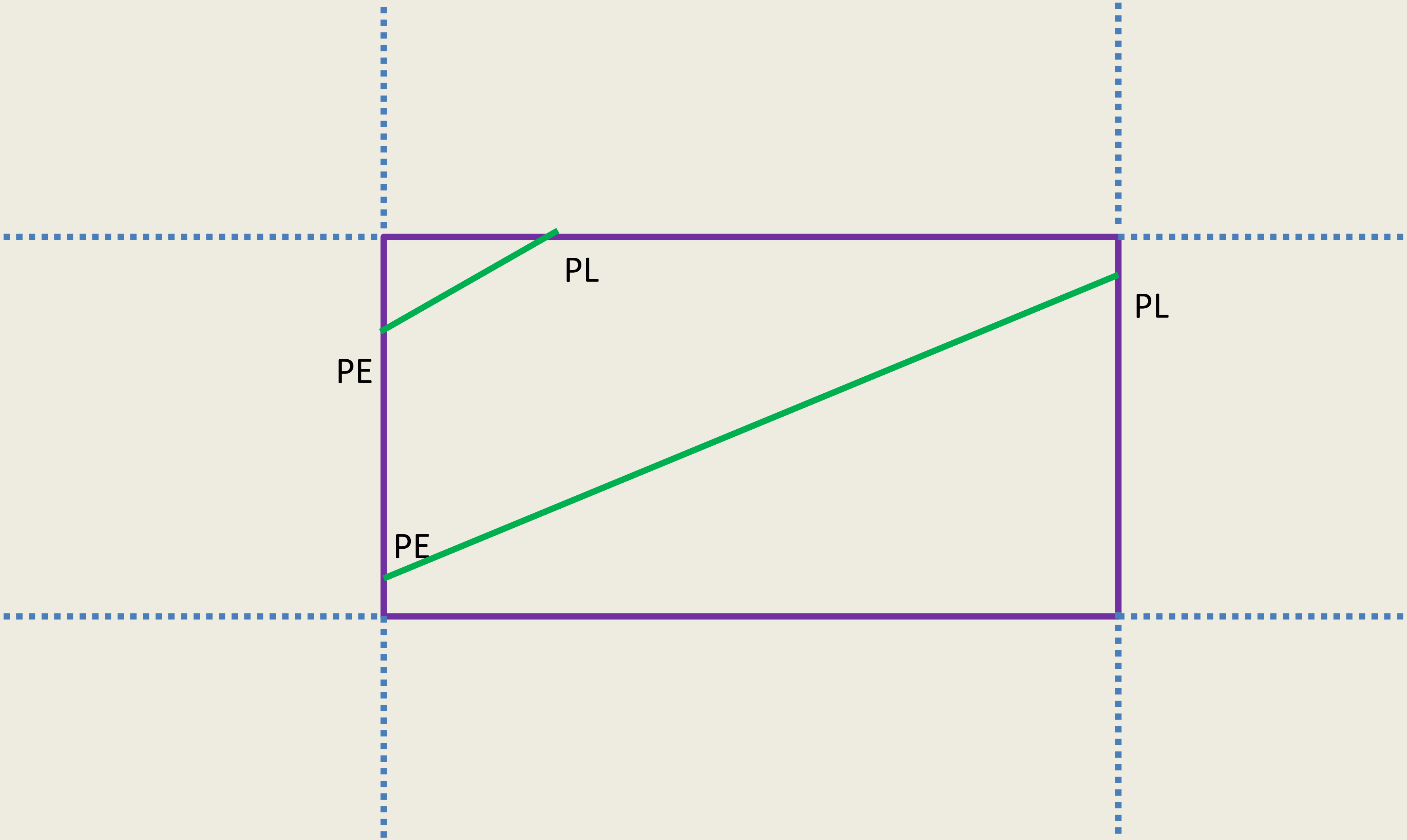
- Compute PE with largest t
- Compute PL with smallest t
- Clip to these two points



# Cyrus Beck Line Clipping Algorithm..



# Cyrus Beck Line Clipping Algorithm..



**Thank you**