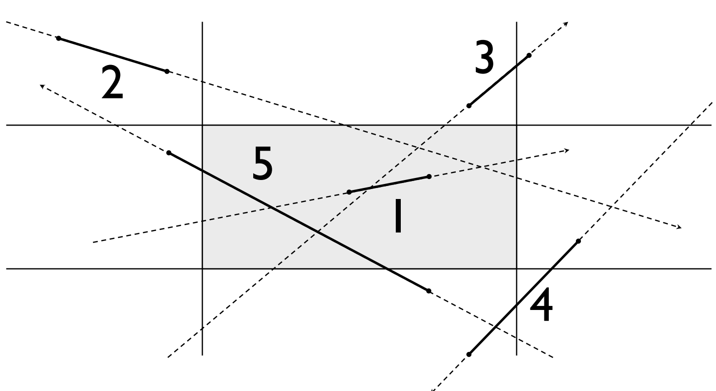
Liang-Barsky Clipping

- Remember linear interpolation? It allows us to write a line in terms of a parameter u from 0.0 to 1.0: $x = x_1 + u(x_2 - x_1)$, $y = y_1 + u(y_2 - y_1)$
- Liang-Barsky asks: for what values of u does a line segment enter or exit the bounds?
- There can be, at most, two of each; we care about the maximum entry value and the minimum exit value

- For each line segment, for each boundary, check the value of u at the intersection of the segment's line with that boundary
- If $u < 0$ on entry and $u > 1$ on exit — *accept*
- If $u > 1$ on entry or $u < 0$ on exit — *reject*
- If u on entry > u on exit — *reject*
- Otherwise, clip and try again — note how we don't need to perform an extra calculation, because the new point can be derived from u



Line 1: max entry < 0, min exit > 1 — *accept*

Line 2: max entry > 1, min exit > 1 — *reject*

Line 3: max entry < 0, min exit < 0 — *reject*

Line 4: max entry > min exit — *reject*

Line 5: max entry > 0, min exit < 1, max entry < min exit — *clip*

Liang-Barsky Algorithm

- For a given line segment (x_1, y_1) to (x_2, y_2), derive the parametric form of its line: $x = x_1 + u(x_2 - x_1)$, $y = y_1 + u(y_2 - y_1)$
- For each boundary (L, R, T, B), calculate the value of u for that line at that boundary; note that a point is within the boundary if:

$$\mathrm{L} \leq x \leq \mathrm{R} \text{ and } \mathrm{B} \leq y \leq \mathrm{T}$$

- Substituting the parametric form, let:

$$dx = x_2 - x_1,\ dy = y_2 - y_1$$

$$\mathrm{L} \leq x_1 + u(dx) \leq \mathrm{R} \text{ and } \mathrm{B} \leq y_1 + u(dy) \leq \mathrm{T}$$

- If we break these inequalities up, we get these conditions:

$$-\,dx\ (u) \leq x_1 - \mathrm{L} \rightarrow \text{let } C = -\,dx,\ q = x_1 - \mathrm{L}$$

$$dx\ (u) \leq \mathrm{R} - x_1 \rightarrow \text{let } C = dx,\ q = \mathrm{R} - x_1$$

$$-\,dy\ (u) \leq y_1 - \mathrm{B} \rightarrow \text{let } C = -\,dy,\ q = y_1 - \mathrm{B}$$

$$dy\ (u) \leq \mathrm{T} - y_1 \rightarrow \text{let } C = dy,\ q = \mathrm{T} - y_1$$

- Note how, for each C and its corresponding boundary:

  $C < 0 \Rightarrow$ line goes out $\rightarrow$ in: *entry*

  $C > 0 \Rightarrow$ line goes in $\rightarrow$ out: *exit*

  $C = 0 \Rightarrow$ line is parallel to the boundary

- So, we can calculate u for each boundary by calculating q and C; the value of C tells us if we are looking at an entry or exit point for the boundary. Thus, we can now apply the conditions:
  - If $u < 0$ on entry and $u > 1$ on exit — *accept*
  - If $u > 1$ on entry or $u < 0$ on exit — *reject*

```
procedure ClipAndDrawLine(x1, y1, x2, y2: real) is
    u1: real := 0.0;    dx: real := x2 - x1;
    u2: real := 1.0;    dy: real := y2 - y1;

    function Reject(C, q: real) return boolean is
        u: real := q / C;
    begin
        if C < 0 then
            if u > u2 then return true; elsif u > u1 then u1 := u; end if;
        elsif C > 0 then
            if u < u1 then return true; elsif u > u2 then u2 := u; end if;
        else
            if q < 0 then return true;
        end if;
        return false;
    end Reject;

begin
    if Reject(-dx, x1 - L) then return; end if;
    if Reject(dx, R - x1) then return; end if;
    if Reject(-dy, y1 - B) then return; end if;
    if Reject(dy, T - y1) then return; end if;
    if u2 < 1.0 then (x2, y2) := (x1 + u2 * dx, y1 + u2 * dy); end if;
    if u1 > 0.0 then (x1, y1) := (x1 + u1 * dx, y1 + u1 * dy); end if;
    DrawLine(x1, y1, x2, y2);
end ClipAndDrawLine;
```