Draw lines between two points

Nicolas FLASQUE

nicolas.flasque@efrei.fr

Reminder of the situation

A line is defined by two points A and B, of coordinates .
$(x_A, y_A)$and (x_B, y_B) . These coordinates are entire coordinates.

We want to find which pixels **make up the line to be drawn between A and B.**

Therfore, pixels have entire coordinates: they are points.

A line is therefore a set of points.

The line between A and B being the same as that between B and A**, we therefore choose that $\boldsymbol{x_A \leq x_B}$**

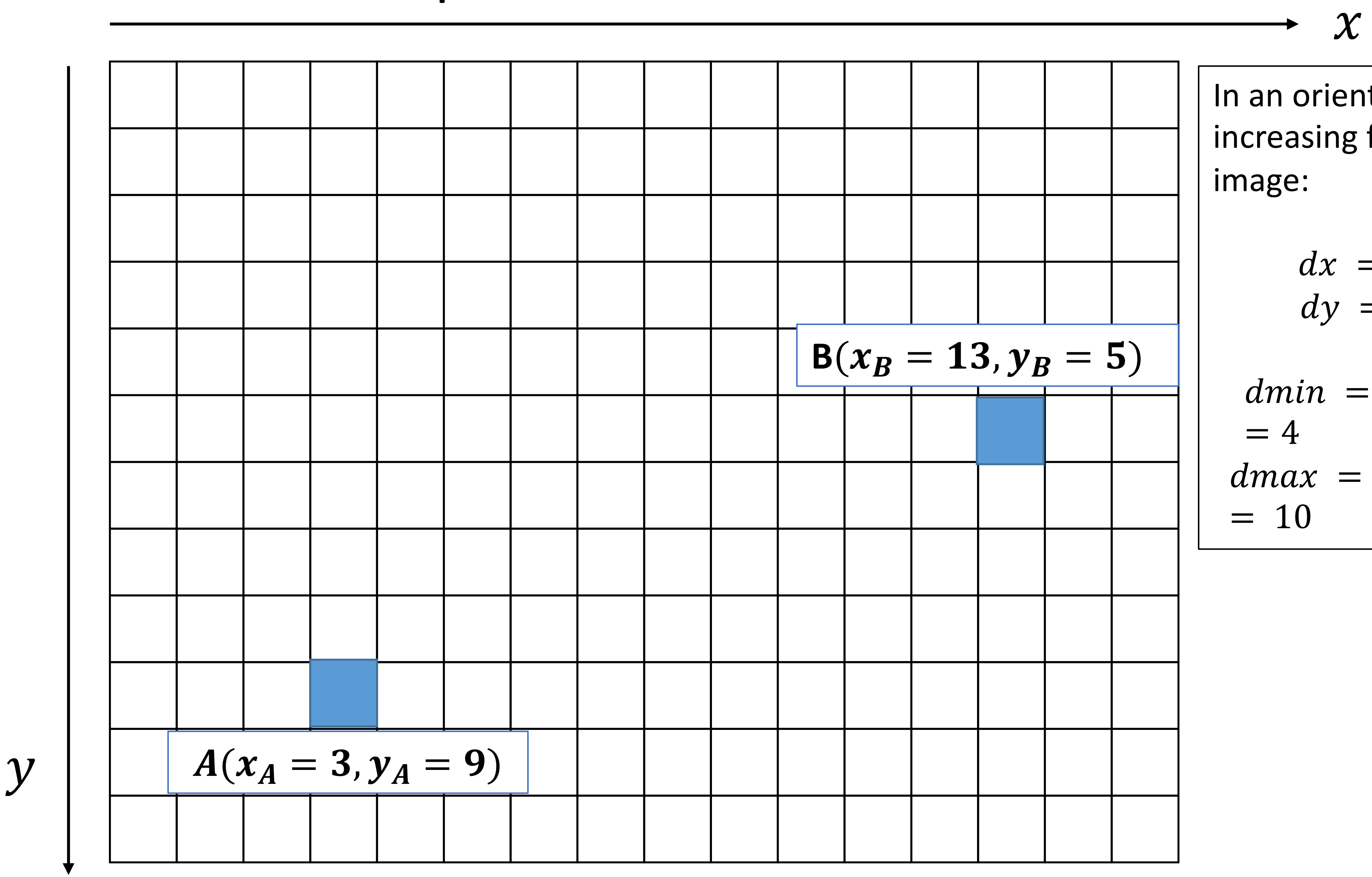
We calculate $dx = x_B - x_A$ and $dy = y_B - y_A$. We have $dx >= 0$

We will try to find out how many straight segments (imagine this as stair steps) we must draw to materialize the line from A to B:

We calculate $dmin = \min(dx, |dy|)$ and $dmax = \max(dx, |dy|)$

$dmin$ is the smallest difference between coordinates, $dmax$ is the largest difference between coordinates.

Our example



In an orientated landmark, where *y* is increasing from top to bottom of the image:

$$dx = x_B - x_A = 13 - 3 = 10$$
$$dy = y_B - y_A = 5 - 9 = -4$$

$$dmin = \min(dx, |dy|) = \min(10,4) = 4$$
$$dmax = \max(dx, |dy|) = \max(10,4) = 10$$

The number of segments to 'draw' is then $(dmin + 1)$

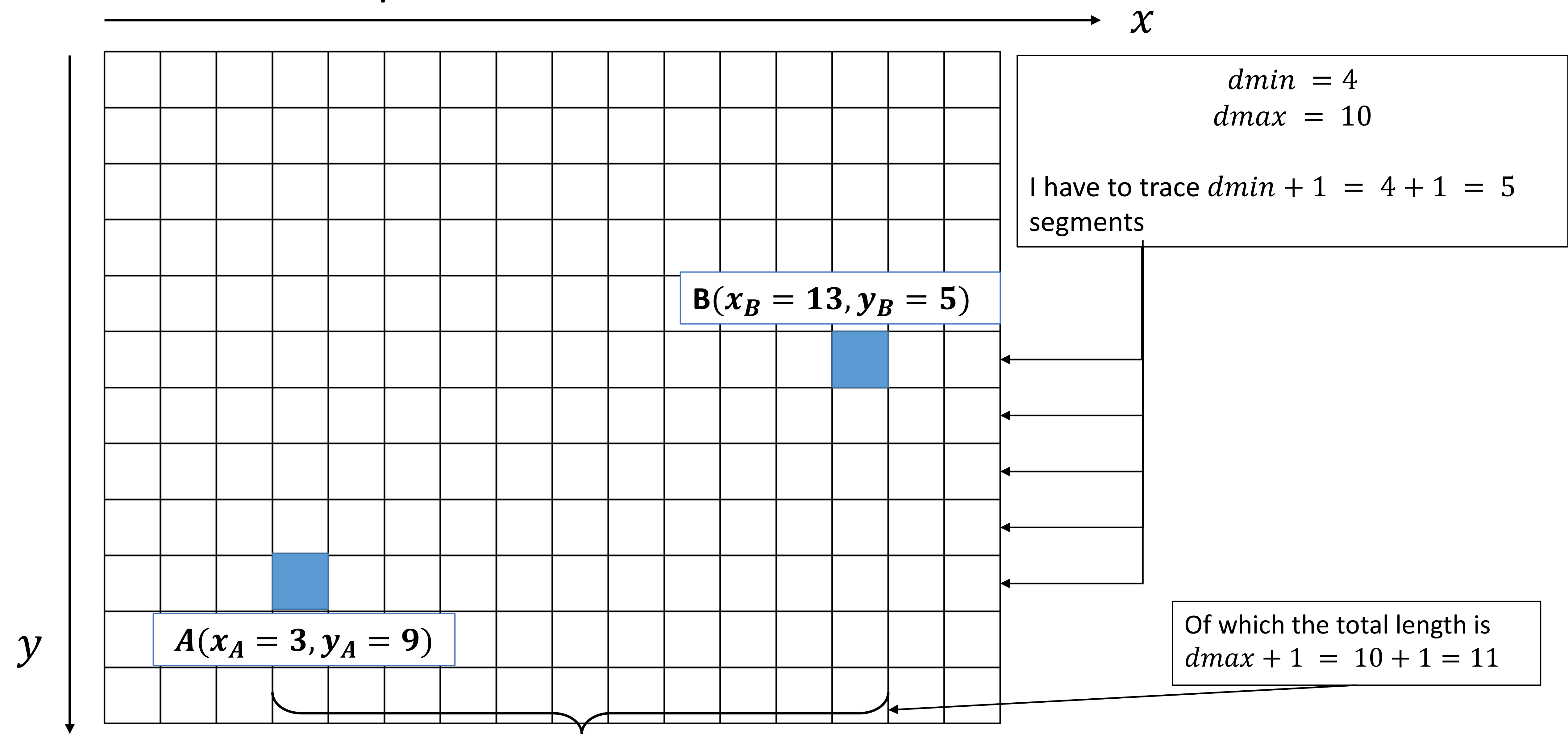
For example, if $dmin$ is 0, then a single segment must be drawn that connects the two points, which are aligned horizontally or vertically.

An entire variable is used for this purpose: $nb\_segs \ = \ (dmin + 1)$

How many dots (pixels) in each segment?

The idea is to distribute in a balanced way, the number of points: there are at least : $(dmax + 1)/(dmin + 1)$

Our example



x
y
$dmin = 4$
$dmax = 10$
I have to trace $dmin + 1 = 4 + 1 = 5$ segments
$\mathrm{B}(x_B = 13, y_B = 5)$
$A(x_A = 3, y_A = 9)$
Of which the total length is $dmax + 1 = 10 + 1 = 11$

First part: calculate the 'base' segment size:

It is given to us by: $(dmax + 1)/(dmin + 1)$

$dmax$ and being integers, in C, it is an integer division, so we get $dmin$ an integer, which is the 'base' size of each segment.

In our example, we calculate $11\ /\ 5$: the total size divided by the number of segments:$11\ /\ 5\ =\ 2$ (integer division)

We will then get 5 segments with size 2 $(pixels)$ as the base of our line

so we create an array $segments$ with nb_segs elements, and we initialize all its elements with the basic size: here, an array with 5 elements, each element worth 2

We then try to distribute the missing pixels on the segments:

So we still have $\mathbf{remaining} = (\boldsymbol{dmax} + \mathbf{1})\%(\boldsymbol{dmin} + \mathbf{1})$

On our example:$\boldsymbol{remaining} =$ **11 % 5 = 1 pixel to distribute. Which segment will receive this pixel?**

To do this, we calculate the sum of the leftovers: we create a table that indicates how many pixels we must add to each segment (this table will contain 0 and 1)

Pixels to be distributed (continued): This code calculates the number of pixels remaining and updates the table of segments.

We assume we have the segments `segments`

```
int *cumuls = (int *)malloc(nb_segs*sizeof(int));
cumuls[0]=0;
for (int i = 1; i < nb_segs;i++)
{
       cumulated[i] = ((i*remaining)%(dmin+1) < (i-1)*remaining)%(dmin+1);
       segments[i] = segments[i]+cumuls[i];
}
```

We now know the segments connecting A to B and their size.

For the plot: we start from the coordinates of point A and we trace segment by segment: we must know if they are horizontal or vertical

You need to know if you are tracing ‘upward' or 'downward'

Si $dy < 0$

we trace down

Si $dx > |dy|$

the segments are horizontal (they are covered by increasing x)

otherwise

the segments are vertical (they are covered by decreasing y)

Otherwise

we trace upwards

If dy < 0// we trace down

    Si dx > |dy|

        the segments are horizontal (they are covered by increasing x)

        with each change of segment, we decrease y

    Otherwise

        the segments are vertical (they are covered by decreasing y)

        with each change of segment, we increase x

Otherwise// we trace up

    Si dx > dy

        the segments are horizontal (they are covered by increasing x)

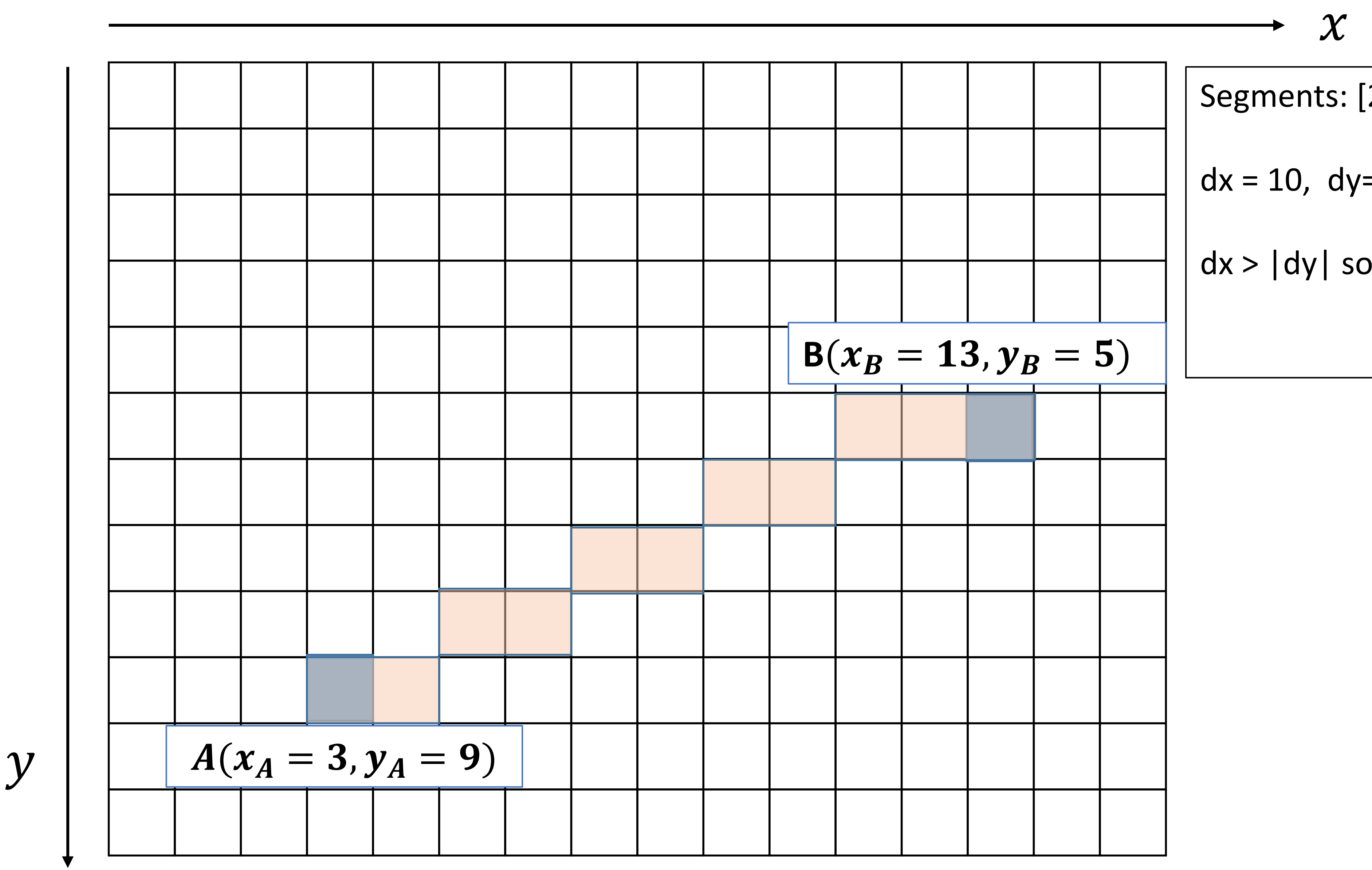
        with each change of segment, we increase y

    Otherwise

        the segments are vertical (they are covered by increasing y)

        with each change of segment, we increase x

illustration



Segments: [2.2.2.2.3]

dx = 10, dy= -4

dx > |dy| so horizontal segments

Finally, to trace the segments:

We use a double loop:

The starting point is A

For i from 0 to nb_segs-1

    for j from 0 to segments[i]

        Add to the pixel table the pixel coordinates (so increase or decrease x or y) according to the situation described in slide 11.

    move to the next segment (so increase or decrease x or y)