Jak narysować odcinek?

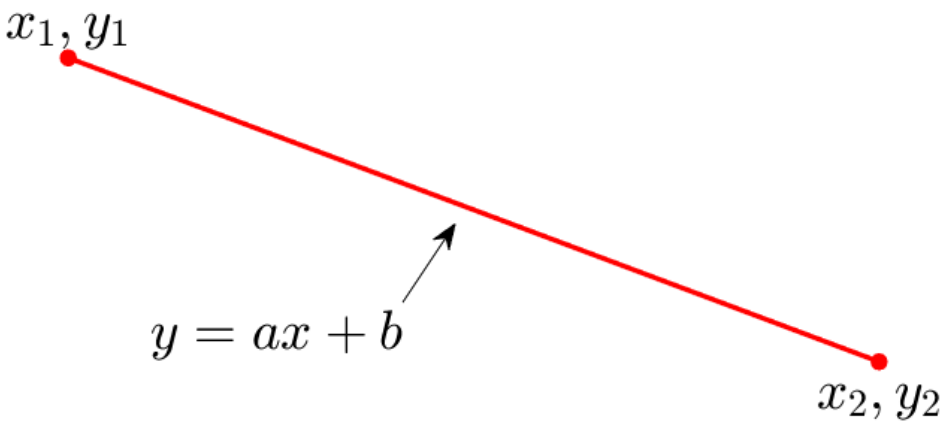
Geneza problemu

- **Grafika wektorowa**

Oparta o geometrię analityczną.

Każdy obiekt opisany jest przy pomocy współrzędnych x, y oraz wzorów arytmetycznych.

Współrzędne x, y nie muszą być liczbami rzeczywistymi.

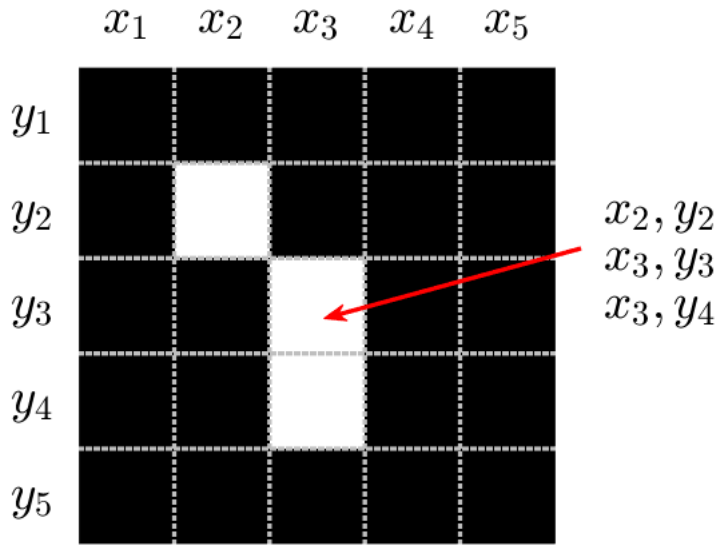


- **Grafika rastrowa**

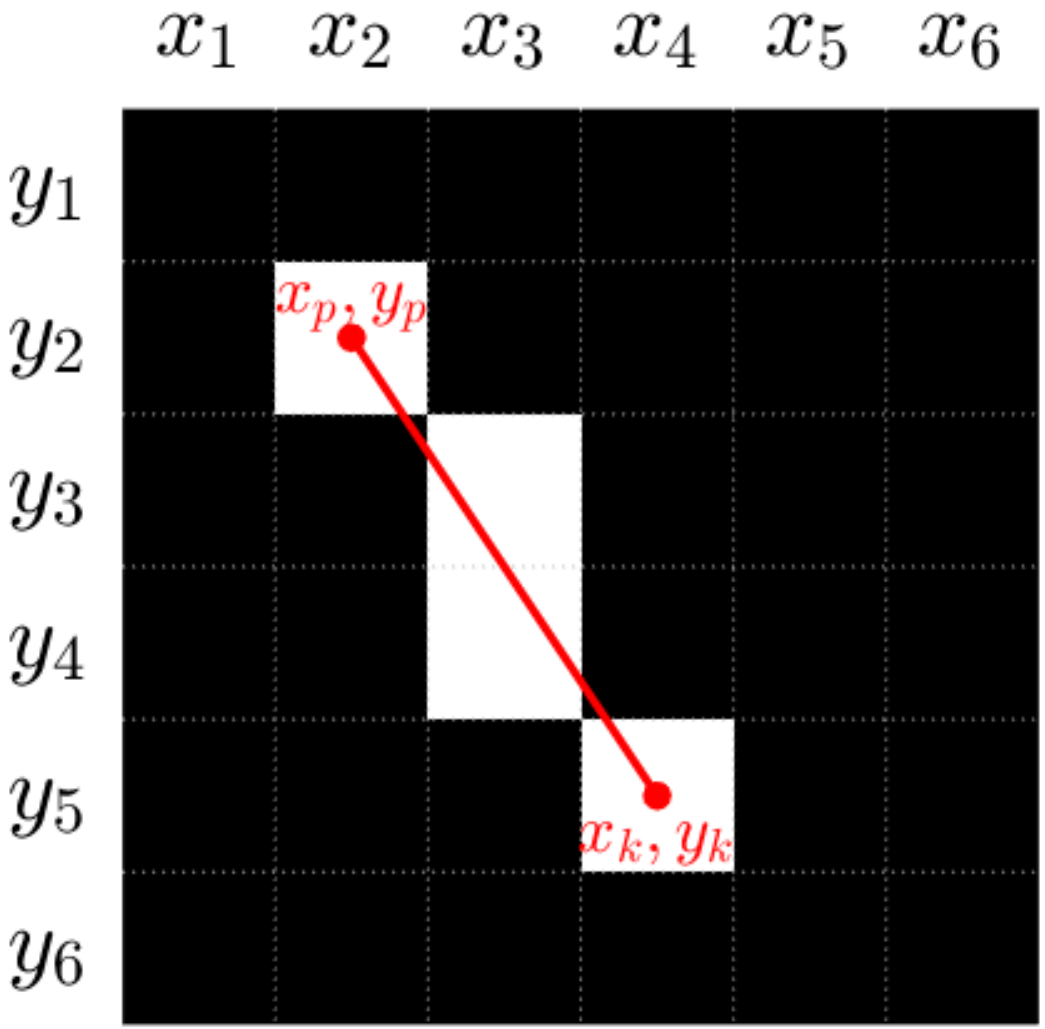
Oparta o siatkę ze współrzędnymi całkowitymi.

Każdy element sadki określany jest jako piksel oraz posiada swoje współrzędne oraz wartość.

Każdy obiekt opisany jest przez listę pikseli.



Jak wyznaczyć współrzędne pikseli?



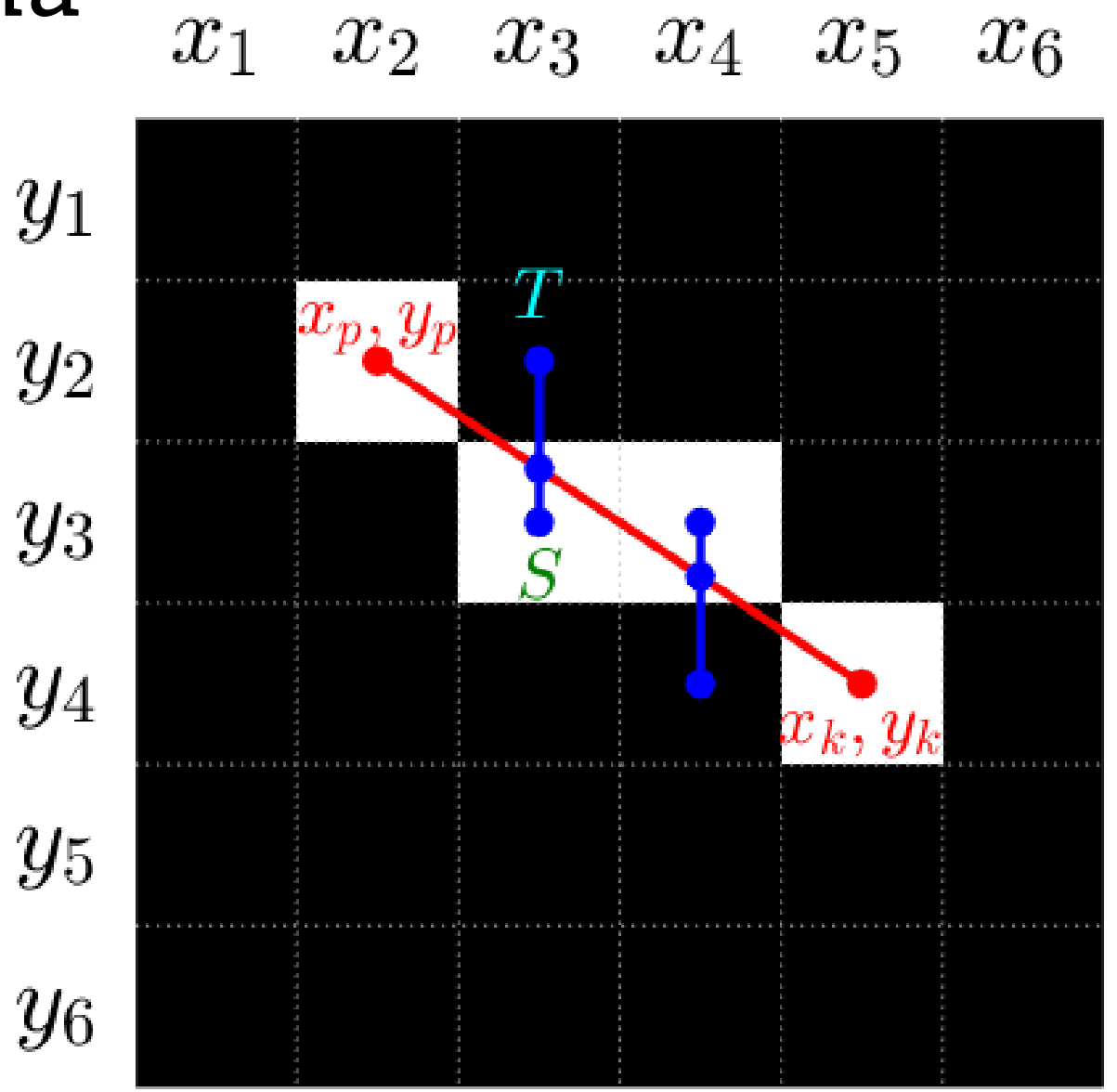
Algorytm Bresenhama

- opracowany przez Jacka Bresenham w 1965r.
- zalety:
  - oparty o proste obliczenia
  - wykonuje operacje tylko na liczbach całkowitych
  - kolejny piksel obliczony jest po wykonaniu jednego porównania i jednego dodawania
- rozwinięty do rysowania krzywych
- znalazł zastosowanie w sterownikach dla wyświetlaczy monochromatycznych

Algorytm Bresenhama

- Obliczenia wykonywane są w pętli dla $x_p, x_p+1, x_p+2, \ldots, x_k$
- Możliwe są dwa przejścia
  - do punktu $T(x_i+1, y_i)$
  - do punktu $S(x_i+1, y_i+1)$
- Decyzja podjęta na podstawie zmiennej d_i
  - $d_i > 0$, przejście do punkt S
  - $d_i < 0$, przejście do punktu T
- Początkowa wartość zmiennej

- Po każdej iteracji następuję modyfikacja wartości zmiennej d_i

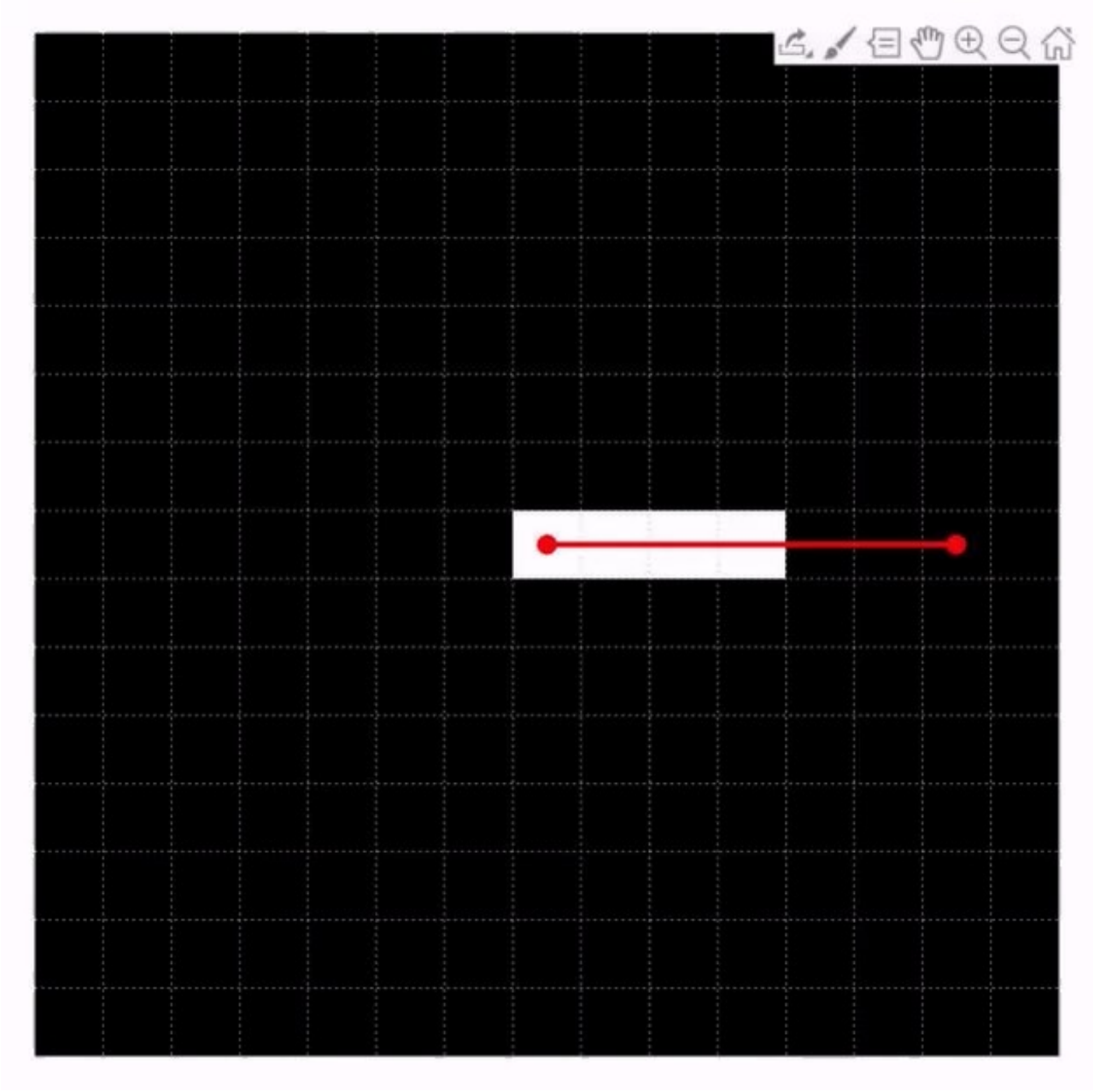


Algorytm Bresenhama

- Obliczenia wykonywane są w pętli dla $x_p, x_p+1, x_p+2, \ldots, x_k$
- Możliwe są dwa przejścia
  - do punktu $T(x_i+1, y_i)$
  - do punktu $S(x_i+1, y_i+1)$
- Decyzja podjęta na podstawie zmiennej d_i
  - $d_i > 0$, przejście do punkt S
  - $d_i < 0$, przejście do punktu T
- Początkowa wartość zmiennej
- Po każdej iteracji następuję modyfikacja wartości zmiennej d_i

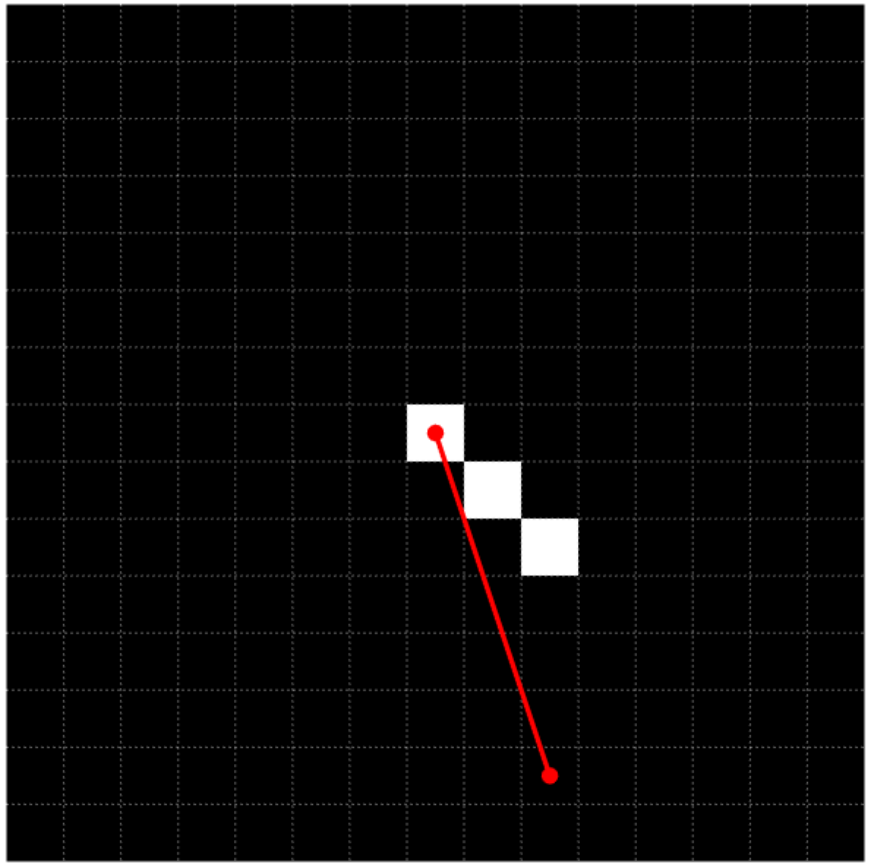
```
def bresenham(x1, y1, x2,
y2):
    dy = y2 - y1
    dx = x2 - x1

    d = 2*dy - dx
    y = y1
    punkty = []
    for x in range(x1,
x2+1):
        pt = (x, y)
        punkty.append(pt)
        if d > 0:
            y += 1
            d -= 2*dx
        d += 2*dy
    return punkty
```

Czy to wszystko?



Co poszło nie tak?

- Problem
  - pojedyncza iteracja to zmiana współrzędnej osi x o jeden
- Potencjalne rozwiązanie:
  - obróćmy odcinek o 90°
  - narysujmy zgodnie z algorytmem
  - obróćmy wynik o -90°
- Najprostsze rozwiązanie
  - zamiana osi x z osią y

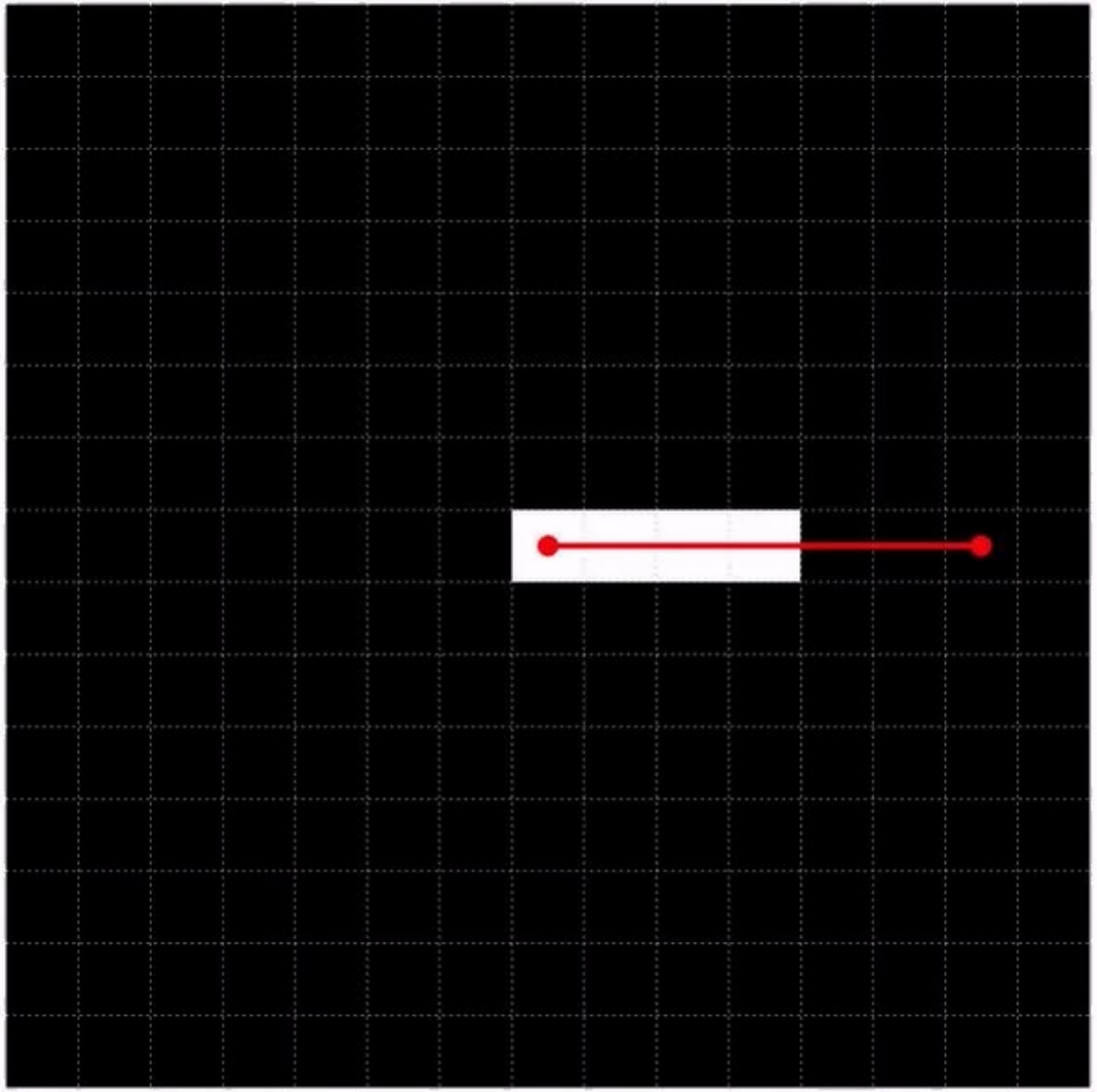


Algorytm Bresenhama – linie strome

- Potrafimy rysować odcinki o kącie nachylenia z przedziału 0-45°
- Jak narysować odcinki o kącie nachylenia powyżej 45° ?
- Potencjalne rozwiązanie:
  - obróćmy odcinek o 90°
  - narysujmy zgodnie z algorytmem
  - obróćmy wynik o -90°
- Najprostsze rozwiązanie
  - Zamiana osi x z osią y

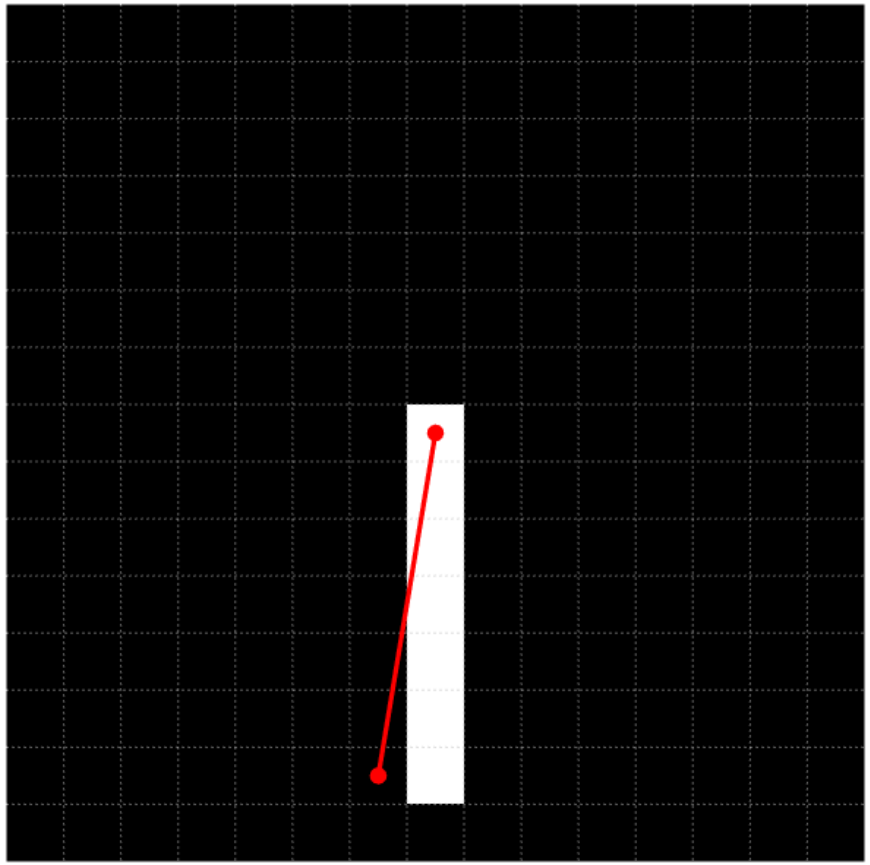
```
def bresenhamHigh(x1, y1, x2, y2):
    dy = y2 - y1
    dx = x2 - x1

    czy_stromy = abs(dy) > abs(dx)
    if czy_stromy:
        x1, y1 = y1, x1
        x2, y2 = y2, x2
    dy = y2 - y1
    dx = x2 - x1

    d = 2 * dy - dx
    y = y1
    punkty = []
    for x in range(x1, x2+1):
        pt = (y, x) if czy_stromy else (x,
y)
        punkty.append(pt)
        if d > 0:
            y += 1
            d -= 2 * dx
        d += 2 * dy
    return punkty
```

Czy teraz to wszystko?



Co tym razem nie działa?

- Problem 1:
  - Algorytm przemieszcza się z lewej do prawej
- Rozwiązanie:
  - Zamiana kolejności punktów wejściowych
- Problem 2:
  - Algorytm w osi y porusza się tylko w dół
- Rozwiązanie
  - Wprowadzić zmianę kierunku uzależnianą od znaku *dx*

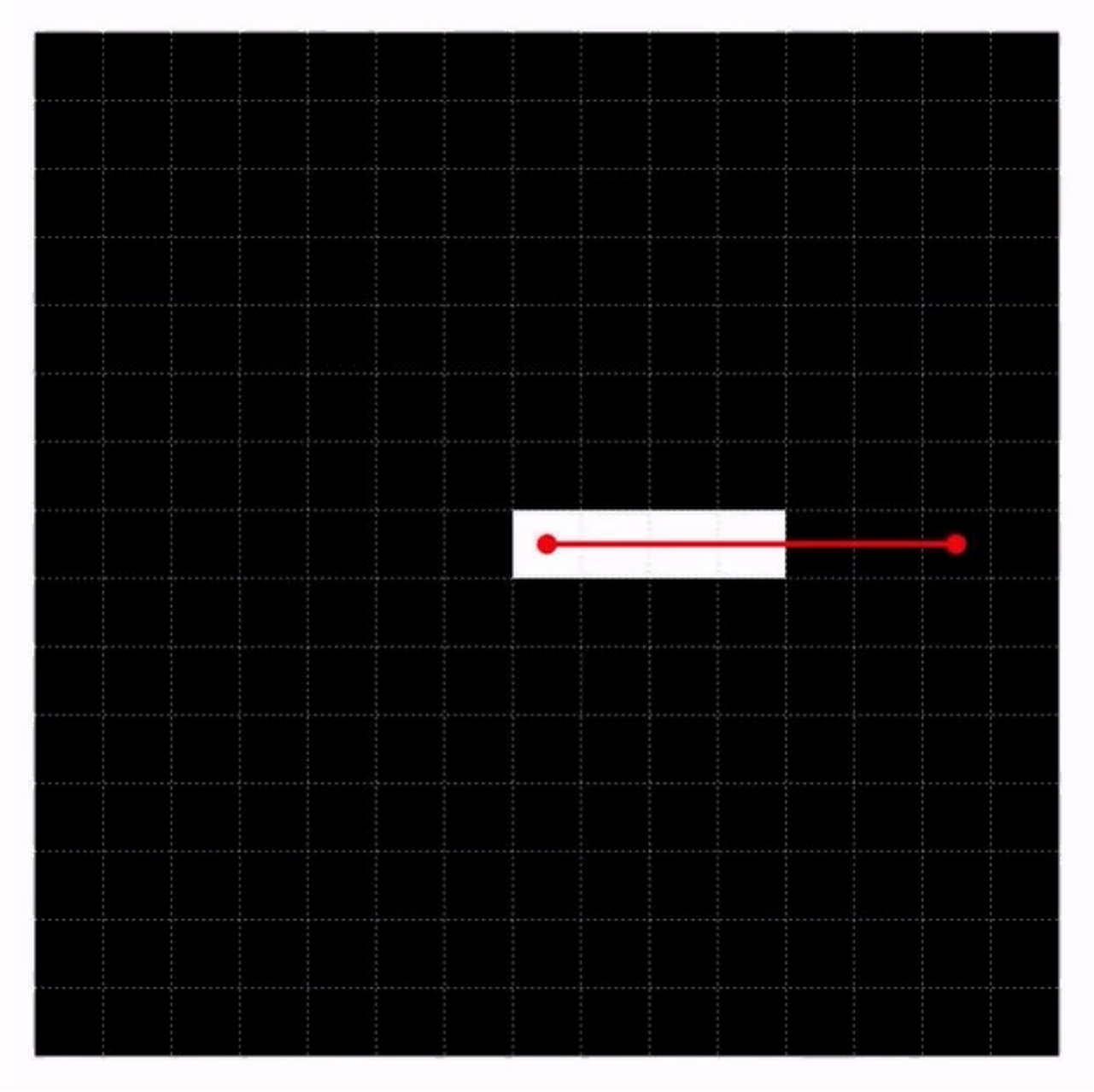


Kompletny algorytm?

- Problem 1:
  - Algorytm przemieszcza się z lewej do prawej
- Rozwiązanie:
  - Zamiana kolejności punktów wejściowych
- Problem 2:
  - Algorytm w osi y porusza się tylko w dół
- Rozwiązanie
  - Wprowadzić zmianę kierunku uzależnianą od znaku *dx*

```
def bresenhamFull(x1, y1, x2, y2):
    dy = y2 - y1
    dx = x2 - x1

    czy_stromy = abs(dy) > abs(dx)
    if czy_stromy:
        x1, y1 = y1, x1
        x2, y2 = y2, x2

    czy_z_lewej = x1 < x2
    if not czy_z_lewej:
        x1, x2 = x2, x1
        y1, y2 = y2, y1

    dy = y2 - y1
    dx = x2 - x1

    czy_w_dol = dy > 0
    if czy_w_dol:
        ystep = 1
    else:
        ystep = -1
        dy = -dy

    d = 2 * dy - dx
    y = y1
    punkty = []
    for x in range(x1, x2+1):
        pt = (y, x) if czy_stromy else
(x, y)
        punkty.append(pt)
        if d > 0:
            y += ystep
            d -= 2 * dx
        d += 2 * dy
    if not czy_z_lewej:
        punkty.reverse()
    return punkty
```

Czy teraz będzie działać?



Dziękuje za uwagę

Fundusze Europejskie Polska Cyfrowa
Rzeczpospolita Polska
CENTRUM PROJEKTÓW POLSKA CYFROWA
Unia Europejska Europejski Fundusz Rozwoju Regionalnego

Projekt "Centrum Mistrzostwa Informatycznego" współfinansowany jest ze środków Unii Europejskiej z Europejskiego Funduszu Rozwoju Regionalnego w ramach Programu Operacyjnego Polska Cyfrowa na lata 2014 - 2020