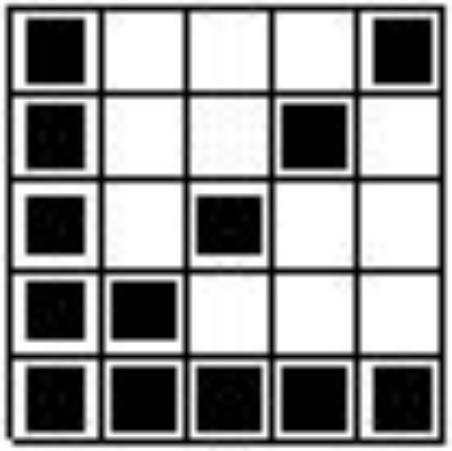
Интерактивная Компьютерная Графика

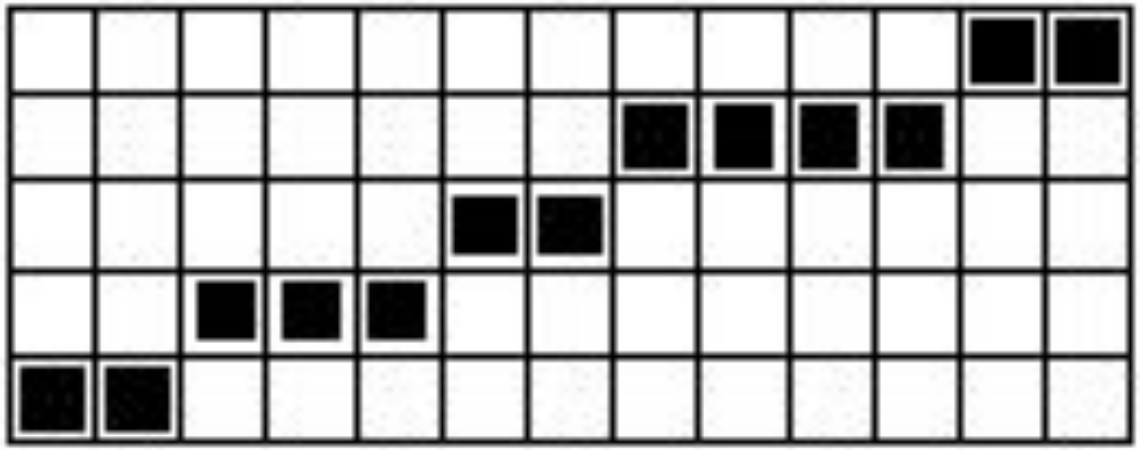
Часть 1-3

(растеризация)

Требования к растеризации отрезка

Требования	Факт
Концы **должны** находиться в заданных точках	Концы попадают только в пиксели (**неоднозначность**)
Отрезки **должны** выглядеть как прямые	**Только** для вертикального и горизонтального случаев
Яркость **не должна** меняться вдоль отрезка	Расстояние до пикселей в каждой точке отрезка различно, т.е. яркость - **непостоянна**





1. Простейший алгоритм ЦДА

DDA – Digital Differential Analyzer (Цифровой Дифференциальный Анализатор)
вычислительное устройство, применявшееся ранее для генерации векторов

$$\begin{pmatrix} x0 \\ y0 \end{pmatrix} \rightarrow \begin{pmatrix} px0 \\ py0 \end{pmatrix} = \begin{pmatrix} \text{round}(x0) \\ \text{round}(y0) \end{pmatrix}$$ начало отрезка (вещественное и пиксельное)

$$\begin{pmatrix} x1 \\ y1 \end{pmatrix} \rightarrow \begin{pmatrix} px1 \\ py1 \end{pmatrix} = \begin{pmatrix} \text{round}(x1) \\ \text{round}(y1) \end{pmatrix}$$ конец отрезка (вещественное и пиксельное)

$$len = \max(|px1 - px0|, |py1 - py0|)$$

$$dx = \begin{pmatrix} x1 - x0 \\ len \end{pmatrix}$$

$$dy = \begin{pmatrix} y1 - y0 \\ len \end{pmatrix}$$

$$i = \overline{1, len}$$

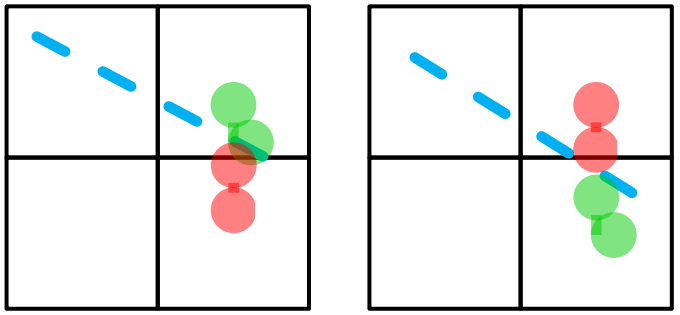
$$x_i = x_{i-1} + dx \rightarrow px_i = \text{round}(x_i)$$

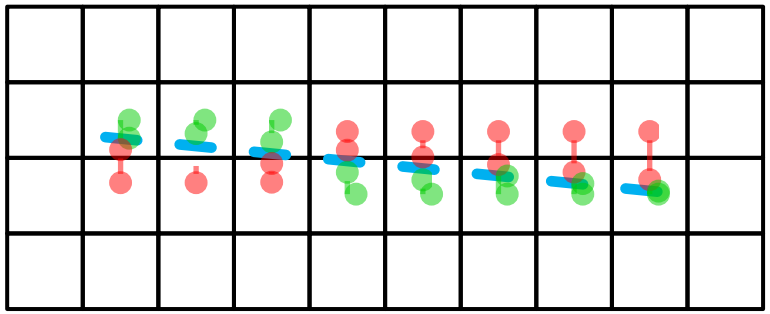
$$y_i = y_{i-1} + dy \rightarrow py_i = \text{round}(y_i)$$

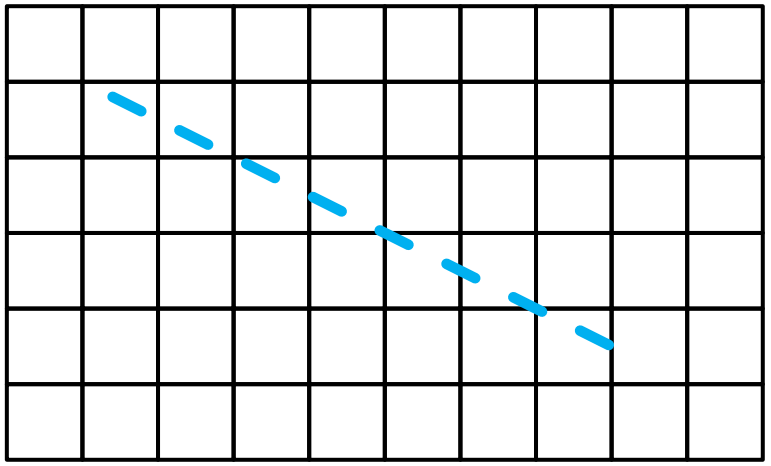
2. Алгоритм Брезенхема (Bresenham's line algorithm)

- Рассмотрим ситуацию:
  - угол наклона отрезка в диапазоне **[0°, -45°]**
- Для каждого столбца пикселей вычисляется **ошибка** – расстояние между реальной координатой **y** и ближайшим пикселем.
- Если ошибка не превышает половину высоты ячейки (пикселя), то она заполняется

Возможные ситуации:

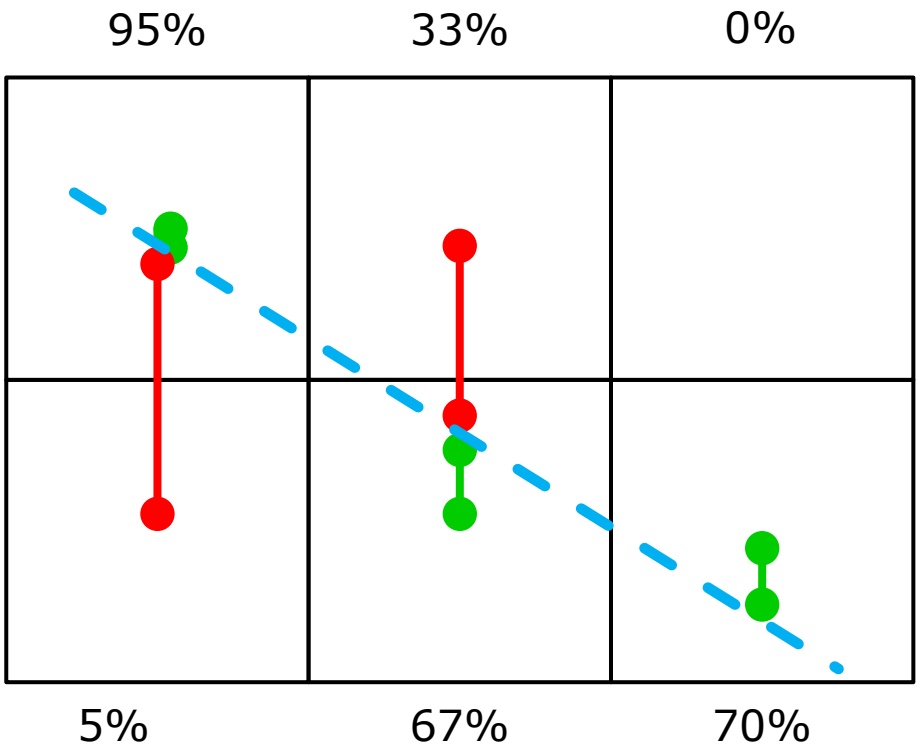


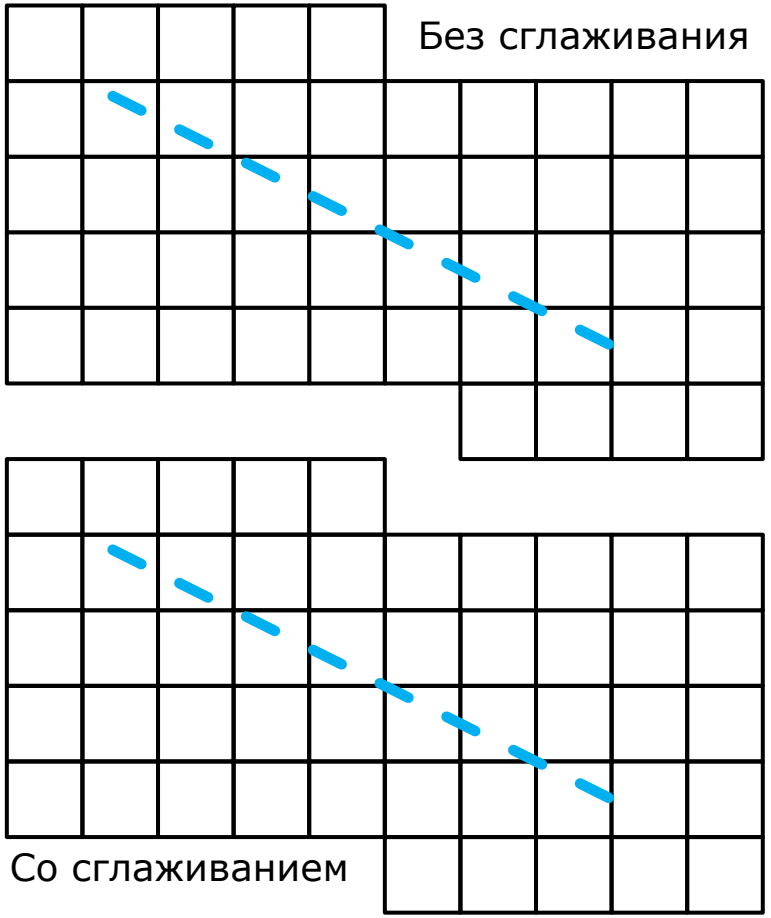




3. Алгоритм By (Xiaolin Wu)

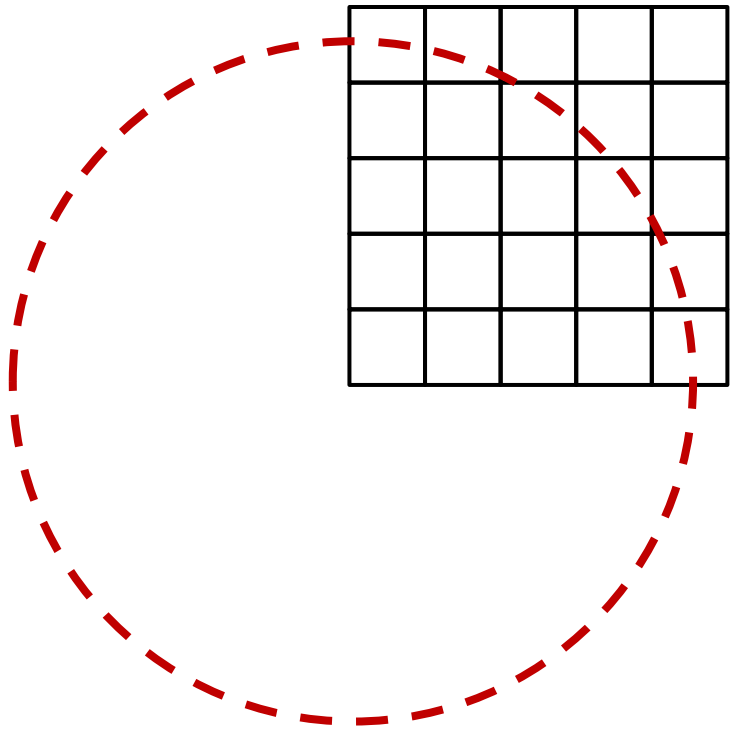
Интенсивность закрашивания *пропорциональна* расстоянию до отрезка



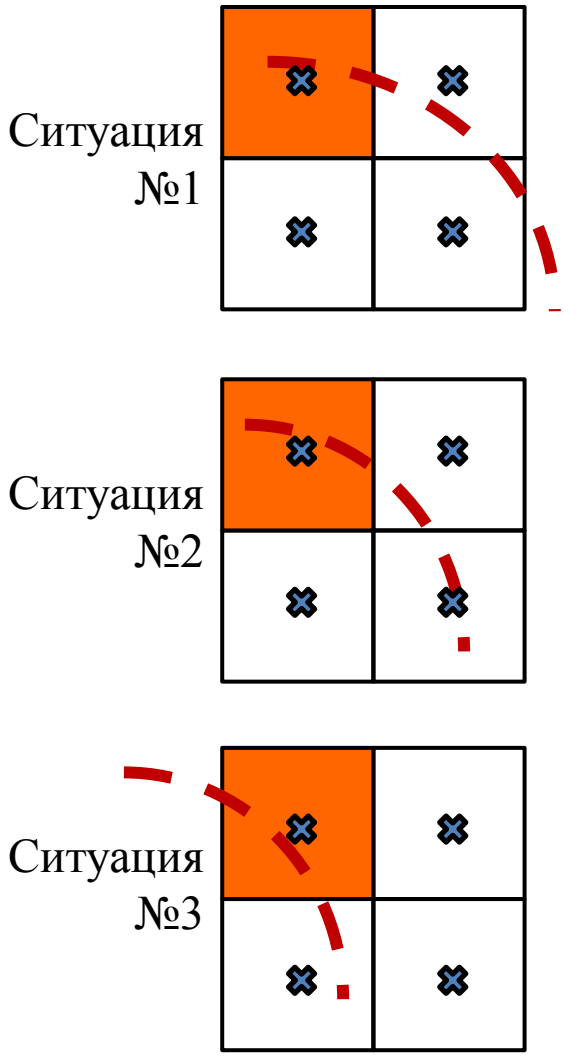


Растеризация окружности

- **ошибка** – разница между радиусом и расстоянием между центром окружности и серединой пикселя (ячейки)



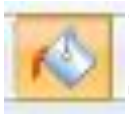
Возможные ситуации выбора следующего пикселя



Задача заполнения цветом

Понятия:

1) **Заливка**



требуется перекрасить пиксели одного цвета другим цветом

2) **Затравка**

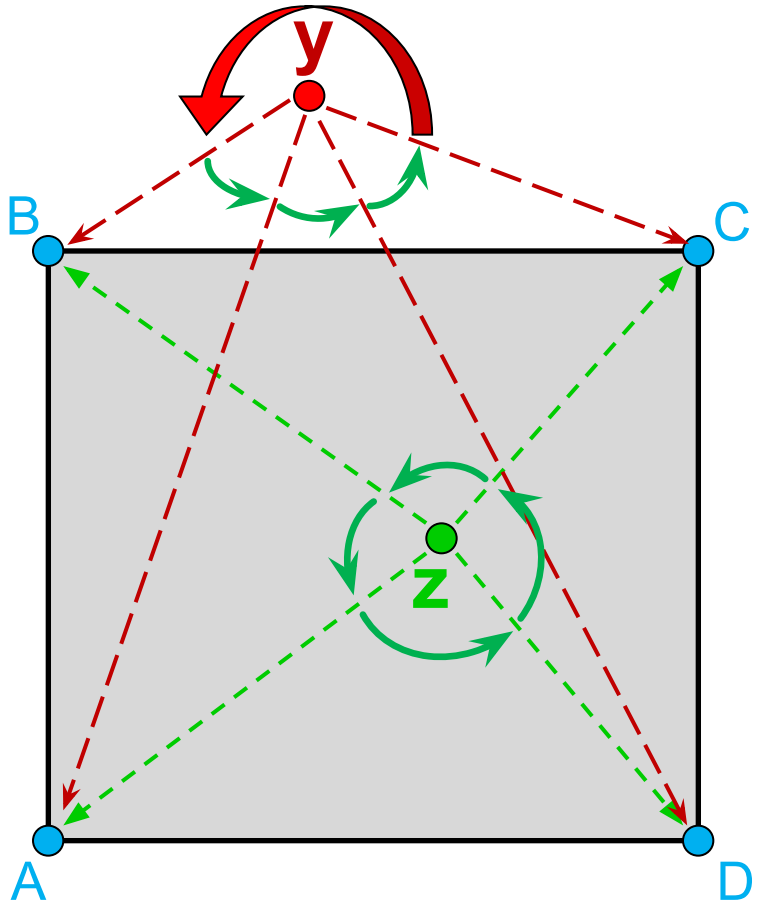
требуется закрасить нужным цветом фигуру, ограниченную граничными пикселями

Алгоритмы:

1) Определения принадлежности точки многоугольнику
   - расчет суммы углов (площадей)
   - расчет числа пересечений (оборотов) луча с ребрами (чет – внутри, нечет - снаружи)
   - …
2) Построение линий (рядов) пикселей, расположенных горизонтально или вертикально между ребрами

1. Алгоритм поточечной закраски

1) Перебираем все пиксели (точки) и закрашиваем, если они лежат внутри



<u>Задача определения принадлежности точки многоугольнику</u>

Для **любой** *внутренней точки* справедливо:

- суммарный угол точки с вершинами равен 360° (метод углов):

  AzD + BzA + CzB + DzC = 360°

- сумма всех треугольников, образованных точкой и сосмежными вершинами, равна площади фигуры (метод площадей) :

  AZD + BZA + CZB + DZC = ABCD

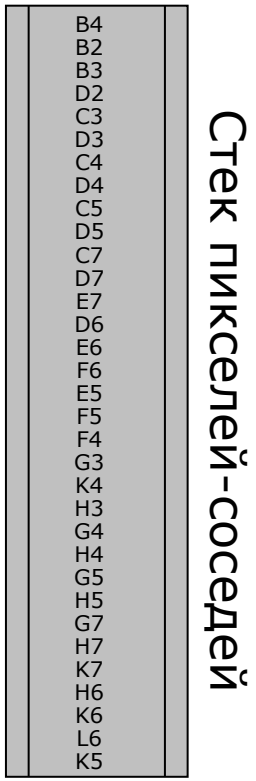
2. Алгоритм закраски соседей

1) Выбираем пиксель
2) Находим всех его соседей и заносим в стек
3) Закрашиваем пиксель

Варианты связности:

- по 4 соседа
- по 8 соседей

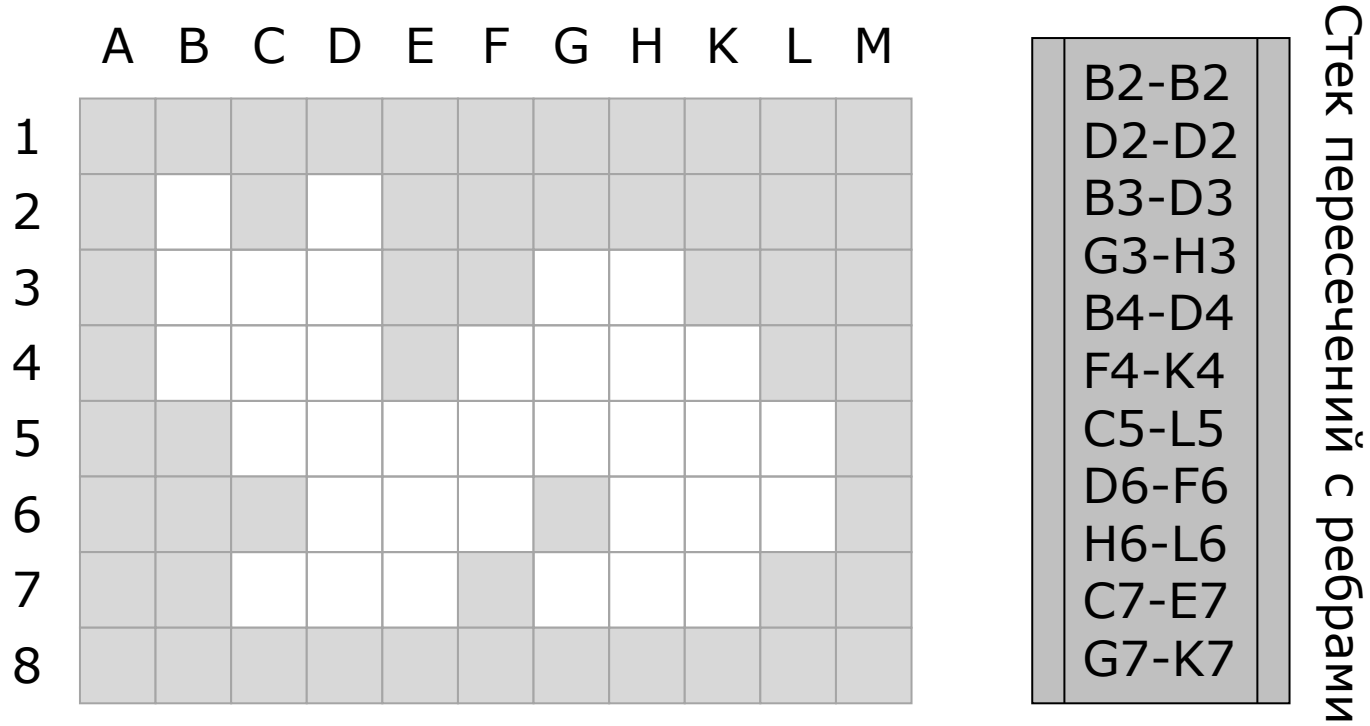
	A	B	C	D	E	F	G	H	K	L	M
1											
2		B2		D2							
3		B3	C3	D3			G3	H3			
4		B4	C4	D4		F4	G4	H4	K4		
5			C5	D5	E5	F5	G5	H5	K5	L5	
6				D6	E6	F6		H6	K6	L6	
7			C7	D7	E7		G7	H7	K7		
8											



Стек пикселей-соседей

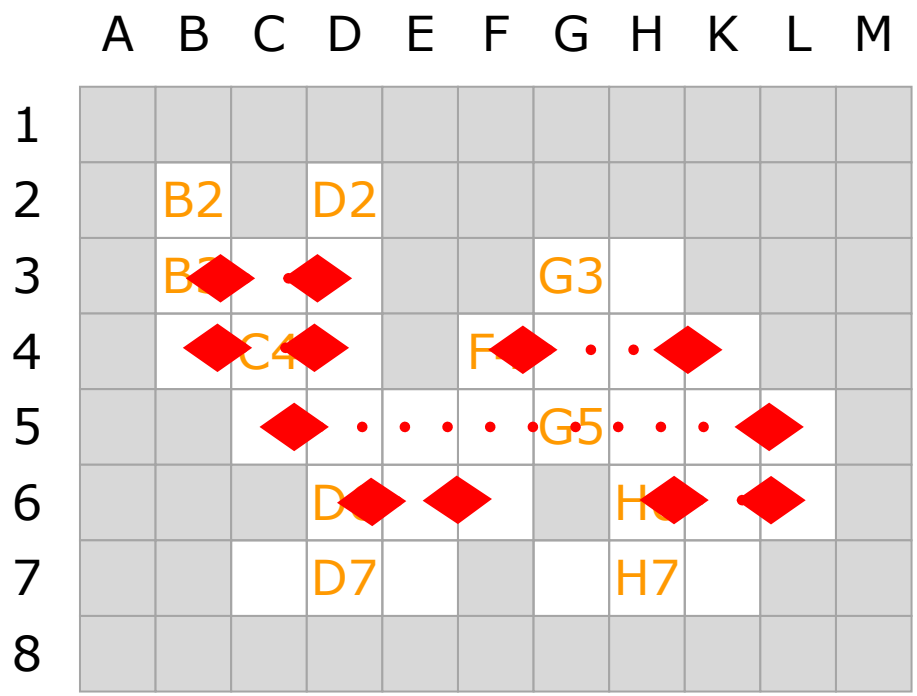
3. Алгоритм построчной закраски

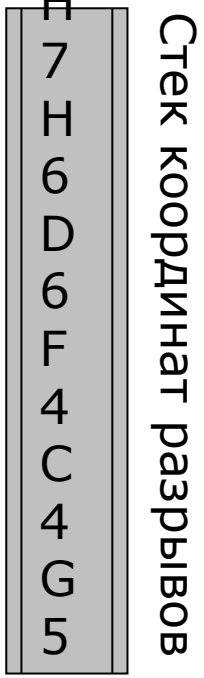
1) Запоминаем все пересечения ребер многоугольника с пикселями
2) Последовательно их выбираем и закрашиваем в этих пределах



4. Алгоритм закраски сериями

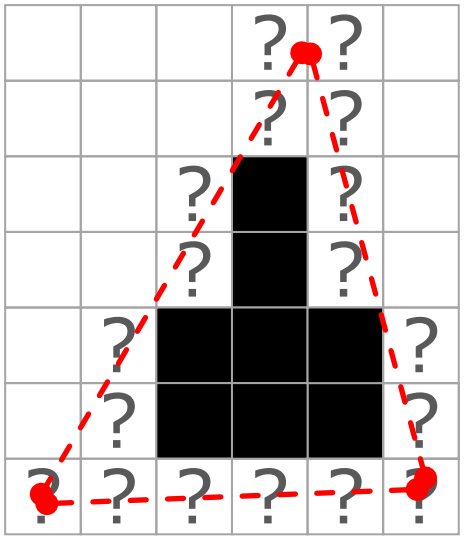
1) Выбираем и закрашиваем пиксель
2) Закрашиваем серию влево/вправо
3) Поднимаемся выше/ниже и запоминаем в стеке координаты пикселей-разрывов



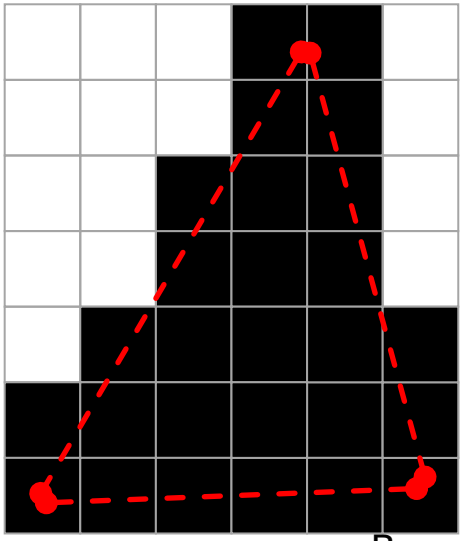


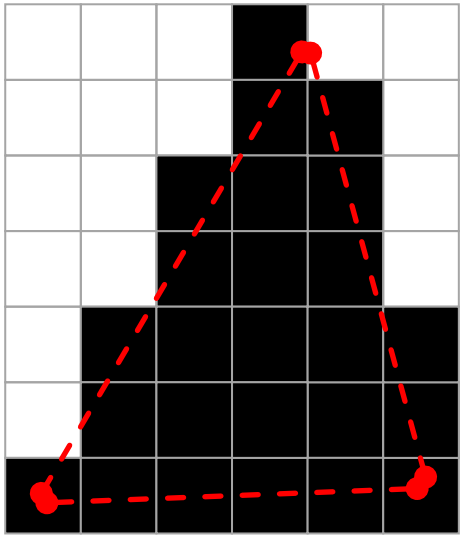
Стек координат разрывов

Растеризация треугольника со сглаживанием

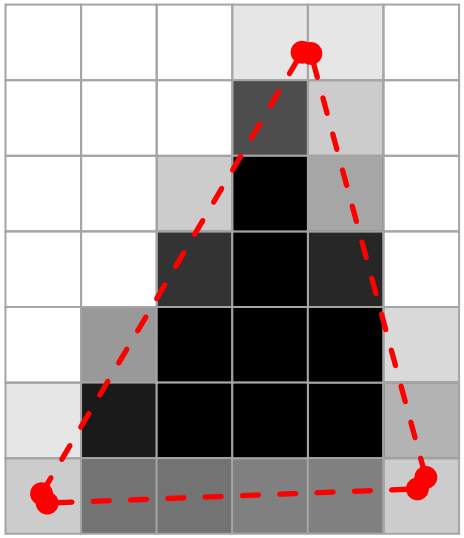


Вопрос:
как быть
с *граничными пикселями*
(которые только
частично попали
внутрь треугольника)?

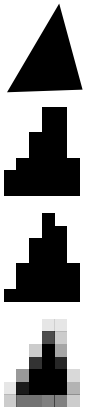




Вариант 1: закрашивать
полностью или **по Брезенхему**



Вариант 2:
закрашивать
с яркостью, зависящей
от площади попадания



Растеризация буквы S со сглаживанием