Алгоритмы растеризации

- **Растеризация** — это перевод изображения, описанного векторным форматом в пиксели или точки, для вывода на дисплей или принтер.
- Простейшие растровые алгоритмы:

- переведение идеального объекта (отрезка, окружности и др.) в их растровые образы;

- обработка растровых изображений.

Понятие связности

Определение связной области:

Множество пикселей, у каждого пикселя которого есть хотя бы один сосед, принадлежащий данному множеству.

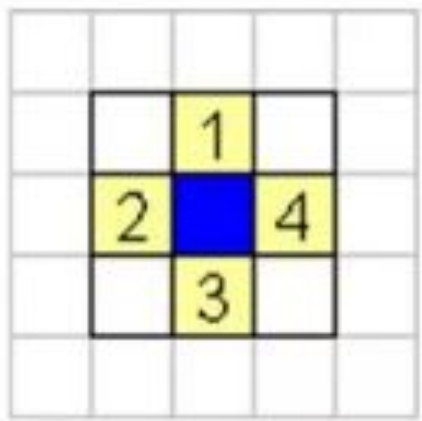
- **8 – связность**, когда пикселы считаются соседними, если их **x** и **y** координаты отличаются не более чем на единицу, т.е.

$$|x_1 - x_2| + |y_1 - y_2| \leq 1.$$

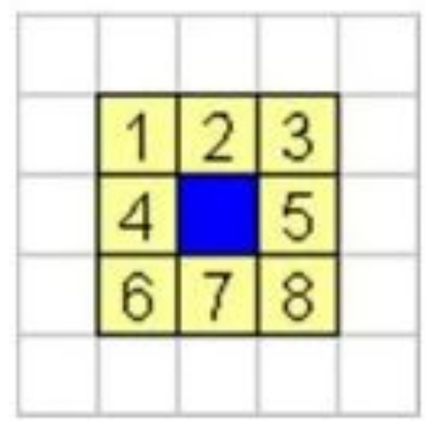
- **4 – связность**, когда пикселы считаются соседними, если либо их **x**, либо **y** координаты отличаются не более чем на единицу, т.е.

$$|x_1 - x_2| \leq 1; |y_1 - y_2| \leq 1.$$

Соседи пик



4-связность



8-связность

Понятие 4-связности является более сильным, чем 8-связность:

любые два 4-связных пиксела являются и 8-связными, но не наоборот

- **Линией** на растровой сетке выступает набор пикселов $P_1, P_2, \ldots, P_n$, где любые два пиксела P_k, P_{k+1} являются соседними.
- Простые элементы, из которых складываются сложные объекты, будем называть **графическими примитивами**.
- Простейшим растровым графическим примитивом является **пиксел**.
- Сложными графическими примитивами являются линии и фигуры.

Задача построения **растрового изображения отрезка**, соединяющего точки (x_1, y_1) и (x_2, y_2).

- $y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) = kx + b, x \in [x_1, x_2].$

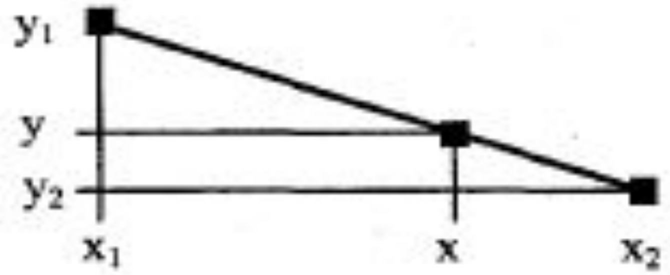
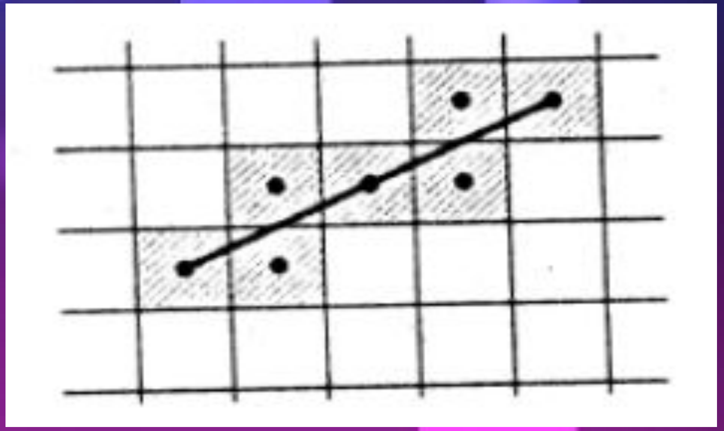


Рис. Отрезок прямой



Простейший алгоритм растрового представления отрезка

- Будем считать, что $0 \le y_2 - y_1 \le x_2 - x_1$

```
// File Line1.cpp
void Line ( int x1, int y1, int x2,
int y2, int color )
{
double k =
((double)(y2-yl))/(x2-xl):
double b = y1 - k*x1;
for ( int x = x1; x <= x2; x++ )
putpixel ( x, round ( k*x + b ),
color );
}
```

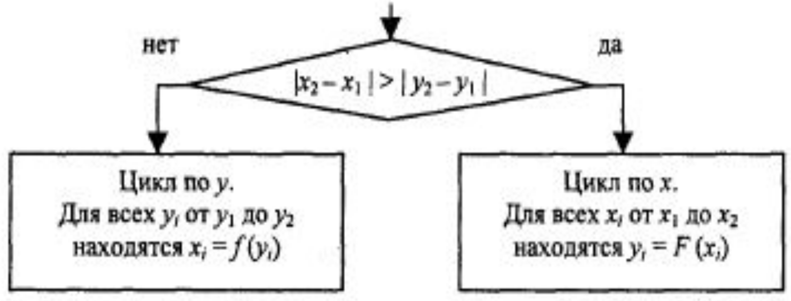


Рис. Алгоритм вывода отрезка прямой линии

Процесс определения пикселов, наилучшим образом аппроксимирующих заданный отрезок, называется **разложением в растр**.

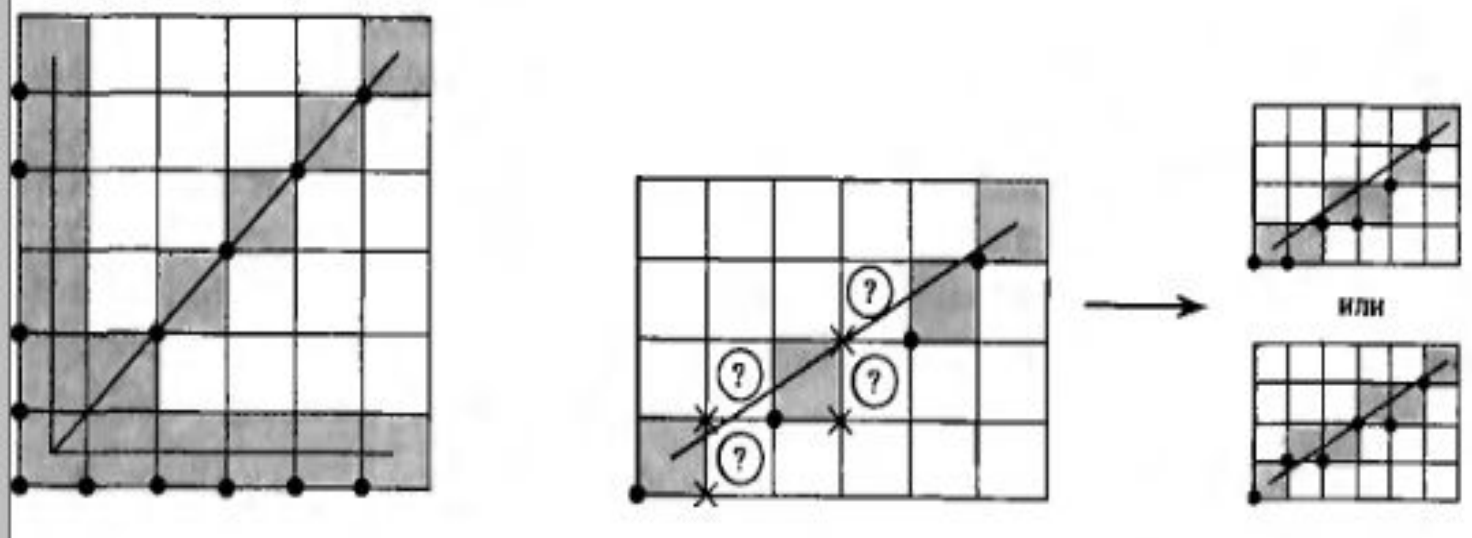
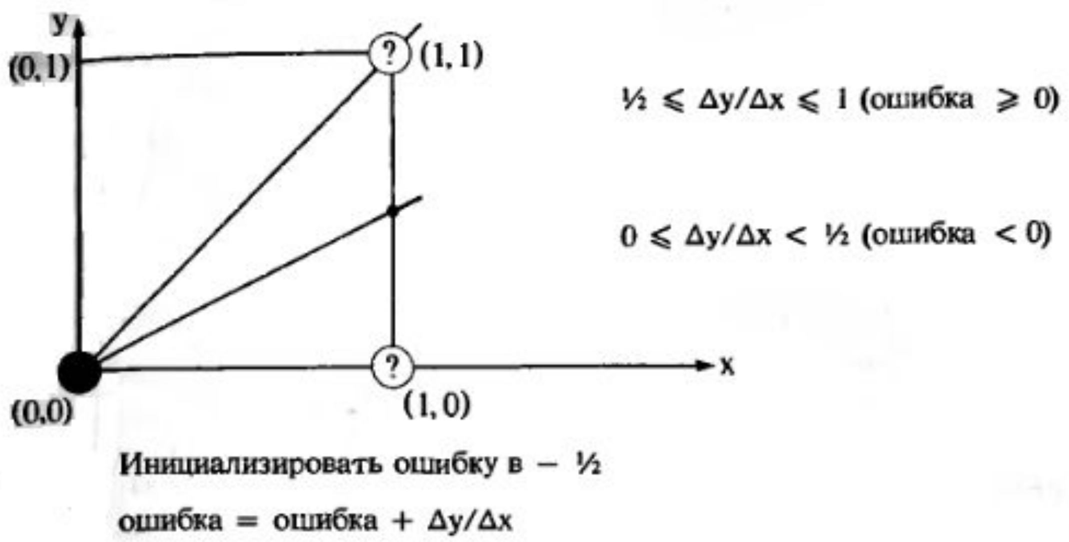


Рис. Разложение в растр отрезков прямых.

- В 1965 году Брезенхейм предложил простой целочисленный алгоритм для растрового построения отрезка, первоначально предназначенный для использования в графопостроителях.
- Алгоритм выбирает оптимальные растровые координаты для представления отрезка.
- В процессе работы одна из координат – либо **x**, либо **y** изменяется на единицу.
- Изменение другой координаты (либо на нуль, либо на единицу) зависит от расстояния между действительным положением отрезка и ближайшими координатами сетки. Такое расстояние назовем **ошибкой**.

- Угловой коэффициент $k \in [0,1]$.
- Если $k \geq \frac{1}{2}$, то точка пересечения с прямой $x=1$ будет расположено ближе к прямой $y=1$, чем к прямой $y=0$.
- Следовательно, точка растра (1,1) лучше аппроксимирует ход отрезка, чем точка (1,0).
- Если $k < \frac{1}{2}$, то верно обратное.



Основная идея алгоритма Брезенхема.

- Не все отрезки проходят через точки растра.
- На рис. иллюстрируется отрезок с $k = \frac{3}{8}$.

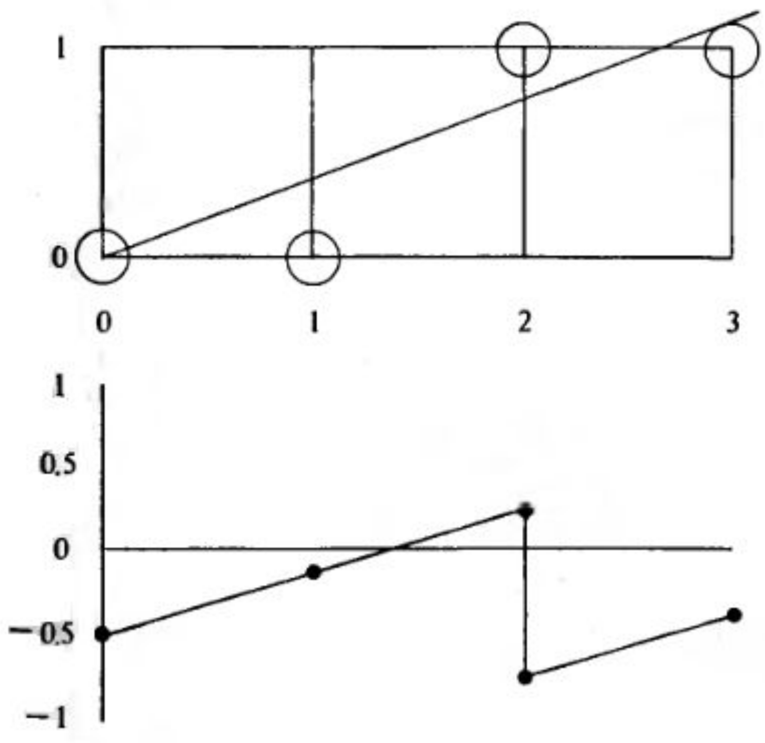


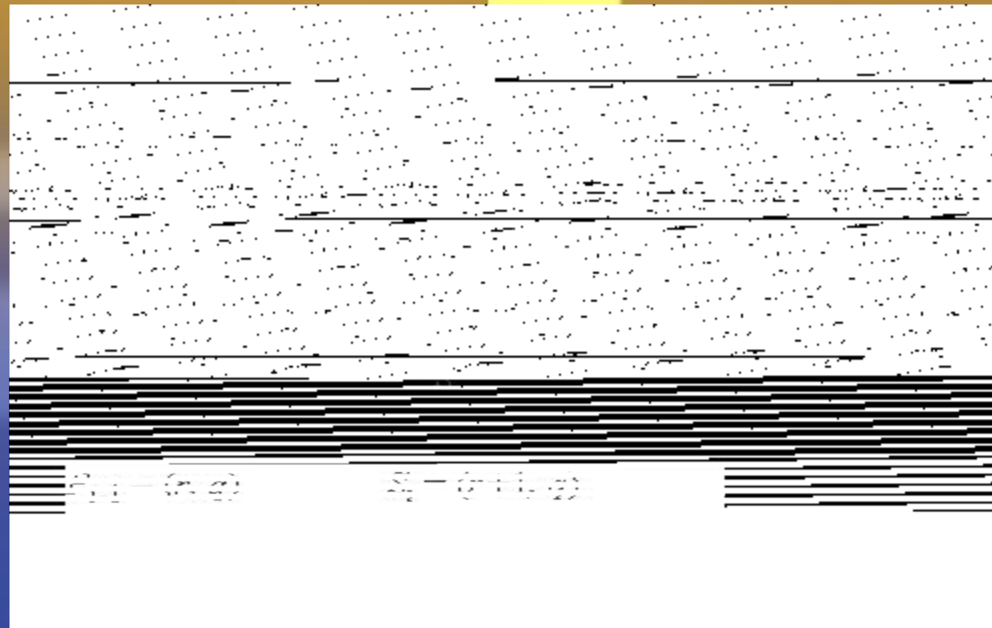
Рис. График ошибки в алгоритме Брезенхема.

- Отрезок проходит через точку растра (0,0) и последовательно пересекает три пиксела.
- Также иллюстрируется вычисление ошибки при представлении отрезка дискретными пикселами.

Алгоритмы построения отрезка имеют ряд недостатков:

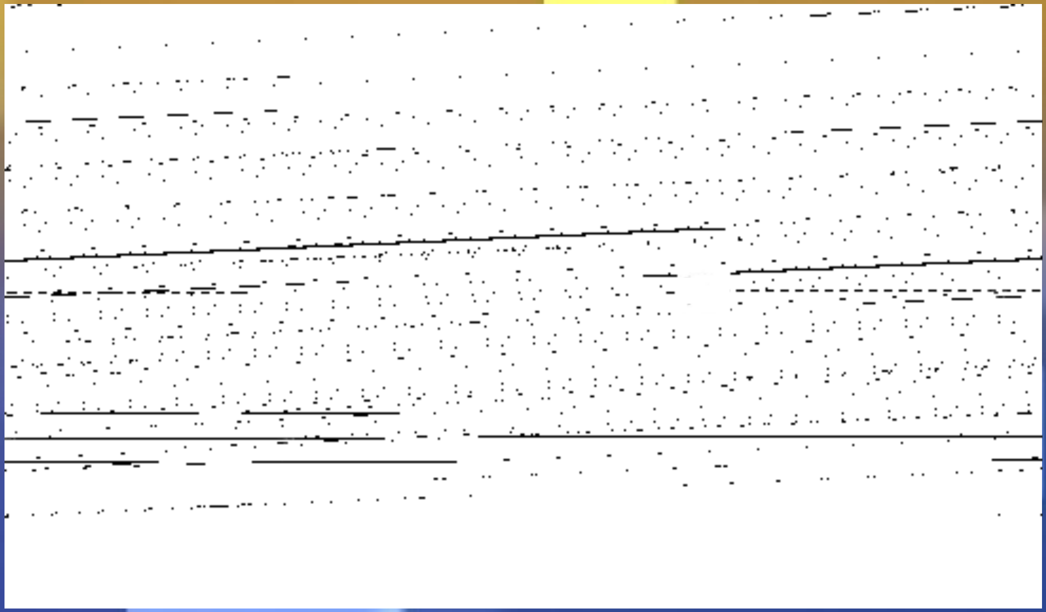
1. выполняют операции над числами с плавающей точкой, а желательно было бы работать с целочисленной арифметикой;

2. на каждом шаге выполняется операция округления, что также снижает быстродействие.



- Пиксель P_{i-1} уже найден как ближайший к реальному отрезку.
- Требуется определить, какой из пикселов (T_i или S_i) будет установлен следующим.

- В алгоритме используется управляющая переменная d_i, которая на каждом шаге пропорциональна разности между S и T.
- Если $S < T$, то S_i ближе к отрезку, иначе выбирается T_i.



- Пусть отрезок проходит из точки (x_1, y_1) в точку (x_2, y_2).
- Исходя из начальных условий, точка (x_1, y_1) ближе к началу координат.
- Перенесем оба конца отрезка с помощью преобразования T($-x_1$, $-y_1$), так чтобы первый конец отрезка совпал с началом координат.
- Начальной точкой отрезка стала точка (0, 0), конечной точкой – (dx, dy), где $dx = x_2 - x_1$, $dy = y_2 - y_1$

- Уравнение прямой $y = x\frac{dy}{dx}$.
- Обозначим $P_{i\text{-}1}$ (r, q).
- Тогда $S_i = (r+1, q)$ и $T_i = (r+1, q+1)$.
- Из подобия треугольников $\frac{dy}{dx} = \frac{S+q}{r+1}$.
- Тогда $\mathrm{S} = \frac{dy}{dx}(r + 1) - q$.
- $T = 1 - S$.
- $T = 1 - \frac{dy}{dx}(r + 1) + q$.
- $\mathrm{S} - T = 2\frac{dy}{dx}(r + 1) - 2q - 1$.
- Поскольку $r = x_{i\text{-}1}$ и $q = y_{i\text{-}1}$, обозначим $d_i = dx\,(S - T)$
- $d_i = 2\,x_{i\text{-}1}\,dy - 2\,y_{i\text{-}1}\;dx + 2\,dy - dx$.

- $d_i = 2\,x_{i-1}\,dy - 2\,y_{i-1}\,dx + 2\,dy - dx.$
- $d_{i+1} = 2\,x_i\,dy - 2\,y_i\,dx + 2\,dy - dx.$
- $d_{i+1} - d_i = 2\,\mathrm{dy}\,(x_i - x_{i-1}) - 2\,\mathrm{dx}\,(y_i - y_{i-1}).$
- Известно, что $x_i - x_{i-1} = 1$, тогда
- $d_{i+1} - d_i = 2\,\mathrm{dy} - 2\,\mathrm{dx}\,(y_i - y_{i-1}).$

или

- $d_{i+1} = d_i + 2\,\mathrm{dy} - 2\,\mathrm{dx}\,(y_i - y_{i-1})$ - итеративная формула вычисления управляющего коэффициента d_{i+1} по предыдущему значению d_i.

- Если $d_i \geq 0$, тогда выбирается T_i и

$$y_i = y_{i-1} + 1,\ d_{i+1} = d_i + 2\,(dy - dx).$$

- Если $d_i < 0$, тогда выбирается S_i и

$$y_i = y_{i-1} \text{ и } d_{i+1} = d_i + 2\,dy.$$

- Начальные значения d_1 с учетом того, что

$$(x_0, y_0) = (0, 0),\ d_1 = 2\,dy - dx.$$

- Преимущество: *для работы алгоритма требуются минимальные арифметические возможности: сложение, вычитание и сдвиг влево для умножения на 2.*

```
void MyLine(int x1, int y1, int x2, int y2, int c)
{
int dx, dy, inc1, inc2, d, x, y, Xend;
 dx = abs(x2 - x1);
 dy = abs(y2 - y1);
 d = dy << 1 - dx;
 inc1 = dy << 1;
 inc2 = (dy - dx) << 1;
 if (x1>x2)
   {
   x = x2;
   y = y2;
   Xend = x1;
   }
 else
   {
   x = x1;
   y = y1;
   Xend = x2;
   };
 putpixel(x, y, c);
 while (x < Xend)
  {
  x++;
  if (d < 0) d = d + inc1;
    else
{
y++;
d = d + inc2;
};
  putpixel(x, y, c);
  };
}
```

Если *dy* > *dx*, то необходимо будет использовать этот же алгоритм, но пошагово увеличивая *y* и на каждом шаге вычислять *x*.

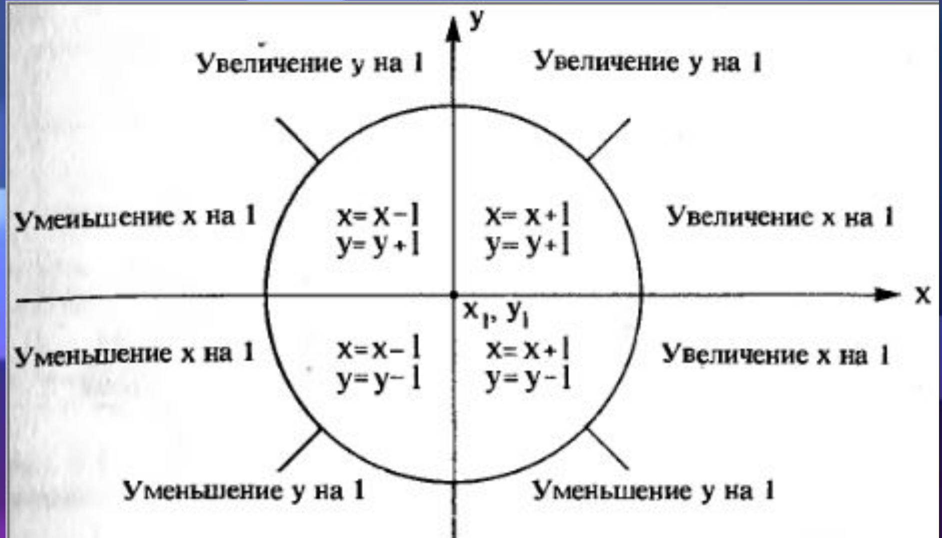


Рис. Разбор случаев для обобщенного алгоритма Брезенхема.

Методы устранения ступенчатости

Основная причина появления лестничного эффекта заключается в том, что отрезки, ребра многоугольника, цветовые границы и пр. имеют непрерывную природу, тогда как растровые устройства дискретны.

Лестничный эффект проявляется:

1) при визуализации мелких деталей;

2) при прорисовке ребер и границ;

3) при анимации мелких деталей.

Метод увеличения частоты выборки

- Каждый пиксель делится на **подпиксели** в процессе формирования растра более высокого разрешения.
- В некоторой степени можно получить лучшие результаты, если рассматривать больше подпикселов и учитывать их влияние с помощью весов при определении атрибутов.

+	+
+	+

Увеличение разрешения в два раза

+	+	+	+
+	+	+	+
+	+	+	+
+	+	+	+

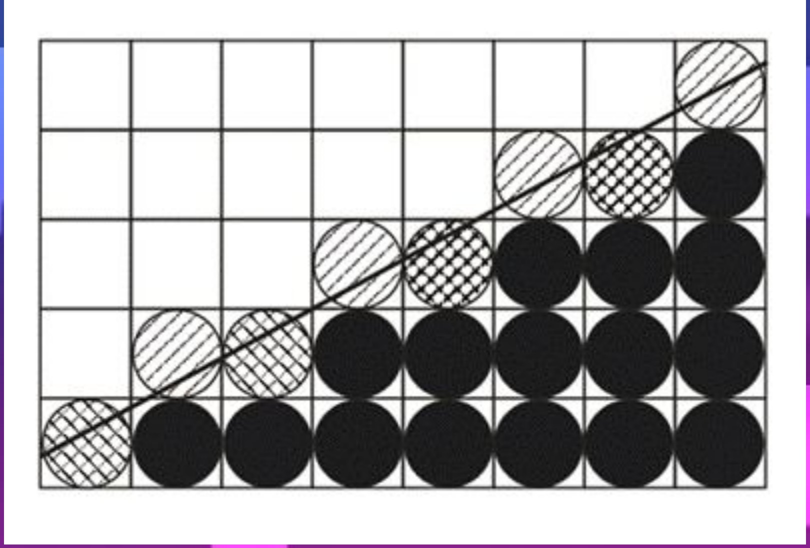
Увеличение разрешения в четыре раза

1	2	1
2	4	2
1	2	1

1	2	3	4	3	2	1
2	4	6	8	6	4	2
3	6	9	12	9	6	3
4	8	12	16	12	8	4
3	6	9	12	9	6	3
2	4	6	8	6	4	2
1	2	3	4	3	2	1

Метод, основанный на использовании полутонов

Интенсивность пикселя на ребре устанавливается пропорционально площади части пикселя, находящегося внутри многоугольника.



Растровое представление окружности

- $$x^2 + y^2 = R^2$$

Генерируется 1/8 часть окружности.

Остальные ее части получаем последовательными отражениями.

Пусть центр окружности и начальная точка находятся точно в точках растра.

Выбираем генерацию по часовой стрелке с началом в точке *x=0, y=R*

y — монотонно убывающая функция аргумента *x*.

m_H — горизонтально вправо,
m_D — по диагонали вниз и вправо,
m_V — вертикально вниз.

$$m_H = \left|(x_i + 1)^2 + (y_i)^2 - R^2\right|$$
$$m_D = \left|(x_i + 1)^2 + (y_i - 1)^2 - R^2\right|$$
$$m_V = \left|(x_i)^2 + (y_i - 1)^2 - R^2\right|$$

Алгоритм выбирает пиксел соответствующий $\min\{m_H, m_D, m_V\}$.

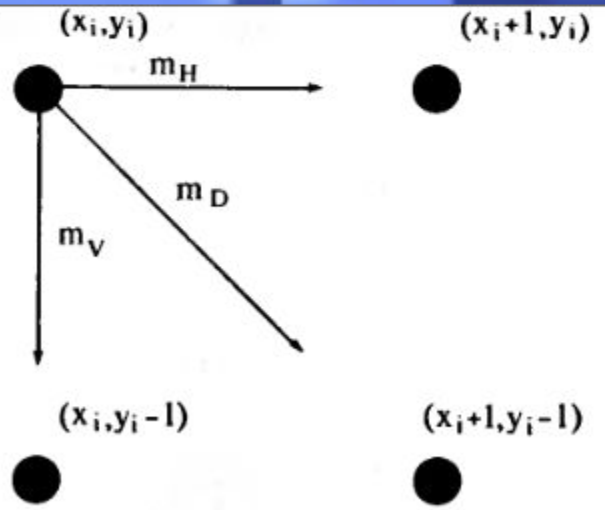


Рис. Выбор пикселов в первом квадранте.

В окрестности точки (x_i, y_i) возможны только пять типов пересечений окружности и сетки растра

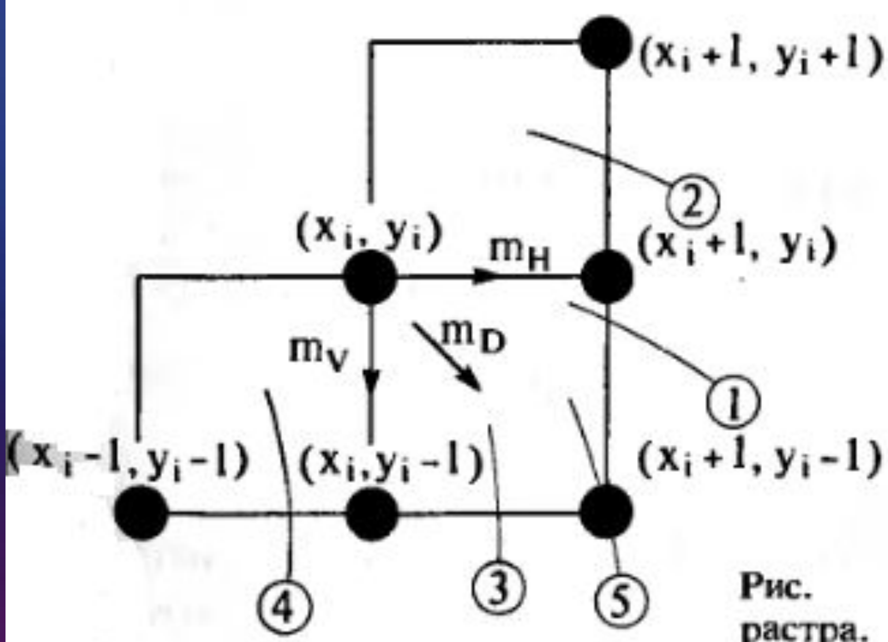


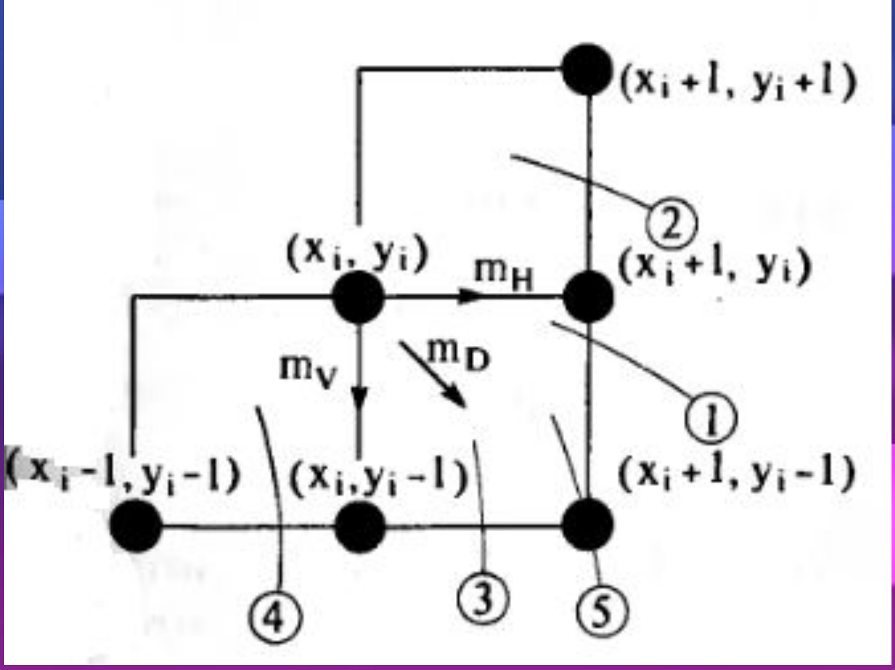
Рис. Пересечение окружности и сетки растра.

Обозначим

$$\Delta_i = (x_i + 1)^2 + \left(y_i - 1\right)^2 - R^2$$

При $\Delta_i < 0$ диагональная точка $(x_i + 1, y_i - 1)$ находится внутри реальной окружности, т.е. это случаи 1 или 2.

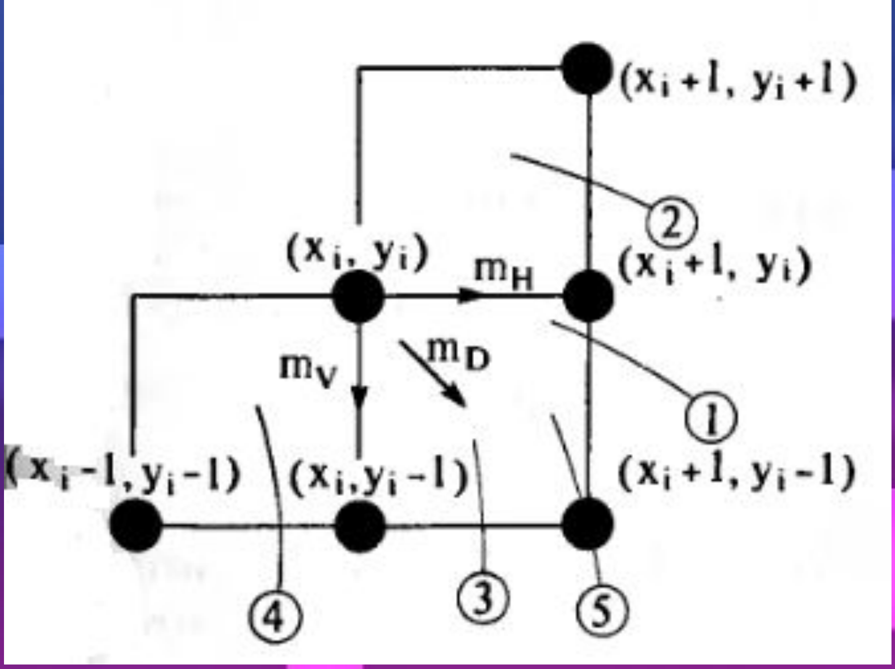
Выбираем либо направление m_H (пиксел $(x_i + 1, y_i)$), либо направление m_D (пиксел $(x_i + 1, y_i - 1)$).



- Вычислим
$$\delta = \left|(x_i + 1)^2 + (y_i)^2 - R^2\right| - \left|(x_i + 1)^2 + (y_i - 1)^2 - R^2\right|$$

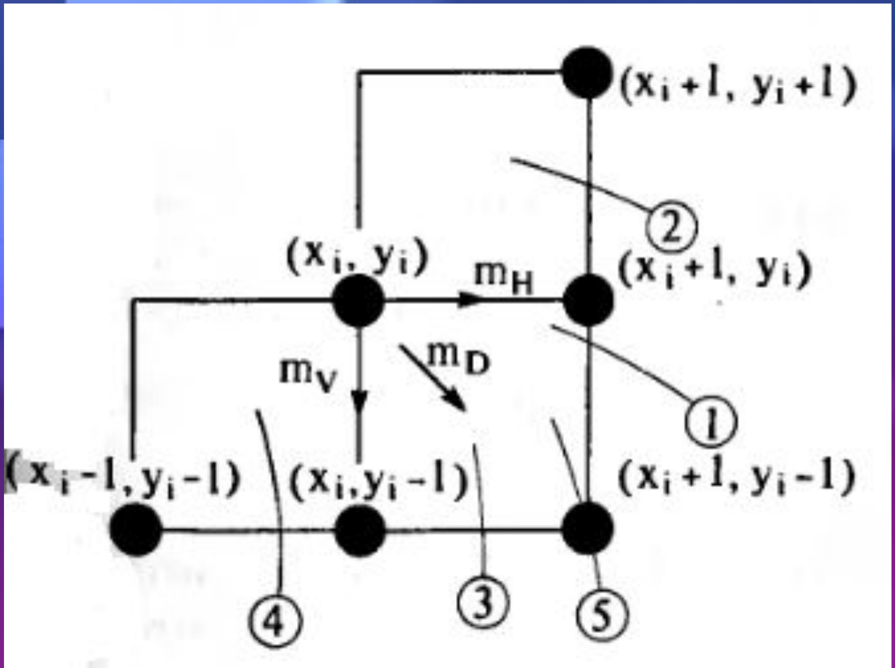
Если $\delta \le 0$, то выбираем направление m_H (пиксел $(x_i + 1, y_i)$).

Если $\delta > 0$, то выбираем направление m_D (пиксел $(x_i + 1, y_i - 1)$).



- При $\Delta_i > 0$ диагональная точка $(x_i + 1, y_i - 1)$ находится вне окружности, т.е. это случаи 3 или 4.

Выбираем либо направление m_V (пиксел $(x_i, y_i - 1)$), либо направление m_D (пиксел $(x_i + 1, y_i - 1)$).



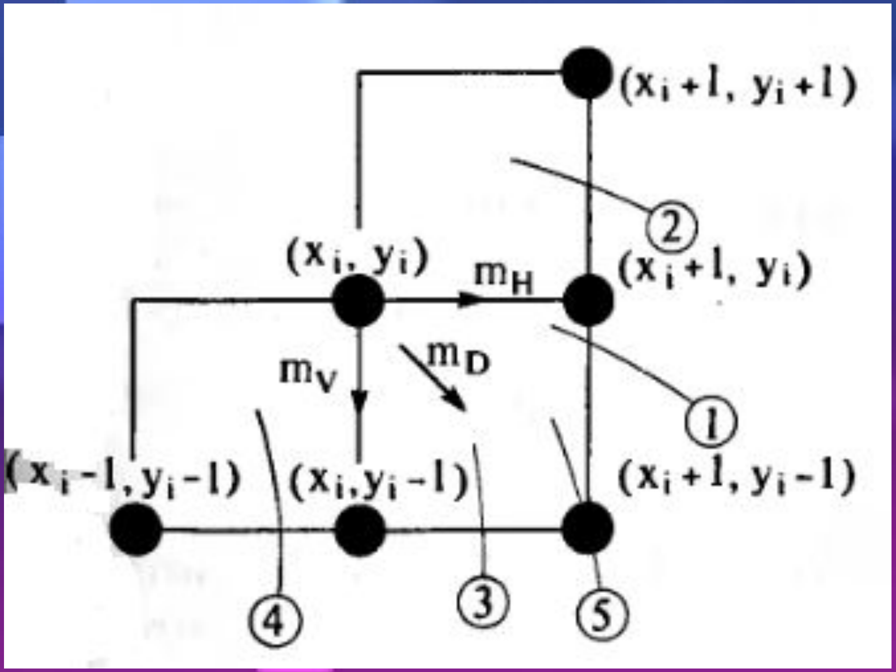
Вычислим

$$\delta' = \left|(x_i+1)^2+(y_i-1)^2-R^2\right| - \left|(x_i)^2+(y_i-1)^2-R^2\right|$$

Если $\delta' \leq 0$, то выбираем направление

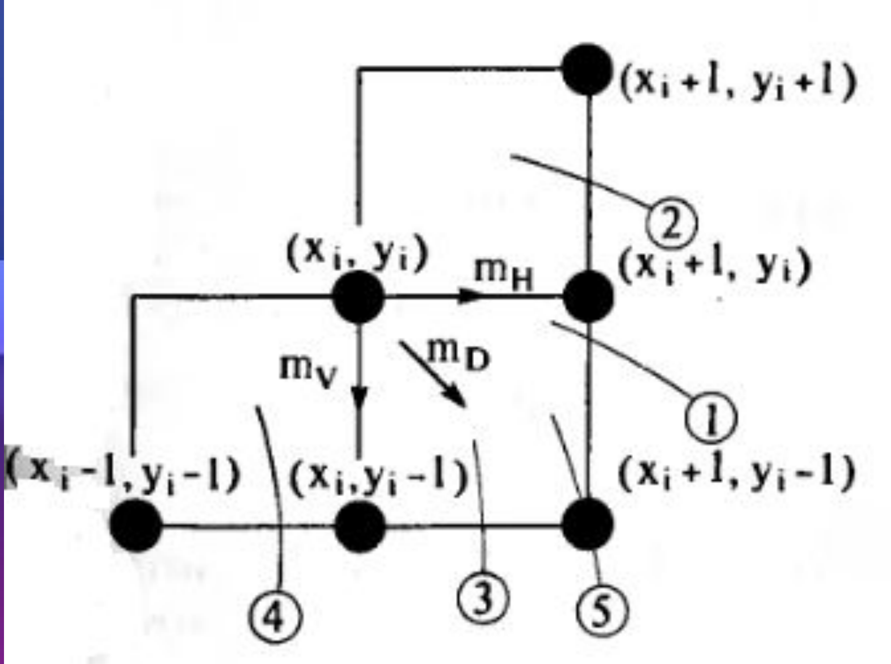
m_D (пиксел (x_i+1, y_i-1)).

Если $\delta' > 0$, то выбираем направление m_V (пиксел (x_i, y_i-1)).



- При $\Delta_i = 0$ диагональная точка $(x_i + 1, y_i - 1)$ находится на реальной окружности, т.е. это случай 5.

В этой ситуации $\delta > 0, \delta' < 0$, то выбираем направление m_D (пиксел $(x_i + 1, y_i - 1)$).



- Рекуррентные соотношения для реализации пошагового алгоритма:

направление m_H:

$x_{i+1} = x_i + 1, y_{i+1} = y_i, \Delta_{i+1} = \Delta_i + 2x_{i+1} + 1$

направление m_D:

$x_{i+1} = x_i + 1, y_{i+1} = y_i - 1,$
$\Delta_{i+1} = \Delta_i + 2x_{i+1} - 2y_{i+1} + 2$

направление m_V:

$x_{i+1} = x_i, y_{i+1} = y_i - 1, \Delta_{i+1} = \Delta_i - 2y_{i+1} + 1$

Окружность в I октанте сгенерирована.

II октант получается зеркальным отражением относительно прямой *y=x*.

Получаем I квадрант.

Отражаем полученную часть окружности относительно прямой *x=0*.

Получаем часть окружности во II квадранте.

Верхнюю часть окружности отражаем относительно прямой *y=0*.

Получаем окружность.

Двумерные матрицы соответствующих преобразований

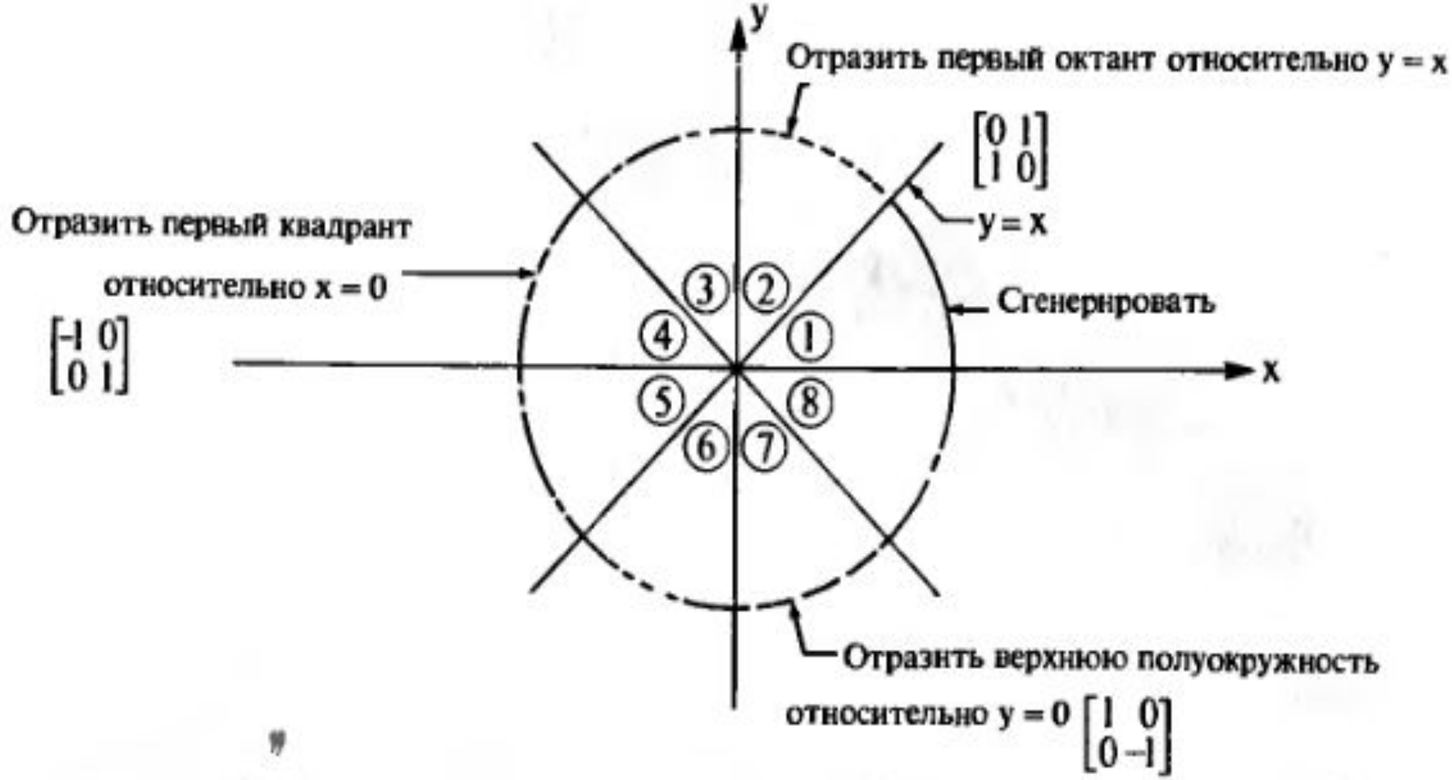
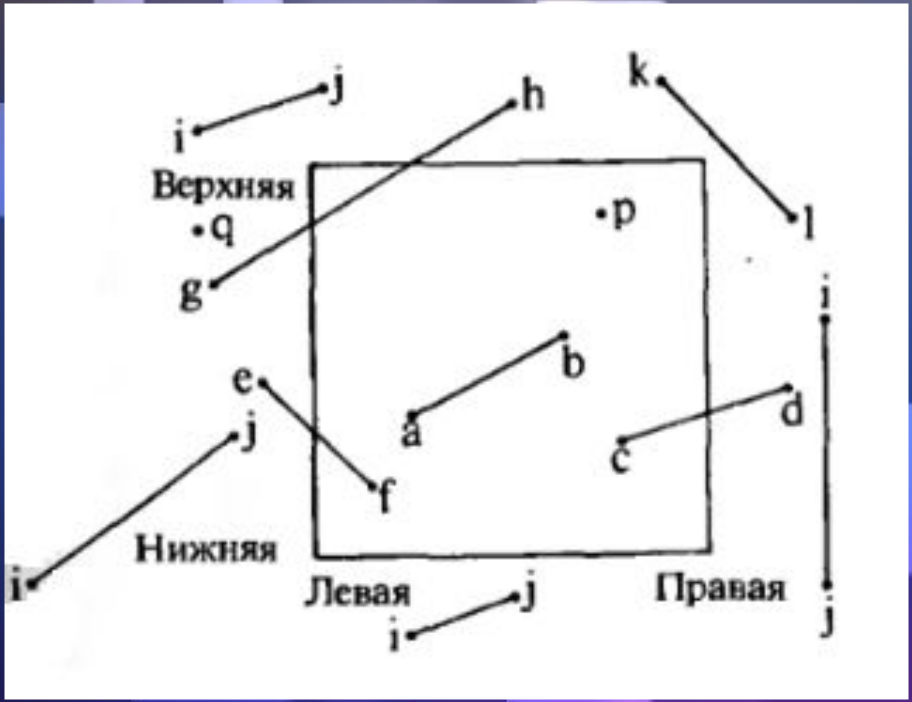


Рис. Генерация полной окружности из дуги в первом октанте.

Отсечение отрезка. Алгоритм Сазерленда'Кохена

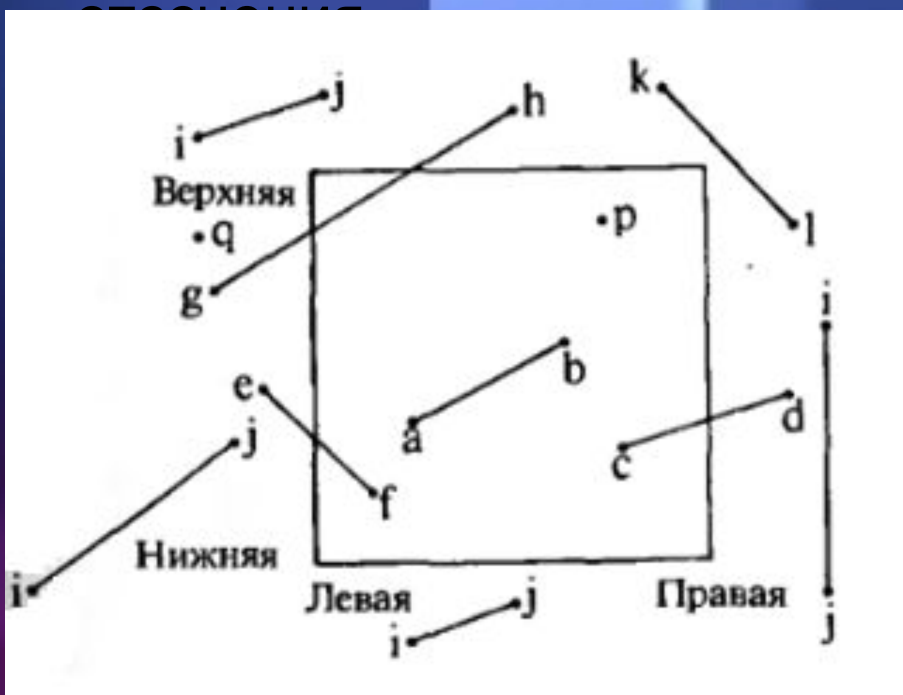


Точки, лежащие внутри отсекающего окна, удовлетворяют условию:

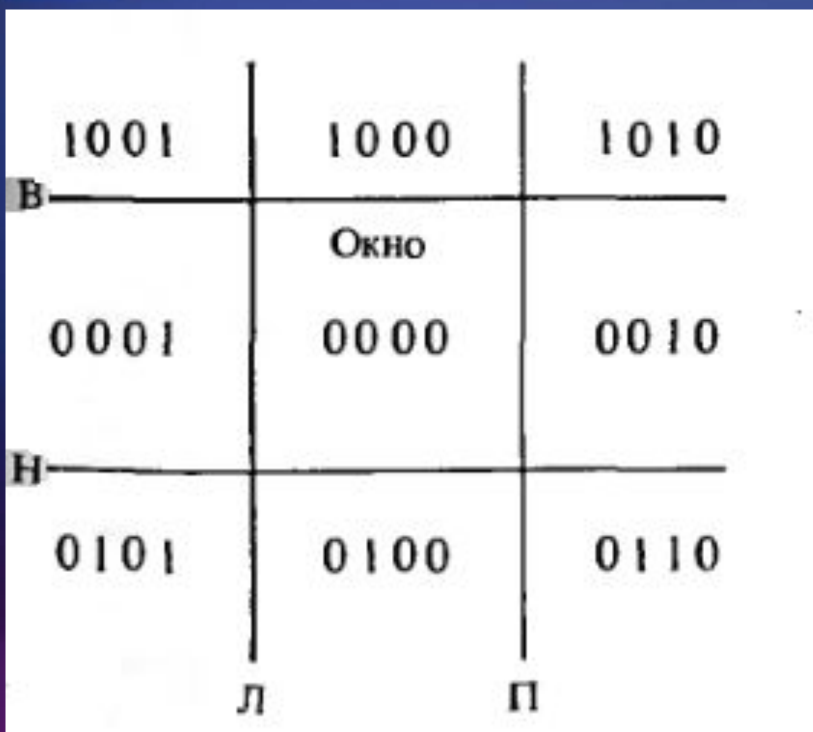
$$x_Л \le x \le x_П$$
$$y_Н \le y \le y_В$$

Цель алгоритма: определение тех точек отрезка, которые лежат внутри отсекающего окна.

- Отрезок называется **видимым** (лежащим внутри окна), если обе его концевые точки лежат внутри окна, например, отрезок ab.
- Если оба конца отрезка лежат справа, слева, выше или ниже окна, то этот отрезок называется **невидимым** (целиком лежит вне окна), например отрезок ij.
- Все остальные отрезки называются **частично видимыми** и к ним нужно применять алгоритмы отсечения



- для 1 бита – если точка левее окна
- для 2 бита – если точка правее окна
- для 3 бита – если точка ниже окна
- для 4 бита – если точка выше окна



u	v	u∧v
0	0	0
0	1	0
1	0	0
1	1	1

Алгоритм Сазерленда – Коэна для произвольного отрезка P_1P_2

- Для каждого отрезка P_1P_2 определить, не является ли он полностью видимым или может быть тривиально отвергнут как невидимый.
- Если P_1 вне окна, то продолжить выполнение, иначе поменять P_1 и P_2 местами.
- Заменить P_1 на точку пересечения P_1P_2 со стороной окна.

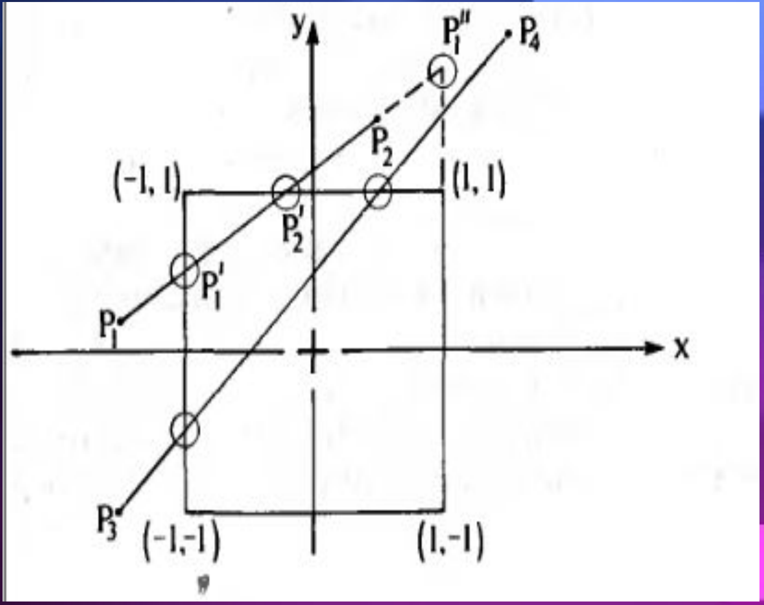
Таблица Коды концов отрезков

Отрезок (рис. 3.1)	Коды концов (рис. 3.2)	Результаты логического умножения	Примечания
ab	0000 0000	0000	Целиком видим
ij	0010 0110	0010	Целиком невидим
ij	1001 1000	1000	—"—
ij	0101 0001	0001	—"—
ij	0100 0100	0100	—"—
cd	0000 0010	0000	Частично видим
ef	0001 0000	0000	—"—
gh	0001 1000	0000	—"—
kl	1000 0010	0000	Целиком невидим

Пример

Координаты концевых точек
$P_1\left(-\frac{3}{2},\frac{1}{6}\right), P_2\left(\frac{1}{2},\frac{3}{2}\right), P_3\left(-\frac{3}{2},-1\right), P_4(\frac{3}{2},2)$.
Окно $P=\{-1\le x\le 1, -1\le y\le 1\}$.

$P_1: 0001, P_2: 1000 \Longrightarrow P_1P_2$ - частично видимым $\Longrightarrow P_1$ лежит вне окна.
Пересечение P_1P_2 с x=-1 в $P_1{}'(-1,\frac{1}{2})$.
Заменим точку P_1 на точку $P_1{}'$.
Отрезок P_1P_2, где $P_1(-1,\frac{1}{2})$.



$P_1: 0000, P_2: 1000 \Longrightarrow P_1P_2$ - частично видимым $\Longrightarrow P_1$ лежит внутри окна $\Longrightarrow$ меняем точки P_1, P_2 местами.
Отрезок P_1P_2 , где $P_1\left(\frac{1}{2},\frac{3}{2}\right), P_2(-1,\frac{1}{2})$.
Пересечение P_1P_2 с y=1 в $P_1{}'(-\frac{1}{4},1)$.
Заменим точку P_1 на точку $P_1{}'$.
Отрезок P_1P_2, где $P_1(-\frac{1}{4},1)$.

$P_1: 0000, P_2: 0000 \Longrightarrow P_1P_2$ - полностью видим.