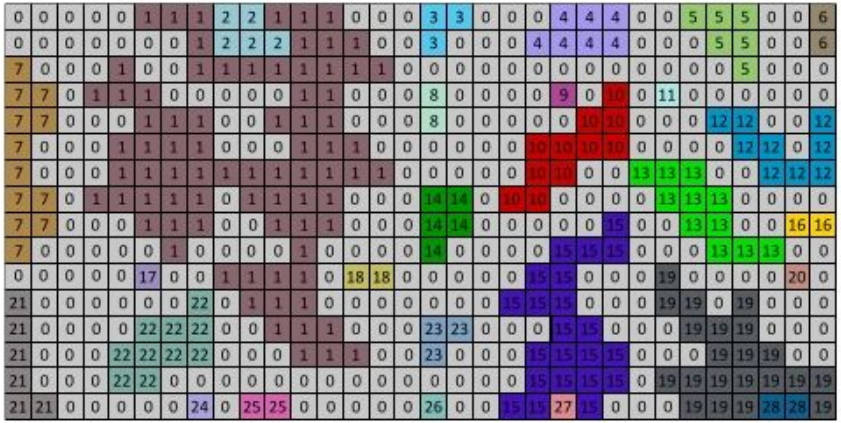
Алгоритм поиска границ Кэнни (Canny Edge). Разметка связных областей (Connected Components Labeling)



Цифровая обработка изображений
Ассистент кафедры КСАИТ
Петровец Александр Александрович

Фильтрация

В общем смысле фильтрация - преобразование изображения в локальной области в другое изображение посредством некой функции:

$$I'(x,y) = H(I(x,y))$$

Линейная

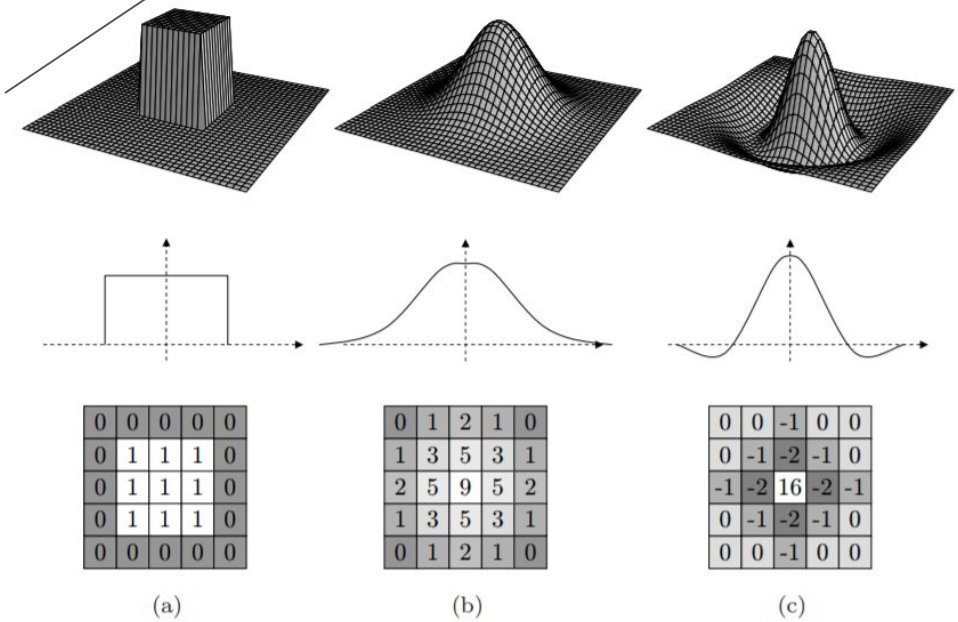
Нелинейная

Вообще говоря и локальную бинаризацию с эквализацией можно назвать фильтрацией

Виды линейных фильтров

- Усреднение (box filter) - все компоненты равны
- Фильтр Гаусса (сглаживание по Гауссу) - функция плотности двумерного нормального распределения с нулевым средним
- Фильтры Превитта, Собеля, Лапласа (Mexican Hat) - могут быть и отрицательные компоненты

Размытие (low-pass)



Резкость - выделение границ (high-pass)

Градиент изображения

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \approx f(x+1) - f(x) \quad (h=1)$$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

Вычисление первой производной цифрового изображения основано на различных дискретных приближениях двумерного градиента

Градиент изображения

$$f'(x) = \lim_{h->0} \frac{f(x+h) - f(x)}{h} \approx f(x+1) - f(x) \quad (h=1)$$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

$$magn(\nabla f) = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} = \sqrt{M_x{}^2 + M_y{}^2}$$

Модуль градиента (магнитуда)

$$dir(\nabla f) = \tan^{-1}(M_y / M_x)$$

Направление градиента (угол)

Дискретные апроксимации первой производной

$$f'(x) = \lim_{h->0} \frac{f(x+h) - f(x)}{h} \approx f(x+1) - f(x) \quad (h=1)$$

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

$$magn(\nabla f) = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} = \sqrt{M_x{}^2 + M_y{}^2}$$

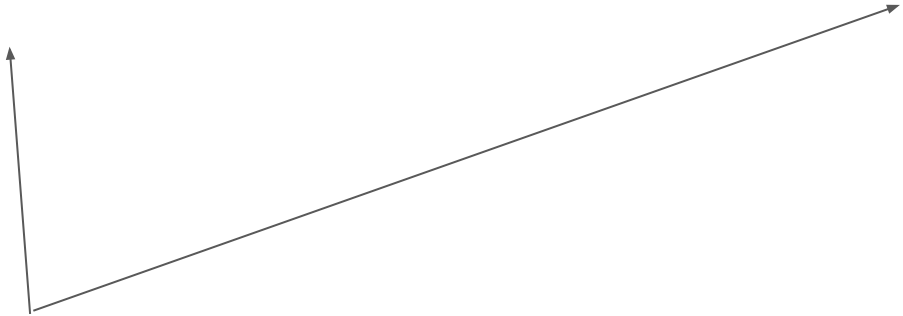
Модуль градиента (магнитуда)

$$dir(\nabla f) = \tan^{-1}(M_y / M_x)$$

Направление градиента (угол)

Оператор Собеля

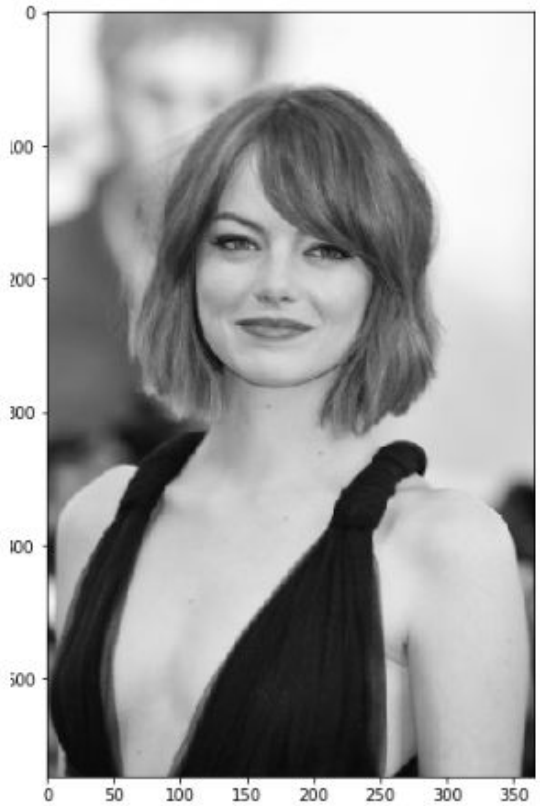
$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A}$$

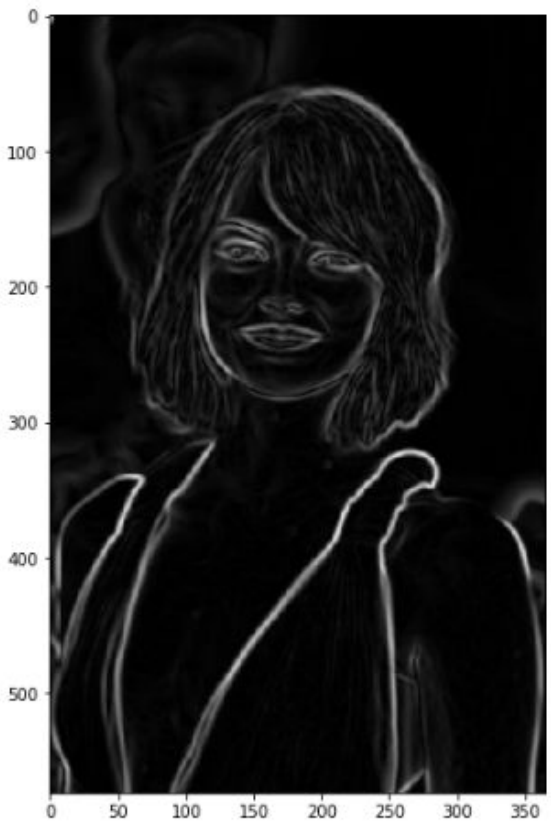


Приближенные производные по x и y

Оператор Собеля

- Подверженность шуму
- Неточные границы
- “Потеря” слабых границ





Алгоритм поиска границ Кэнни (Canny Edge)

Требования

- Низкая вероятность пропустить настоящие границы и пометить граничными точки, не являющиеся ими
- Точка, которая помечается как граничная, должна лежать как можно ближе к центру настоящей границы
- Только один отклик на настоящую границу

A Computational Approach to Edge Detection, John Canny, 1986

Алгоритм поиска границ Кэнни (Canny Edge)

1) Уменьшение шума
2) Вычисление градиента
3) Non-maximum suppression
4) Уточнение границ по верхнему и нижнему порогу
5) Hysteresis threshold

A Computational Approach to Edge Detection, John Canny, 1986

Canny Edge - уменьшение шума

- Наложение фильтра Гаусса
- Размер ядра и значение среднеквадратичного отклонения - параметры алгоритма

A Computational Approach to Edge Detection, John Canny, 1986

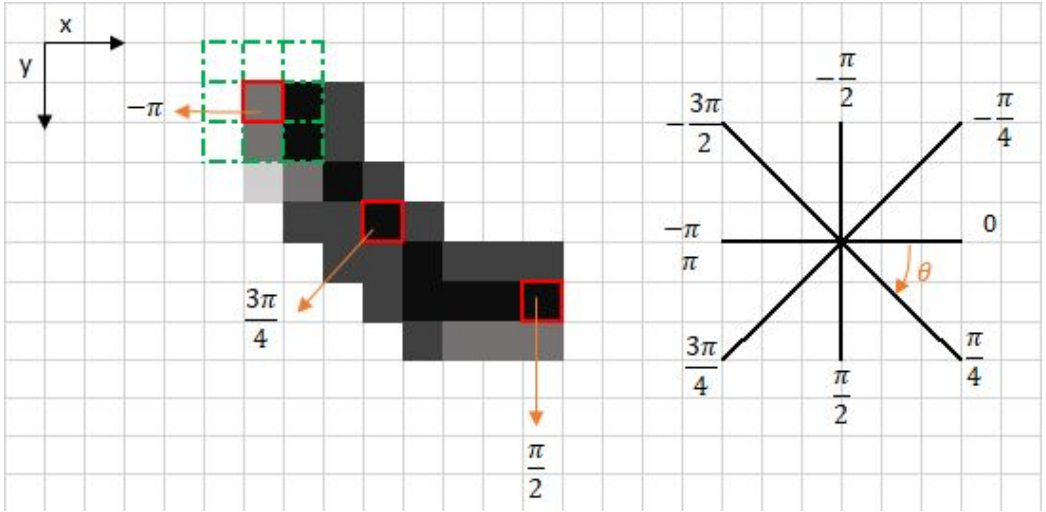
Canny Edge - вычисление градиента

- Вычислить производные по x и y с помощью оператора Собеля
- Рассчитать модуль и направление

A Computational Approach to Edge Detection, John Canny, 1986

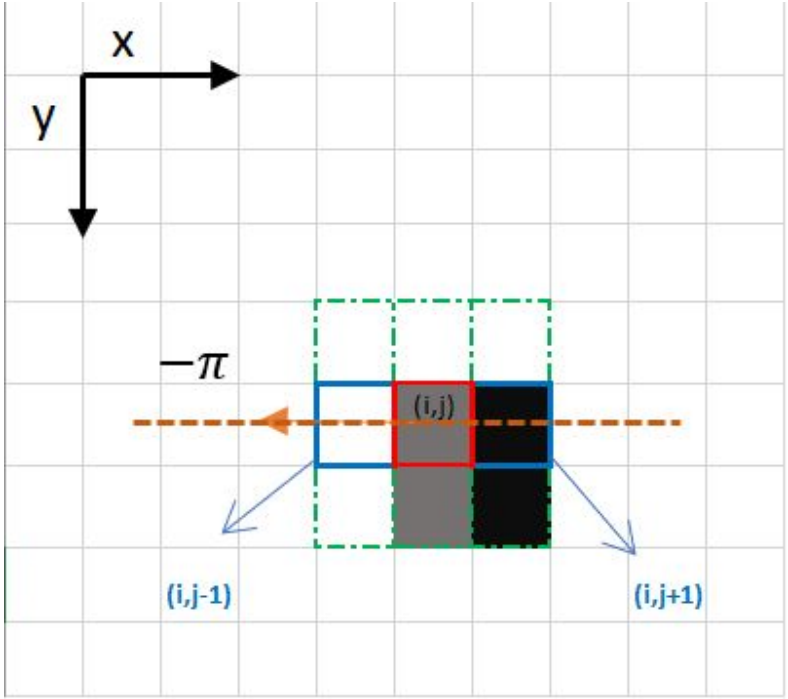
Canny Edge - non-maximum suppression

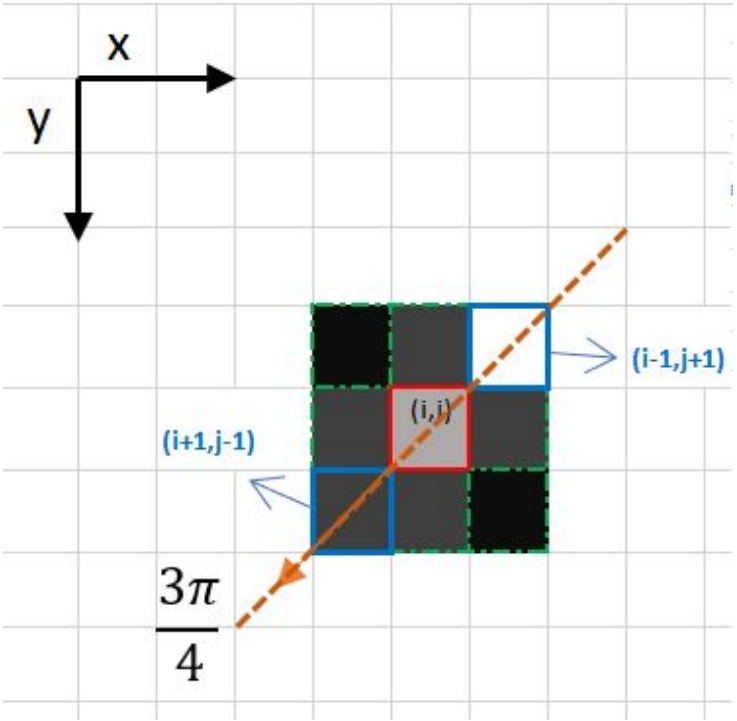
- Для каждой тройки пикселей найти максимальное значение на основе направления градиента
- Если значение модуля градиента не максимальное, то исключить пиксель



A Computational Approach to Edge Detection, John Canny, 1986

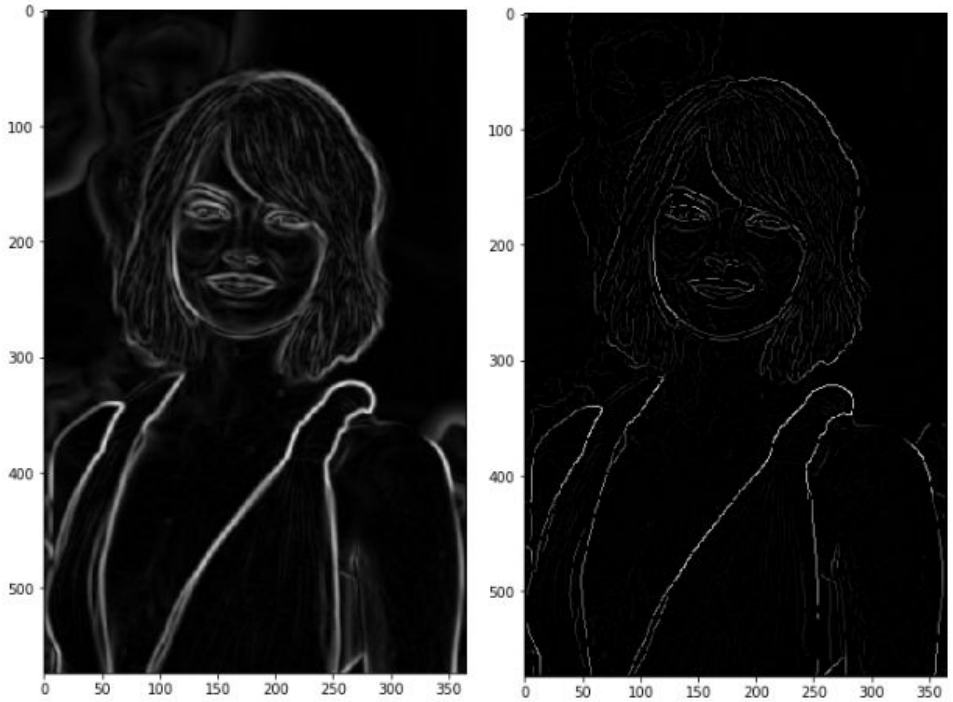
Canny Edge - non-maximum suppression





A Computational Approach to Edge Detection, John Canny, 1986

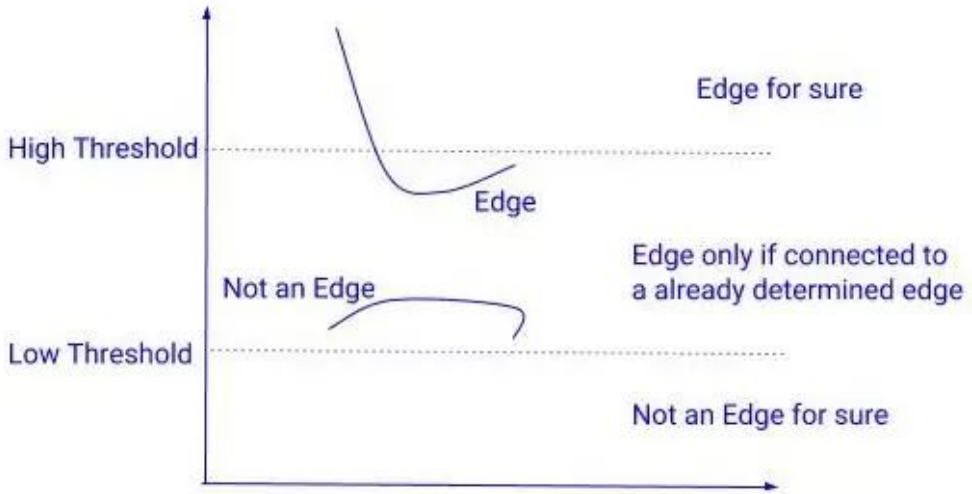
Canny Edge - non-maximum suppression



A Computational Approach to Edge Detection, John Canny, 1986

Canny Edge - уточнение границ по верхнему и нижнему порогу

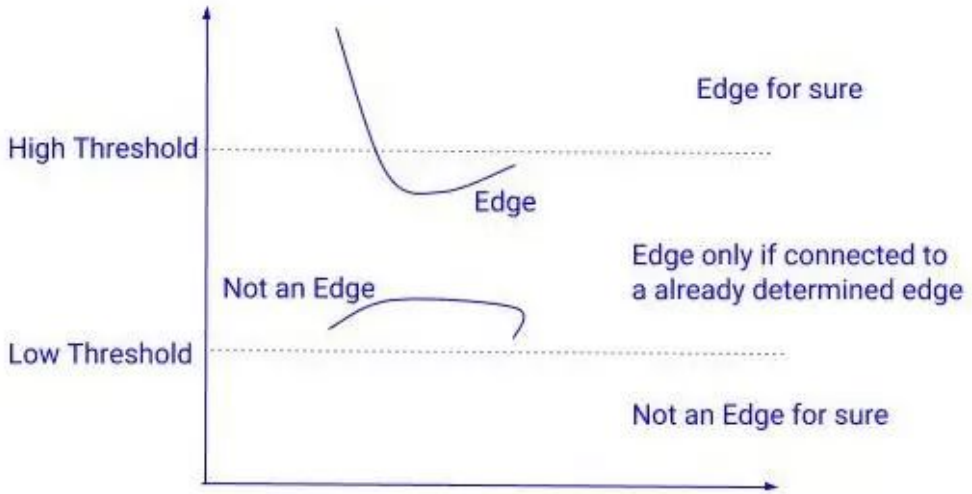
- Ввести два параметра алгоритма **low_threshold** и **high_threshold**
- Все пиксели со значениями модуля градиента больше **high_threshold** - сильные и останутся на итоговом изображении
- Все пиксели со значениями модуля градиента ниже **low_threshold** - исключаются из итогового изображения
- Все пиксели со значениями между **low_threshold** и **high_threshold** - слабые и подвергнутся дополнительной проверке



A Computational Approach to Edge Detection, John Canny, 1986

Canny Edge - Hysteresis threshold

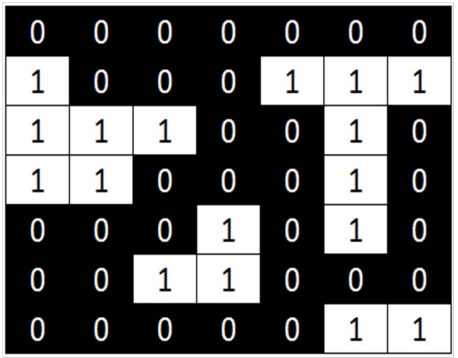
- Слабые пиксели могут быть окружены слабыми, но в каком-то месте соприкасаться с сильными - это должна быть одна граница
- Нужно найти все связные компоненты из сильных и слабых пикселей на изображении
- Connected Components Labeling



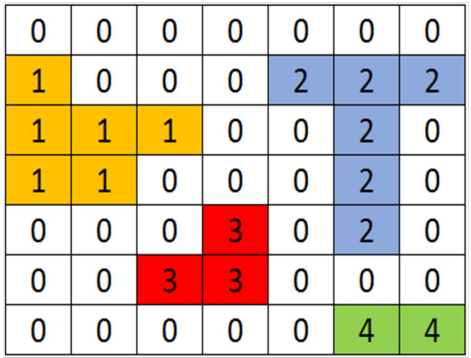
A Computational Approach to Edge Detection, John Canny, 1986

Connected Components Labeling - разметка связных областей

- Плеяда методов основанная на теории графов
- Суть методов для Image Processing - в поиске связных областей на изображении

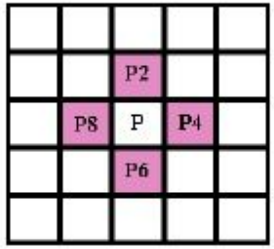


(a)

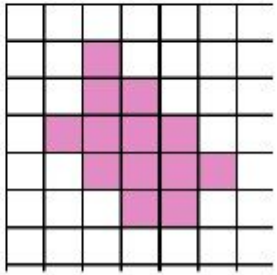


(b)

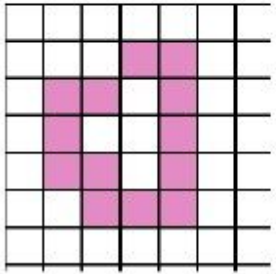
Виды связностей



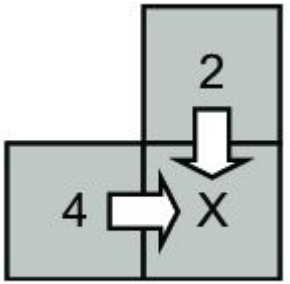
4-Connectivity

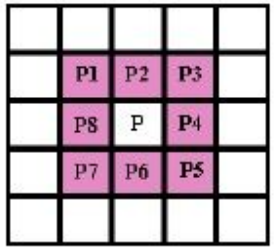


4-Connected

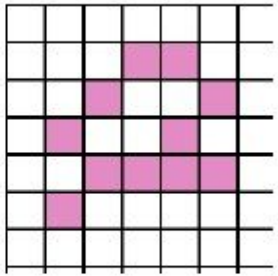


Not 4-Connected

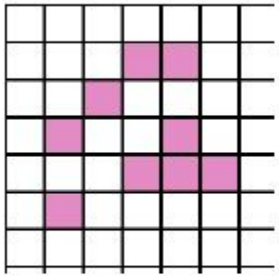




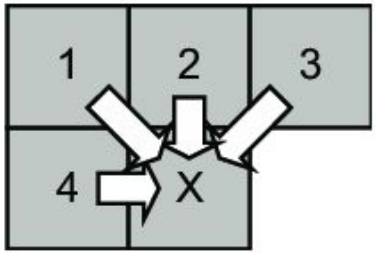
8-Connectivity



8-Connected



Not 8-Connected



Система непересекающихся множеств (disjoint-set, union-find)

Структура данных для администрирования элементов, разбитых на непересекающиеся подмножества.
Использует три абстрактных операции: {**Union**, **Find**, **MakeSet**}

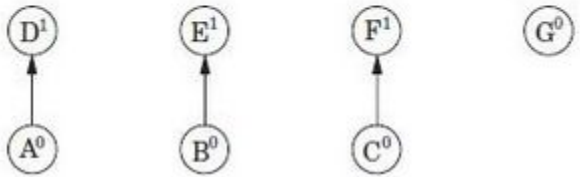
MakeSet(x) - создаёт для элемента x новое подмножество. Назначает этот же элемент представителем созданного подмножества. O(1)
Find(x) - проходит путь от x до корня дерева и возвращает его (корень в данном случае является представителем). O(n)
Union(r,s) - Ищет корни с помощью **Find** для r и s. Если корни различны объединяет оба подмножества, принадлежащие представителям r и s, присоединяя корень r к корню s. O(n)

Нетривиальная реализация - с помощью деревьев и именно она (и некоторые ухищрения) позволяет существенно ускорить работу со структурой

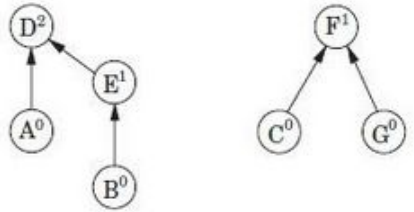
Система непересекающихся множеств (disjoint-set, union-find) - пример Union, MakeSet, Find

After makeset(A),makeset(B),...,makeset(G):

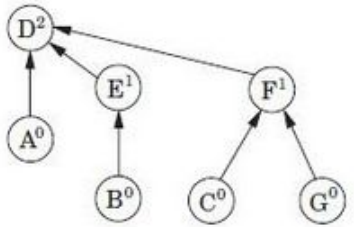
After union(A,D),union(B,E),union(C,F):



After union(C,G),union(E,A):



After union(B,G):

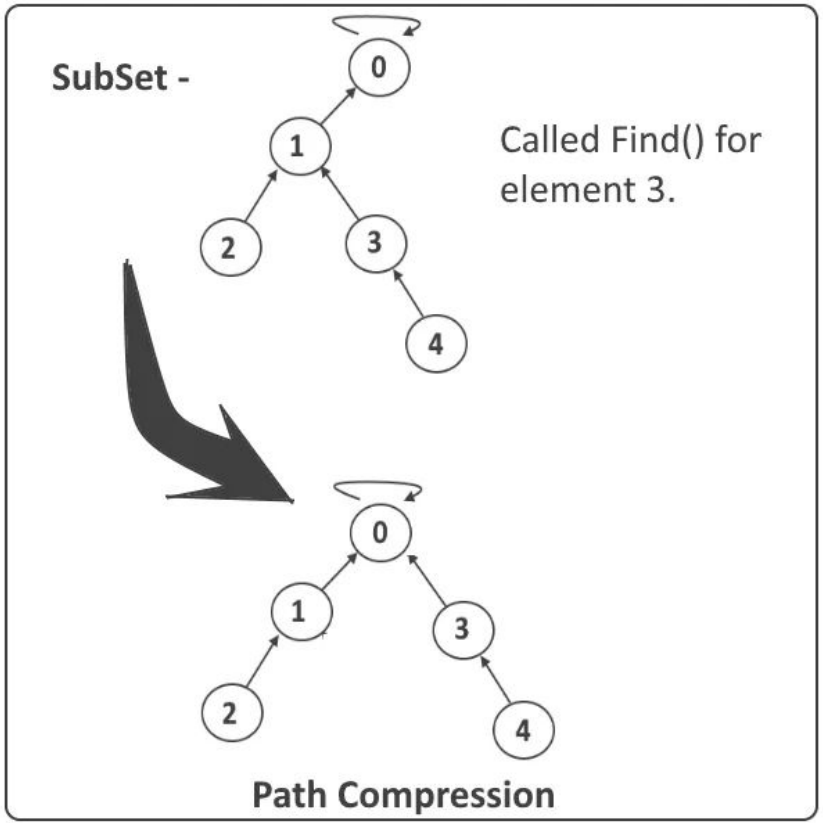


Хорошая реализация предполагает:

1) Path compression
2) Union by Rank

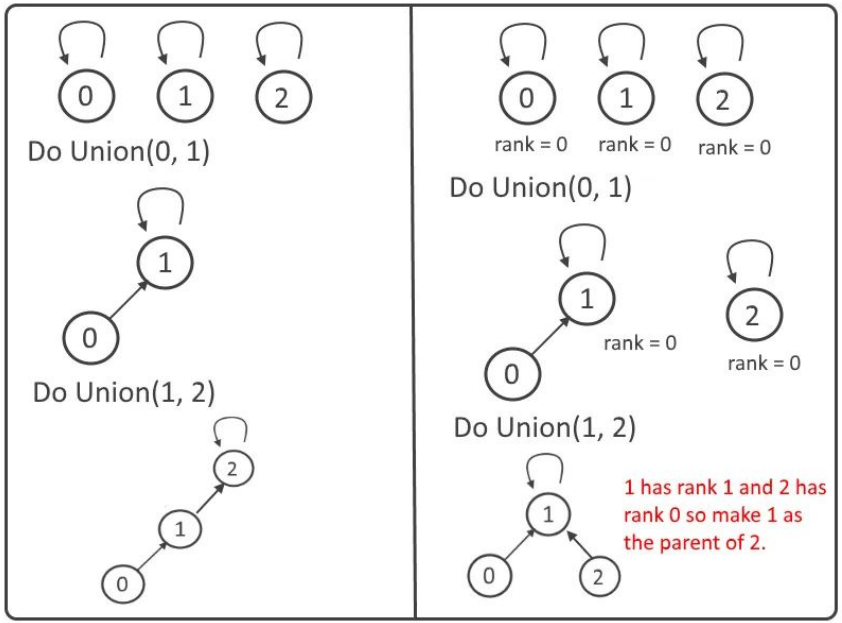
Система непересекающихся множеств (disjoint-set, union-find) - Path Compression

- Path Compression
- Исключение цепных правил при вызове **Find**
- При каждом вызове **Find(x),** принудительно делать родителем **x** корень дерева



Система непересекающихся множеств (disjoint-set, union-find) - Union by Rank

- Union by Rank
- Присоединить более короткое дерево к корню более длинного
- Установить **rank** множества равным нулю при вызове **MakeSet**(x)
- При объединении двух множеств одинакового rank, результирующий **rank**=rank+1
- При объединении двух множеств разных rank0, rank1, результирующий **rank**=max(rank0,rank1)



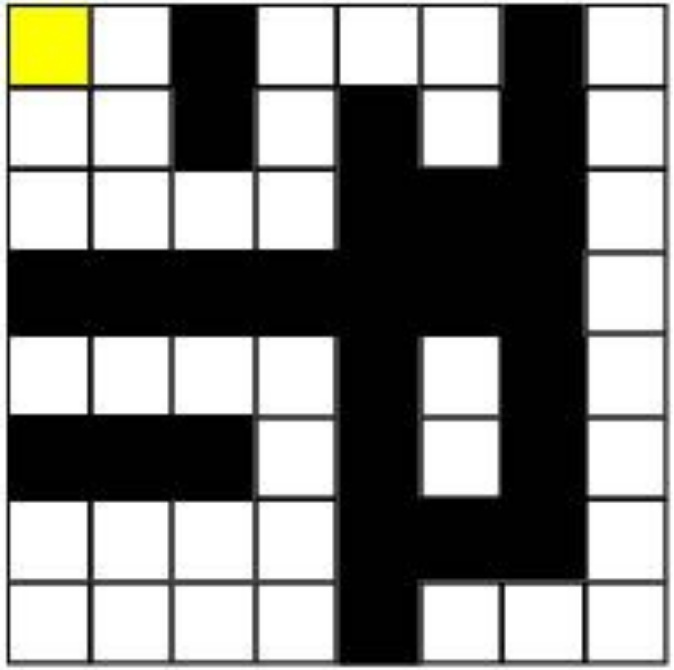
Система непересекающихся множеств (disjoint-set, union-find) - сложность

Construction	O(n)
Union	α(n)
Find	α(n)
Get component size	α(n)
Check if connected	α(n)
Count components	O(1)

Alpha - функция, обратная функции Аккермана. Для всех применяемых на практике значений n принимает значение, меньшее 5.

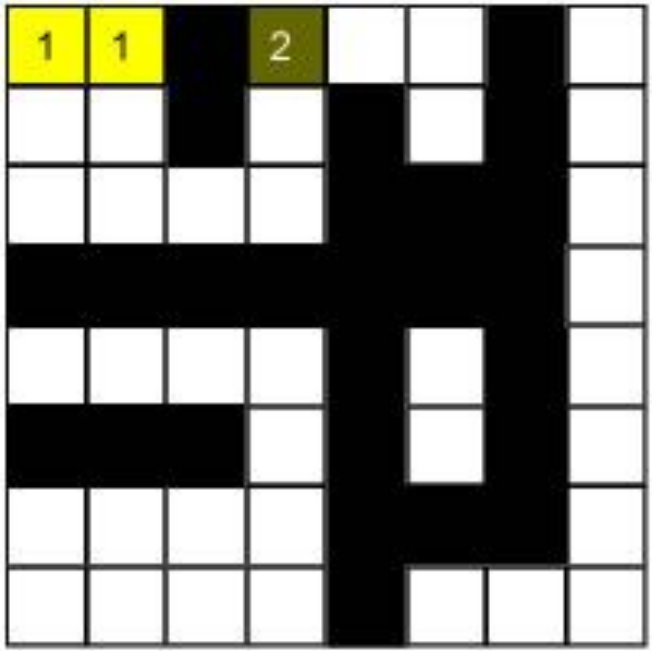
Алгоритм - Two Pass Connected Components Labeling - First Pass

1) Проход по каждому пикселю слева направо, сверху вниз
2) Если значение пикселя не задний фон:
   a) Получить значение меток соседних пикселей согласно окну для текущего пикселя
      i) Если соседей нет - присвоить новую метку этому пикселю (**MakeSet**)
      ii) В противном случае найти соседа с наименьшей меткой и присвоить эту метку текущему пикселю
      iii) Сохранить значение эквивалентности для соседних пикселей (**Union**)



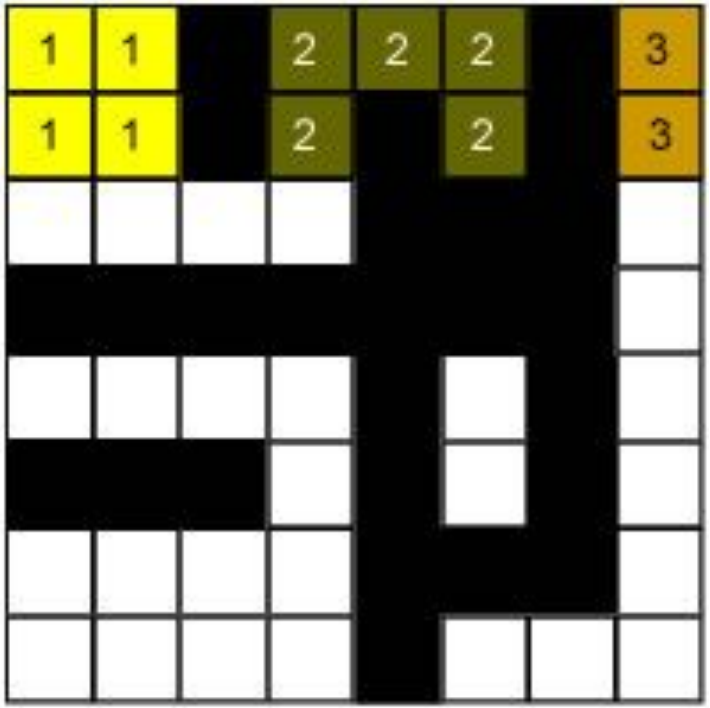
Алгоритм - Two Pass Connected Components Labeling - First Pass

1) Проход по каждому пикселю слева направо, сверху вниз
2) Если значение пикселя не задний фон:
   a) Получить значение меток соседних пикселей согласно окну для текущего пикселя
      i) Если соседей нет - присвоить новую метку этому пикселю (**MakeSet**)
      ii) В противном случае найти соседа с наименьшей меткой и присвоить эту метку текущему пикселю
      iii) Сохранить значение эквивалентности для соседних пикселей (**Union**)



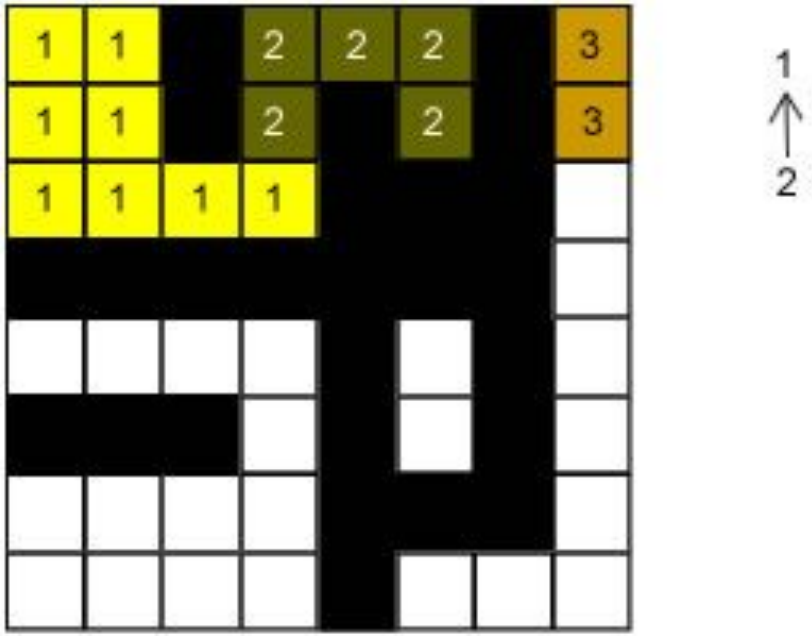
Алгоритм - Two Pass Connected Components Labeling - First Pass

1) Проход по каждому пикселю слева направо, сверху вниз
2) Если значение пикселя не задний фон:
   a) Получить значение меток соседних пикселей согласно окну для текущего пикселя
      i) Если соседей нет - присвоить новую метку этому пикселю (**MakeSet**)
      ii) В противном случае найти соседа с наименьшей меткой и присвоить эту метку текущему пикселю
      iii) Сохранить значение эквивалентности для соседних пикселей (**Union**)



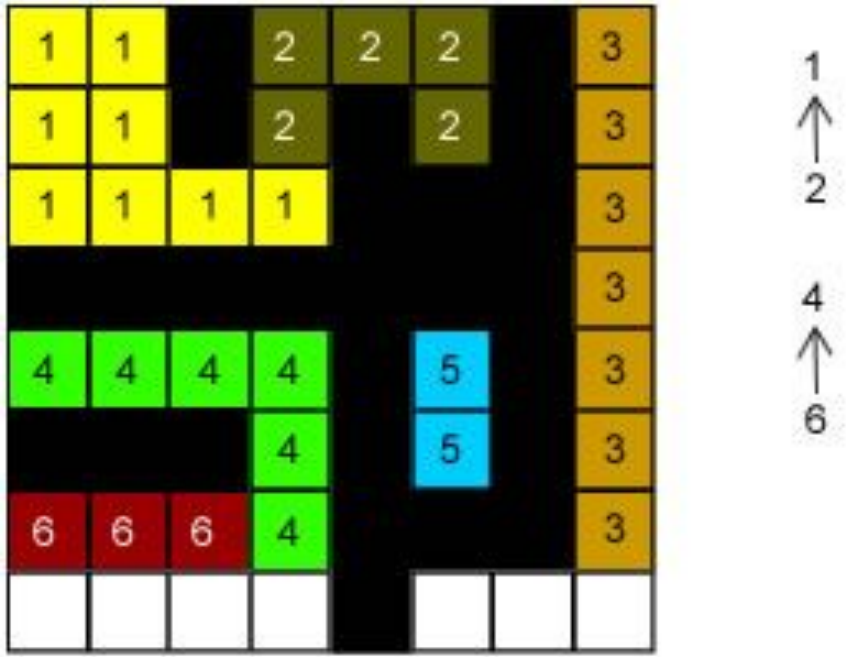
Алгоритм - Two Pass Connected Components Labeling - First Pass

1) Проход по каждому пикселю слева направо, сверху вниз
2) Если значение пикселя не задний фон:
   a) Получить значение меток соседних пикселей согласно окну для текущего пикселя
      i) Если соседей нет - присвоить новую метку этому пикселю (**MakeSet**)
      ii) В противном случае найти соседа с наименьшей меткой и присвоить эту метку текущему пикселю
      iii) Сохранить значение эквивалентности для соседних пикселей (**Union**)



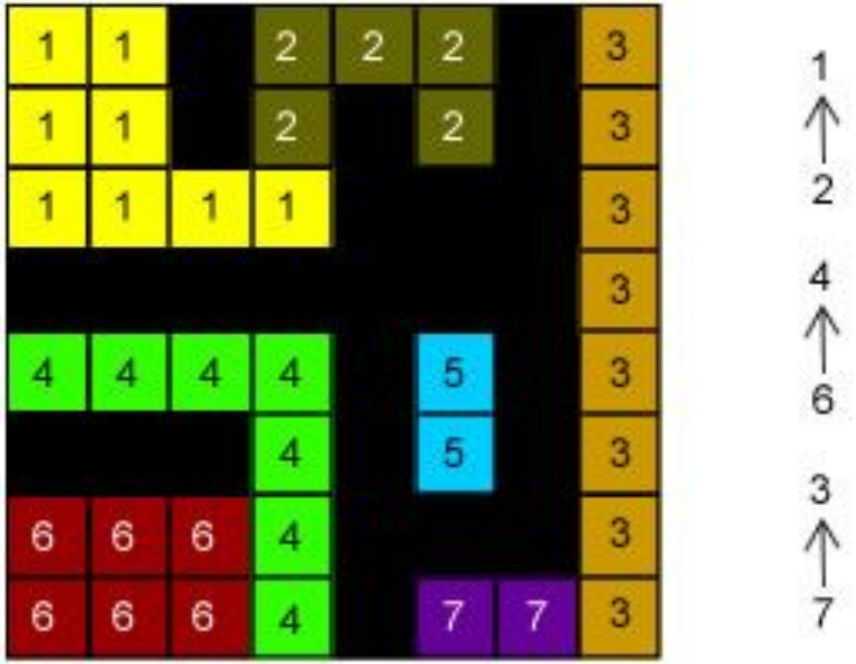
Алгоритм - Two Pass Connected Components Labeling - First Pass

1) Проход по каждому пикселю слева направо, сверху вниз
2) Если значение пикселя не задний фон:
   a) Получить значение меток соседних пикселей согласно окну для текущего пикселя
      i) Если соседей нет - присвоить новую метку этому пикселю (**MakeSet**)
      ii) В противном случае найти соседа с наименьшей меткой и присвоить эту метку текущему пикселю
      iii) Сохранить значение эквивалентности для соседних пикселей (**Union**)



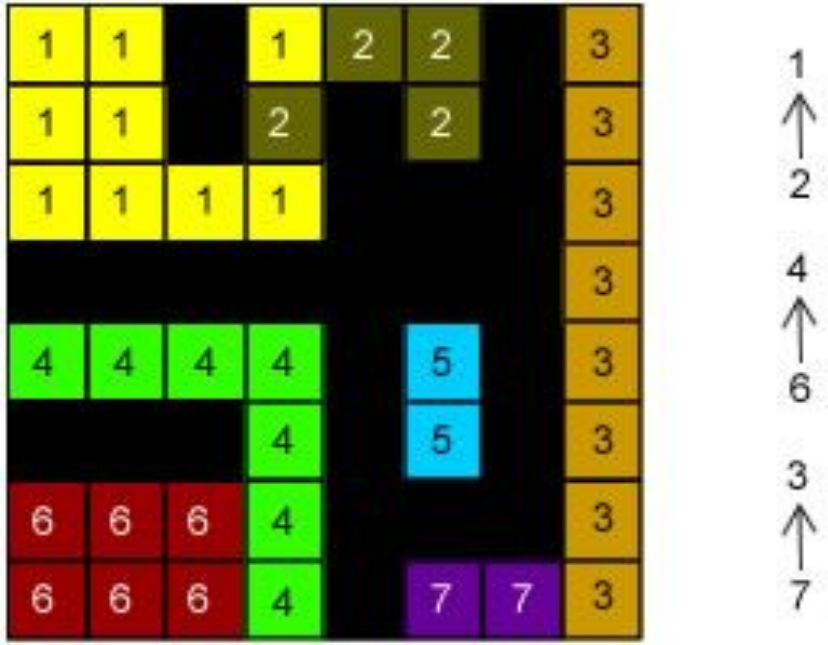
Алгоритм - Two Pass Connected Components Labeling - First Pass

1) Проход по каждому пикселю слева направо, сверху вниз
2) Если значение пикселя не задний фон:
   a) Получить значение меток соседних пикселей согласно окну для текущего пикселя
      i) Если соседей нет - присвоить новую метку этому пикселю (**MakeSet**)
      ii) В противном случае найти соседа с наименьшей меткой и присвоить эту метку текущему пикселю
      iii) Сохранить значение эквивалентности для соседних пикселей (**Union**)



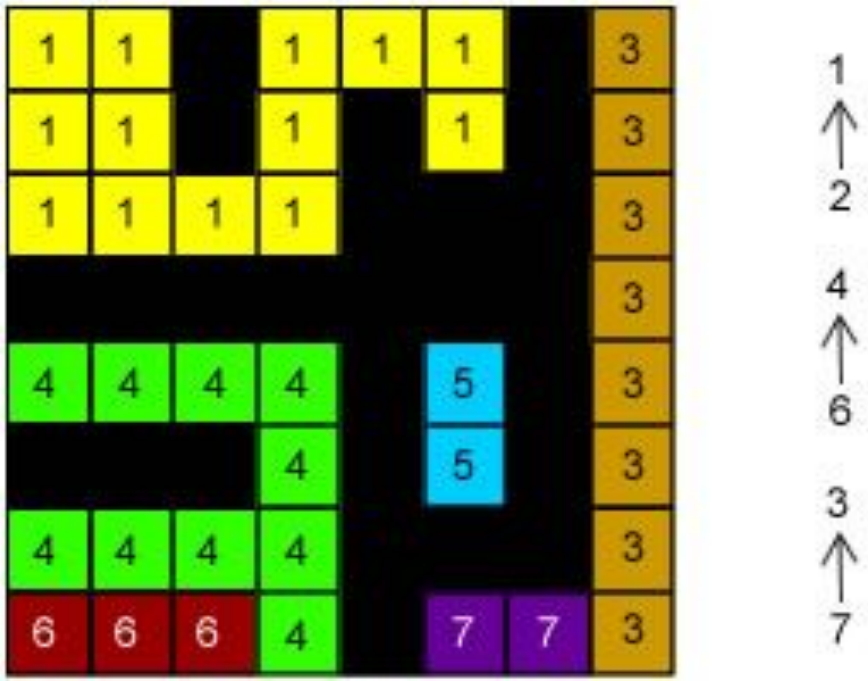
Алгоритм - Two Pass Connected Components Labeling - Second Pass

1) Проход по каждому пикселю слева направо, сверху вниз
2) Если значение пикселя не задний фон:
   a) Переприсвоить значение метки на значение наименьшей метки среди группы эквивалентности (**Find**)



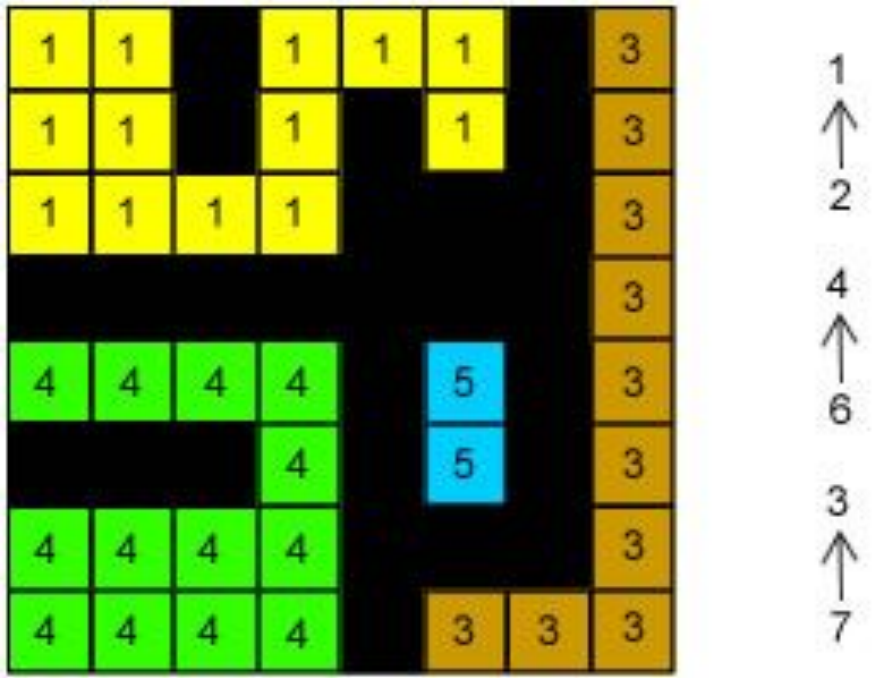
Алгоритм - Two Pass Connected Components Labeling - Second Pass

1) Проход по каждому пикселю слева направо, сверху вниз
2) Если значение пикселя не задний фон:
   a) Переприсвоить значение метки на значение наименьшей метки среди группы эквивалентности (**Find**)



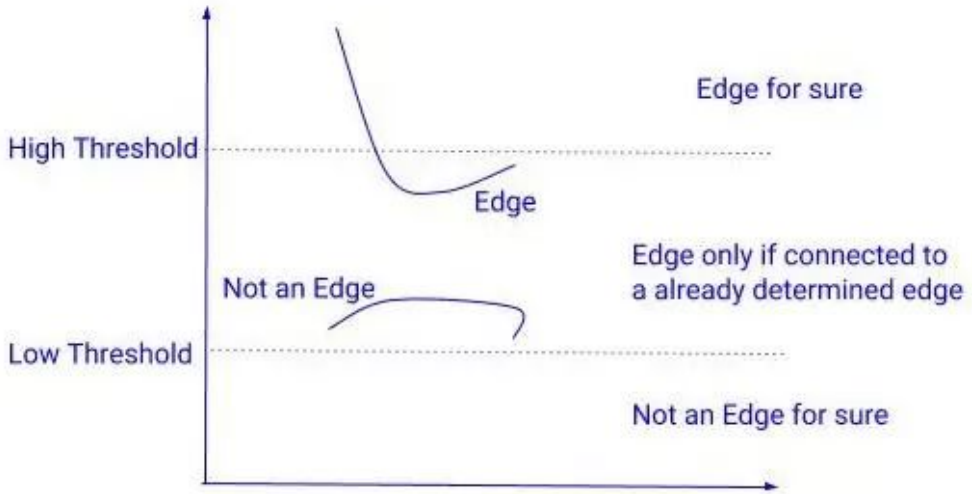
Алгоритм - Two Pass Connected Components Labeling - Second Pass

1) Проход по каждому пикселю слева направо, сверху вниз
2) Если значение пикселя не задний фон:
   a) Переприсвоить значение метки на значение наименьшей метки среди группы эквивалентности (**Find**)



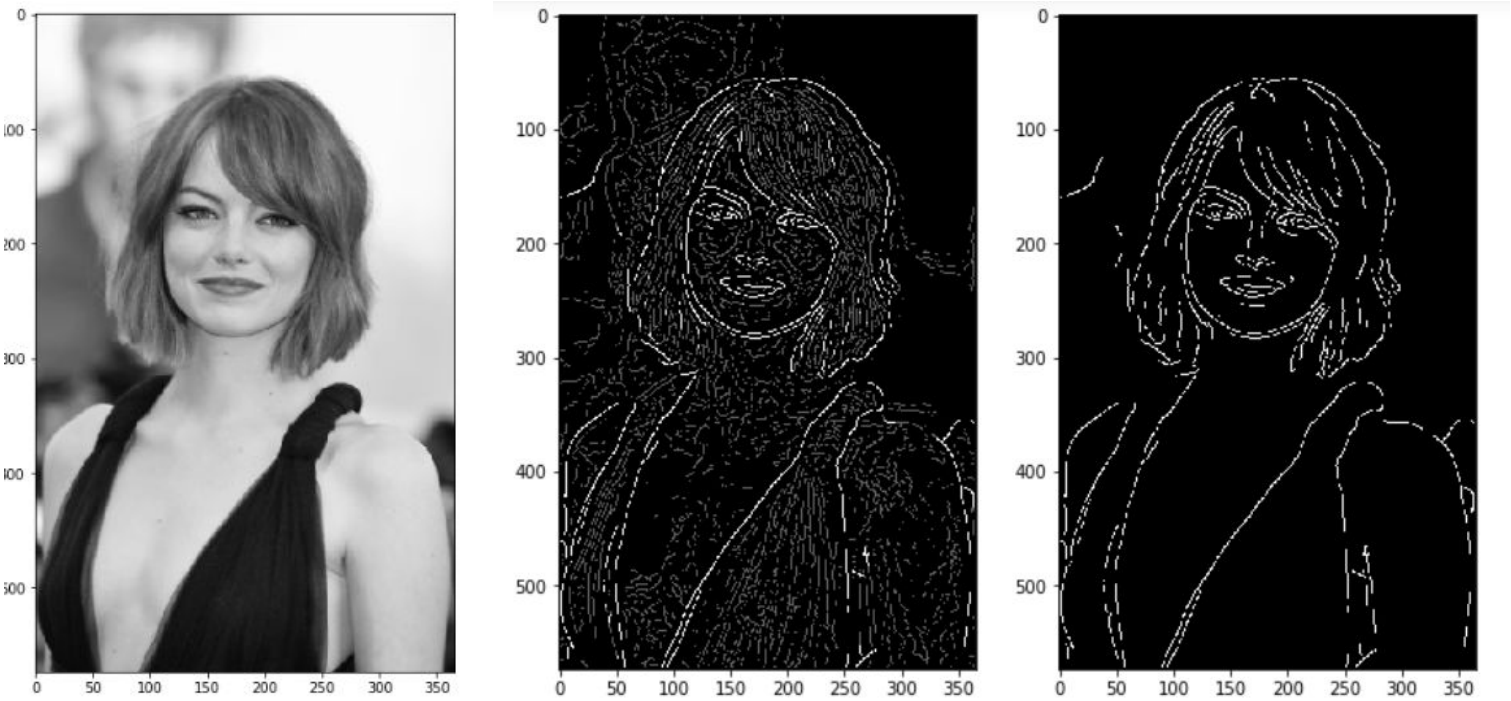
Canny Edge - Hysteresis threshold

- Слабые пиксели могут быть окружены слабыми, но в каком-то месте соприкасаться с сильными - это должна быть одна граница
- Нужно найти все связные компоненты из сильных и слабых пикселей на изображении
- **Connected Components Labeling - ищем связность для всех слабых и сильных пикселей вместе**
- **Оставляем только те компоненты, где есть хотя бы один сильный пиксель**



A Computational Approach to Edge Detection, John Canny, 1986

Canny Edge - Hysteresis threshold



A Computational Approach to Edge Detection, John Canny, 1986