**Матричные операции**

1. Рисунок представляет собой прямоугольную матрицу, каждый пиксель которой может быть черным или белым. Найти площадь максимального субпрямоугольника, у которого все стороны черные. Если решать в «лоб», то трудоемкость равна O(N4).
2. Дана отсортированная матрица. Разработать программу поиска в ней заданного числа эффективностью O(Log2(N)). Отсортированная матрица представляет собой такую матрицу, в которой элементы, находящиеся выше и левее заданного элемента меньше заданного. Пример отсортированной матрицы:

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | 20 | 30 | 40 |
| 50 | 60 | 70 | 80 |
| 90 | 100 | 110 | 120 |
| 130 | 140 | 150 | 160 |

1. **Произведение матриц**

A= B= >> A\*B=

1. **Деление матриц**

A= B= >> A/B=

1. **Нормализация матриц**

A= >>

1. **Обращение (инверсия) матриц**

A= >> A-1=

1. **Возведение в степень**

A= >> A3=

1. **Извлечение корня**

A= >> =

1. **Вычисление определителя**

A= Det(A)=-1

1. **Вращение матриц**

Обычно применяется для 3D-матриц, которые представляют собой матрицы координат точек в трехмерном пространстве. При этом первая координата соответствует оси X, вторая - оси Y, третья – оси Z. Соответственно возможно вращение матрицы вокруг любой из трех осей.

Например, дана матрица:

A=

При ее вращении вокруг оси Y на угол получим:

A=

1. **Сравнение матриц**

A= B=

A=B поскольку, если умножить первую строку матрицы А на 2, а третью строку матрицы А разделить на 3, то получим

А==B

1. Напишите код поиска субматрицы с максимально возможной суммой в матрице N\*N, содержащей положительные и отрицательные числа. Если решать в «лоб», то трудоемкость равна O(N6).
2. Дана матрица N\*N, элементами которой являются различные целые числа. Составить программу линейной сложности, находящую в указанной матрице любой локальный минимум. Локальным минимумом матрицы называется элемент, который меньше всех своих четырёх соседей (или трёх, если этот элемент лежит на границе; или двух, если это угловой элемент). Обратите внимание, что от нас требуется линейное по n время, хотя в матрице квадратичное по n число элементов. Поэтому предполагается, что матрица уже считана в память.
3. **Приведение матриц к треугольному виду (1)**

A= >>

1. **Приведение матриц к треугольному виду (2)**

A= >>

1. **Приведение матриц к треугольному виду (3)**

A= >>

1. **Диагонализация матриц (1)**

A= >>

1. **Диагонализация матриц (2)**

A= >>