In [1]:
```
import numpy as np
```

Answer 1

In [11]:
```
def transposed_convolution_function(matrix, kernel, stride, crop):
    matrix_h, matrix_w = matrix.shape
    kernel_h, kernel_w = kernel.shape

    #Calculating the size of the output matrix
    output_h = (matrix_h - 1) * stride + kernel_h - 2 * crop
    output_w = (matrix_w - 1) * stride + kernel_w - 2 * crop

    #Initializing the output matrix
    output_matrix = np.zeros((output_h, output_w))

    # Performing the transposed convolution
    for i in range(matrix_h): ## i, j are the indices of the input matrix
        for j in range(matrix_w):
            for ki in range(kernel_h):   ## ki, kj are the indices of the kernel
                for kj in range(kernel_w):
                    output_matrix[i * stride + ki, j * stride + kj] += (
                        matrix[i, j] * kernel[ki, kj]
                    )

    #  if crop >0 then copping the output matrix
    if crop > 0:
        output_matrix = output_matrix[crop:-crop, crop:-crop]

    return output_matrix
```

In [12]:
```
# Example
input_matrix = np.array([[1, 2], [3, 4]])
kernel = np.array([[1, 0], [0, 1]])
stride = 2
crop = 0
```

```
output_matrix=transposed_convolution_function(input_matrix, kernel, stride, crop)
print(f"Output matrix:\n{output_matrix}")
```

```
Output matrix:
[[1. 0. 2. 0.]
 [0. 1. 0. 2.]
 [3. 0. 4. 0.]
 [0. 3. 0. 4.]]
```

Answer 2

In [13]:
```
def intersection_over_union(matrix1, matrix2): ##both matrix1 and matrix2 are binary matrices


    # Calculate intersection and union areas

    intersection = np.logical_and(matrix1, matrix2).sum()  ##logical_and is used to find the intersection of two matrices and
                                                           ## sum is used to find the total number of 1's in the intersection matr

    union = matrix1.sum() + matrix2.sum() - intersection  ##sum is used to find the total number of 1's in the union matrix an
                                                          ## subtracting the intersection from the sum of both matrices to get the

    # if union =0 then we need to handle division by zero
    if union == 0:
        return 0

    # Calculate IoU
    iou = intersection / union

    return iou
```

In [14]:
```
# Example
matrix1 = np.array([[1, 0, 1], [0, 1, 0]])
matrix2 = np.array([[1, 1, 0], [0, 1, 1]])

iou_value = intersection_over_union(matrix1, matrix2)
print(f"IoU: {iou_value}")
```

```
IoU: 0.4
```