Transpose of a Matrix

```
python
Copy code
def transpose(matrix):
    return list(map(list, zip(*matrix)))

# Example usage:
matrix1 = [[1,2,3],[4,5,6],[7,8,9]]
matrix2 = [[1,2,3],[4,5,6]]
print(transpose(matrix1))  # Output: [[1,4,7],[2,5,8],[3,6,9]]
print(transpose(matrix2))  # Output: [[1,4],[2,5],[3,6]]
```

Number of Valid Pairs

```
python
Copy code
def countPairs(nums1, nums2, diff):
    count = 0
    for i in range(len(nums1)):
        for j in range(i + 1, len(nums1)):
            if nums1[i] - nums1[j] <= nums2[i] - nums2[j] + diff:
                count += 1
    return count

# Example usage:
nums1 = [3,2,5]
nums2 = [2,2,1]
diff = 1
print(countPairs(nums1, nums2, diff))  # Output: 3
```

Nth Digit of Infinite Integer Sequence

```
python
Copy code
def findNthDigit(n):
    length = 1
    count = 9
    start = 1

    while n > length * count:
        n -= length * count
        length += 1
        count *= 10
        start *= 10

    start += (n - 1) // length
    s = str(start)
    return int(s[(n - 1) % length])

# Example usage:
n1 = 3
n2 = 11
print(findNthDigit(n1))  # Output: 3
print(findNthDigit(n2))  # Output: 0
```

Longest Nice Substring

Search Word Prefix in Sentence

```
python
Copy code
def isPrefixOfWord(sentence, searchWord):
    words = sentence.split()
    for i, word in enumerate(words):
        if word.startswith(searchWord):
            return i + 1
    return -1

# Example usage:
sentence = "i love eating burger"
searchWord = "burg"
print(isPrefixOfWord(sentence, searchWord))  # Output: 4
```

Contains Nearby Almost Duplicate

```
python
Copy code
def containsNearbyAlmostDuplicate(nums, indexDiff, valueDiff):
    sorted_set = sorted((num, i) for i, num in enumerate(nums))
    for i in range(len(nums)):
        for j in range(i + 1, len(nums)):
            if sorted_set[j][0] - sorted_set[i][0] > valueDiff:
                break
            if abs(sorted_set[j][1] - sorted_set[i][1]) <= indexDiff:
                return True
    return False

# Example usage:
nums = [1,2,3,1]
indexDiff = 3
valueDiff = 0
print(containsNearbyAlmostDuplicate(nums, indexDiff, valueDiff))  # Output:
True
```

Minimum Length of Array after Deleting Pairs

```
python
Copy code
def minLength(nums):
    count = 0
    while nums:
        i = 0
        while i < len(nums) - 1 and nums[i] >= nums[i + 1]:
            i += 1
        if i < len(nums) - 1:
            nums.pop(i)
            nums.pop(i)
        else:
            break
        count += 2
    return len(nums)
```

```
# Example usage:
nums = [1,2,3,4]
print(minLength(nums))  # Output: 0
```

Convert Sorted Array to BST

```
python
Copy code
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def sortedArrayToBST(nums):
    if not nums:
        return None
    mid = len(nums) // 2
    root = TreeNode(nums[mid])
    root.left = sortedArrayToBST(nums[:mid])
    root.right = sortedArrayToBST(nums[mid+1:])
    return root

# Example usage:
nums = [-10,-3,0,5,9]
root = sortedArrayToBST(nums)
# The output will be the root of the constructed BST
```

Strings that are Substrings of Another Word

```
python
Copy code
def stringMatching(words):
    result = []
    for i in range(len(words)):
        for j in range(len(words)):
            if i != j and words[i] in words[j]:
                result.append(words[i])
                break
    return result

# Example usage:
words = ["mass","as","hero","superhero"]
print(stringMatching(words))  # Output: ["as","hero"]
```

Wiggle Sort II

```
python
Copy code
def wiggleSort(nums):
    nums.sort()
    half = len(nums[::2])
    nums[::2], nums[1::2] = nums[:half][::-1], nums[half:][::-1]

# Example usage:
nums1 = [1,5,1,1,6,4]
nums2 = [1,3,2,2,3,1]
wiggleSort(nums1)
wiggleSort(nums2)
```

```
print(nums1)  # Output: [1,6,1,5,1,4] (or similar valid answer)
print(nums2)  # Output: [2,3,1,3,1,2] (or similar valid answer)
```