Name: Ahaan Desai
Roll no.: 231070016
Batch: Batch A (1-20)

DAA Lab Experiment 4

Aim

1. To find inversion count of course choice of students.
2. To multiply two integers using brute force and divide-and-conquer methods

Program

1. Inversion Count

```python
from collections import Counter
import pandas as pd

def count_inversions(arr):
    """Counts the number of inversions in an array."""
    if len(arr) <= 1:
        return arr, 0

    mid = len(arr) // 2
    left, left_inversions = count_inversions(arr[:mid])
    right, right_inversions = count_inversions(arr[mid:])

    merged, split_inversions = merge_and_count_split_inversions(left, right)
    return merged, split_inversions + left_inversions + right_inversions

def merge_and_count_split_inversions(left, right):
    """Merges two arrays and counts the number of split inversions."""
    result = []
    i = j = split_inversions = 0
    n = len(left)
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
```

```
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
            split_inversions += len(left) - i

    result.extend(left[i:])
    result.extend(right[j:])

    return result, split_inversions


def inversions_course_codes(choices_students):
    """Counts the number of inversions in a list of choices, and classifies
them according to the count of inversions."""
    inversions = []
    for choices in choices_students:
        _, t = count_inversions(choices)
        inversions.append(t)
    count = dict(sorted(Counter(inversions).items()))
    # Creates a hashmap/dictionary with key as the inversion count and value as
the number of students that have that count.
    # Sorts the dictionary based on the inversion count.
    return count


df = pd.read_csv('course_choice.csv')
student_ids = df['Student'].tolist()
choices_students = df.drop(columns=['Student']).values.tolist()

for k, v in inversions_course_codes(choices_students).items():
    print(f"{v:2d} students have {k:2d} inversion count.")
```

2. Integer Multiplication

Python

```python
def normal_multiplication(x, y):
    """
    Performs multiplication of two integers by the normal method.
    """
    if type(x) == float or type(y) == float:
        print("Not an integer")
        return -1
    sign = -1 if (x < 0) ^ (y < 0) else 1
    x, y = abs(x), abs(y)

    x = str(x)[::-1]
    y = str(y)[::-1]
    res = 0
    for power1, digit1 in enumerate(x):
        for power2, digit2 in enumerate(y):
            res += int(digit1) * int(digit2) * 10 ** (power1 + power2)

    return sign*res


def karatsuba_multiplication(x, y):
    """
    Performs multiplication of two integers using the divide and conquer
karatsuba algorithm.
    """
    if type(x) == float or type(y) == float:
        print("Not an integer")
        return -1
    sign = -1 if (x < 0) ^ (y < 0) else 1
    x, y = abs(x), abs(y)

    if x < 10 or y < 10:
        return x * y

    m = min(
        len(str(x)),
        len(str(y))
            )
    m2 = m//2
```

```
    high1, low1 = divmod(x, 10**m2)         # Equivalent to splitting in the
middle
    high2, low2 = divmod(y, 10**m2)

    z0 = karatsuba_multiplication(low1, low2)
    z2 = karatsuba_multiplication(high1, high2)
    z1 = karatsuba_multiplication(low1 + high1 , low2 + high2) - z2 - z0
    # The function outputs low1*low2 + high1*low2 + high1*high2 + high2*low1
    # from which we must subtract low1low2(z0) and high1high2(z2)

    return sign * (z2*(10 ** (2*m2)) + z1*(10 ** m2) + z0)

def tests():
    numbers = [
        (12342342352354534553346356, 3045603456834760346356356 5),
        (53849450494776394611921152, -506335097398631071777706 22),
        (25069.51, 77777.321),
        (0, 1155328940683083679213231),
        (97672243867560276084372410, 0),
        (-25518178525865466802925 33, 7344195291378678037219346 8.454),
        (-74494213327602150702262607, -68674501952269147188782896),
        (222.01, 0),
        (-444.05, 1000.001),
        (-36363636, 1515.15)
    ]

    for x,y in numbers:
        print(f"Testcase: {x}, {y}")
        print(normal_multiplication(x, y))
        print(karatsuba_multiplication(x, y))


tests()
```

Output

1. Inversion count

```
1 students have 0 inversion count.
2 students have 3 inversion count.
4 students have 2 inversion count.
3 students have 1 inversion count.
```

2. Integer multiplication

```
Testcase: 12342342352354534553346356, 3045603456834760346356356 5
375898805337690381498432772995135242849587467119140
375898805337690381498432772995135242849587467119140
Testcase: 538494504947763946119211 52, -506335097398631071777706 22
-272658667611353679245106205373101817881793580599654 4
-272658667611353679245106205373101817881793580599654 4
Testcase: 25069.51, 77777.321
Not an integer
-1
Not an integer
-1
Testcase: 0, 1155328940683083679213231
0
0
Testcase: 976722438675602760843724 10, 0
0
0
Testcase: -2551817852586546680292533, 7.344195291378678e+25
Not an integer
-1
Not an integer
-1
Testcase: -744942133276021507022626 07, -686745019522691471887828 96
511585299859916822197964903851970148328779770196987 2
511585299859916822197964903851970148328779770196987 2
Testcase: 222.01, 0
Not an integer
-1
Not an integer
-1
Testcase: -444.05, 1000.001
Not an integer
-1
Not an integer
-1
Testcase: -36363636, 1515.15
Not an integer
-1
Not an integer
-1
```

Conclusion

1. We have found the inversion count of course choice of students using divide and conquer method, which reduces the time complexity to O(nlogn) from O(n^2) of the brute force method. We classified the students according to their inversion count.
2. We have multiplied two integers using the divide and conquer method, which reduces the time complexity to O(nlogn) from O(n^2) of the brute force method. We reduced the number of recursive calls to the function, which increased the efficiency. This algorithm can be used to multiply numbers of large size efficiently.